UiA University of Agder

Master

Machine learning and IMU: Smart Gait applications for use in real-time systems

Gait Detection and Gait Identification based on single-sensor IMU data

Martin Bjørklund Gresli

SUPERVISOR
Dr. Filippo Sanfilippo

University of Agder, 2024
Faculty of Engineering and Science
Department of Engineering and Sciences

Acknowledgements

Thank you

The author of this thesis would like to extend gratitude to some individuals that were instrumental in both my scientific endeavours and personal growth.

To my team at UIA:

Hamza Zafar, for your genuine interest in my work and for being an excellent sparring partner in problem solving.

Dr. Filippo Sanfilippo, for letting me form my own thesis and work with research questions that genuinely interest me. Your ability to provide insight to the scientific literature is second to none.

To my friends at UIA:

To **JE**, for being not only interested in my work, but also capable of delving into complex topics and providing fresh perspectives on my research process.

To **E**, for being a good friend during personal health struggles.

Abstract

Inertial Measurement Units hold a great potential for real-time gait phase measurements, gait kinematics measurement, and gait analysis. They offer convenience in the form of their light weight and portability. There are however known drawbacks in the form of sensor drift and susceptibility to noise.
Extrapolating data for gait analysis directly from IMU's can therefore be problematic, but emerging Machine learning / Artificial intelligence (ML/AI) modalities holds the potential of estimating gait characteristics to a new level of accuracy. Applications such as interpolation of user intent in prosthetic device control could through this be improved further, as well as movement analysis, and gait identification (GI) applications.

In this thesis, IMU sensor data is used in combination with machine learning (ML) models to extrapolate significant information about the users walking gait. The information is extrapolated implicitly, through Machine Learning algorithms. The is centered around two applications: Gait detection (GD, i.e Gait phase detection), and Gait Identification (GI, i.e identify human subjects based on their walking gait). Both applications is already present in the scientific literature. The computational load for some of the systems is however quite high, and not fit for a live system running on a microcontroller or a mobile device. The proposed solution attempts to eliminate some of the computational load by reducing the dimension of input data, and reducing ML model complexity.

The work yields two implementations of Machine Learning on a Gait Identity task, and includes discussion and analysis of important features for a ML Gait detection task for a dataset consisting of 18 features.

For the first Gait Identity task, a TCN ML model architecture scored a perfect 1.0 on a Max F1-Averaged Stratified k-Fold validation for k=5 on a self-acquired single-IMU dataset with 13 subjects.

On the second Gait Identity task, a TCN ML model architecture scored a 0.98 on a Max F1-Averaged Stratified k-Fold validation for k=5 on a publicly available single-IMU dataset with 30 subjects.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis the combination of wearable sensors and ML is explored through the scope of two applications: Gait Identification (GI), and Gait Detection (GD). The two applications recognize the identification of the user, and the users progression in the gait cycle respectively.

Discussing two different applications in the same body of work is done with the overarching goal of combining IMU data with ML to demonstrate valuable real-time applications.

There are several possible applications of GI and GD.

In the field of biomechatronics, GD is frequently used in control of prosthetic devices. For a trans-tibial amputee, an IMU can be placed on the users thigh just above the prosthetic device, and the data can be used to interpolate user intent. The more advanced and accurate such a system is, the more seamless the user experience **??**. Optimally such a system would perform multiple interpolations of the IMU data: Activity recognition, and gait phase detection. The prosthetic device can be fed a reference signal based on the type of activity (i.e standing, sitting, walking, running) and then within the walk/run classes have the system output an estimation of cycle progression for accurate motor control. Such a system could be realised through several architectures. An ensemble method where you have a ML model for determining the current activity, and then having dedicated ML models to output information about the respective activities is considered a viable solution.

GI hold potential for security- and health applications. Developing ML models capable of identifying subject-specific gait characteristics should also yield functionalities such as identifying gait disorders, or identifying deviations from a users normal gait pattern (sudden illness, early signs of neurological deceases, falling or other source of impact). Imagine a model trained for this purpose, and loaded on to a mobile device connected to a wearable sensor. Such a setup could warn either the user or selected people (family, health care professionals) about an event or development in the users gait pattern. This could provide some security to patients without encroaching too much on his/hers privacy, as IMU data is a fairly non-invasive way of monitoring for falls, accidents, and long-time health development. The combination of ML and IMU data could also be used as a security check. One could potentially load a ML model on to a mobile device, and use the onboard IMU as input for the system. If the user is carrying the mobile device in his or hers pocket, the scenario is very similar to having a IMU attached to the thigh. If the ML model does not recognize the gait pattern of the subject carrying the device, the device is wiped for data remotely. In a similar application, IMU 'key cards' could be worn on the pant leg by the staff at high-security facilities. As a worker approaches a door, an ML model embedded in the security system receives and analyses the IMU data. If the workers walking gait is recognized, and the worker in question has been given access to the specific door, the door is unlocked.

The different stages of data acquisition, data processing, and ML model design needed to create such applications from IMU data is demonstrated and discussed in this thesis.

Chapter 2

Related Works

Newly emerged modalities in Artificial intelligence is already being combined with wearable sensors to give yield applications for real-time user motion measurements.

In [18] they combined the data from a conventional visual motion capture system with the data from force plates and IMU's, in an effort to predict user joint moment and joint angles. As the data set was somewhat limited, additional IMU data was simulated based on the optoelectronic motion capture data from past experiments. The simulated IMU data was generated with different sensor positions and orientations, making the model able to generalize the data across varying sensor placements. A mean correlation coefficient of 0.85 for the joint angles and 0.95 for the joint moments was achieved. Such an approach holds great potential, as one could use already existing visual motion capture data to complement the data set of less comprehensive studies.

In [14], Alcaraz and others compared the fused signals of four IMU's to a Machine Learning (ML) -approach using only one IMU placed on the foot. They extrapolated hip, knee and ankle -joint angles from a traditional sensor-fusion approach, normalized to a gait cycle spanning from Initial Contact (IC) to next IC. The data was segmented in to gait cycles by defining events based on the angular movement of the foot, and each cycle was then re-sampled to the length of 100 samples. Through the use of neural networks, they were able to predict the hip, knee, and ankle joint angles to an average RMSE of 1.91°, 2.12°, and 2.57° respectively, based on the signal from only one IMU.

In 2017, Dezangi, Taherisadr and ChangalVala published their work on human gait identification using data from five IMU's. The application of AI and IMU's is slightly different, but the techniques used are highly relevant. They propose a novel time-frequency (TF) expansion in order to extrapolate cyclic gait data from continous measurements. The joint temporal and spectral (2D data) is fed to a Deep Convolutional Neural Network. The network is then trained to extract discriminative features, and to jointly optimize an identification model as well as the spectro-temporal feature extraction [28].

In the work done by Vu, Gomez Et. Al in [26] they extrapolated a highly accurate prediction of the progression of the gait cycle, expressed in the form of percent:

" We propose a model that can predict the gait percentage that was equally divided into 100 one-percent fragments. To achieve this, we reused the method from [44] to segment input signals and label output targets. We began by extracting the lengths of the walking steps from one heel-contact to the next. Then, we sampled each heel-strike window with an interval of 10 ms. This resulted in

several signals that were stored as a matrix X of dimension Rp×(sd), where p is the percentage value, s is the number of sensors, and d is the number of dimensions in an IMU sensor.

" From [26]

They proposed a novel ANN in combination with data from a single IMU attached to the lower shank of seven healthy subjects. The purpose of the system is to provide accurate gait cycle data for powered prostheses. They produced a 100% prediction accuracy thanks to what they call a Exponentially Delayed Fully connected Neural Network (ED-FNN). They were kind enough to share their data, and attempts to create an ML-arcitetchture capable of predicting gait phase was attempted. This proved to be a complex task to solve, and had to be abandoned.

Chapter 3

Theory

3.1 Human walking gait

In this thesis we investigate the application of ML in order to extrapolate more usable information form single-source IMU data.

The references to gait phases, or gait events, is done through the definitions of Jacqueline Perry. See figure 2 in [11].

Typically, walking gait data is located in the frequency range 0.5 to 3 Hz. The work in this thesis uses a similar filtering technique to that in [29].

3.2 Madwick Filter for Sensor Fusion of IMU signals

For applications where accurate predictions of sensor attitude is of importance, sensor fusion algorithms are often used. The concept is centered around eliminating sources of inaccuracy by utilizing the strengths of the different signals in a 6 or 9-axis IMU sensor.

The Madwick filter is a popular algorithm for sensor fusion. It can be used on inertial data only(6 axis), or in a combination of inertial and magnetic data (9-axis). Sebastian Madwick's article from 2010, "An efficient orientation filter for inertial and inertial/magnetic sensor arrays" has been cited 1115 times at the time of writing this report [17].

In appendix A of his article, Madwick lists an implementation of the filter in C language. The implementation has been optimized for computational efficiency. The optimization strategy is based on reducing the number of arithmetic operations, at the expense of increased memory usage.

For each update of the Quaternion values, the filter executes 109 scalar arithmetic operations, including 11 divisions and 3 square roots. This code was adopted, and tweaked to work in a Python language environment for the work in this thesis. More on this in section 4.2.1.1.

3.3 Feature selection for Machine Learning Tasks

3.3.1 Random Forest classifier

This algorithm is praised for it's ability to work well even with a high number of variables, and find sparse, non-linear relationships in complex datasets [15][1]. The model is therefore a good way to avoid the curse of dimensionality in problems with a high number of variables, more on this in section 3.3.4.

3.3.2 Support Vector Machines (SVM)

A linear classification model. SVM's can through optimizing the margins between different classes in the features space effectively determine which features improve model performance. The SVM was selected in this work, due to it's well established ability to detect linear relationships in the data [4].

3.3.3 Other classifiers/feature selction algorithms

Principal Component Analysis (PCA): Widely used as a dimensionality reduction technique. Transforms the original datainto a set of linearly uncorrelated components. K-Nearest Neighbors (KNN): Operates on the assumption that instances of a class is surrounded by other instances of the same class. Linear Discriminant Analysis (LDA): Typically used for dimensionality reduction in preprocessing, aiming to project data into a lower-dimensional space with good class separability [4].

3.3.4 The curse of dimensionality

The curse of dimensionality is a term used to describe the problems that often occurs when the number of dimensions in a dataset increases. As the number of dimensions increase, the number of data points needed to fill the space increases exponentially. Imagine a 1 dimensional stream of data. Only a few points are needed to populate it. As the line l becomes a square (2 dimensions), and the square becomes a cube (3 dimensions), the area to populate has grown by a factor of l^3. The vast space created by the increase in dimension, makes the data very sparse, which in turn makes it hard to classify the points of data.

The increase in dimension also puts higher computational cost on processing the data.

Overfitting, a common problem in ML tasks, can also be caused by having high dimensional data. A ML model with many features are more prone to learn noise or insignificant details of the data rather than the actual relationships between the features and the label values.

Lastly, working with data of high dimension makes visualizing the data a complicated procedure [5].

3.4 Machine Learning

3.4.1 Artificial Neural Networks

Artificial Neural Networks (ANN's) has btheir orgins in the 1940's. The inspiration for these was to simulate the function of the human brain. The initial concept was proposed by McColloch and Pitts, a neurophysiologist and a logicial, in 1943 [19].

It is also worth noting the breaktrhough made by Rumelhart, Hinton and Williams in 1986. They proposed a backpropogation algorithm that spiked the interest for ANN's, as the allowed multi-layer networks to learn efficiently [21].

A deep-dive in to the mechanics of Machine Learning algorithms is beyond the scope of this thesis, but some words about the mathematics behind the output of a neuron is in order. IBM.com provides a very intuitive explanaiton of this, as well as a fairly simple equation for the output of a neuron [13].

$$f(x) = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{3.1}$$

In equation 3.1:

$f(x)$ is the output of the node.
σ represents the activation function.
w_i are the weights associated with each input.
x_i are the input values.

b is the bias.
The summation runs over all n inputs to the node.

3.4.2 The parameters of an ANN

Given a classic neural network the input layer (attributes(x)) is connected to the second layer, and all the activations in the second layer is the sum of the attributes(x) multiplied with their given weights(w) plus a bias(b). For simplicity one can first imagine the second layer to be a linear combination of the attributes(x), the weights(w) and the biases(b).

Imagine trying to predict the presence of precipitation based on a barometer reading and a simple yes/no for the presence of clouds. Our two attributes $x_1 = atmosphericpressure$ and $x_2 = cloudsvisable$. The output is simply either rain (1) or no rain (0). A single output y_1. Layer 2, a hidden layer, contains two activations. Both these activations is a combination of the two attributes, and both activations have a weight assigned to both attributes. Lastly, a bias is added to each activation in order to further adjust the resulting calculation with inputs = attributes(x). Now the second layer holds two functions, each of them tune-able, that provides the model with new features it can use to approximate the desired output(y). Our single output, y_1, is a combination of the two activations in the hidden layer. A weight is assigned to the two activations connection to y_1, and a bias is used here as well. A system such as this is perhaps capable of predicting precipitation, even though it lacks non-linearity in it's calculations. To add this complexity, an activation function is added to each activation. To explain this further, it might be useful to look at the mathematical theorem of Universal Approximation.

3.4.3 ANN's and the Universal Approximation Theorem

To understand how an AI model can solve complex tasks, it can be valuable to have some understanding of the mathematical theorem of Universal Approximation.

Neural Networks (NN's) are simply a collection of neurons (nodes) that are connected through various layers. The goal of the network is to have it map attributes(x) to outputs(y), hence finding a mathematical function $y = f(x)$ [22].

The Universal Approximation Theorem tells us that ANN's have a kind of universality, given that a network exists that can approximately approach the result of a given $f(x)$, and effectively perform the needed calculation. This holds no matter what $f(x)$ actually is, and for any arbitrary number of inputs and outputs. [22].

The non-linearity problem must however be addressed in order to satisfy the theorem. An ANN is only able to handle nonlinearities if there is an activation layer behind at least one linear layer in the network. [16]. Another way of saying this is that an ANN only holds the Universal Approximation property for representing probability functions if the network holds non-linear capability.

In order to allow an activation to extract nonlinear features, an activation function is added.

3.4.4 Activation functions

If an AAN is to find non-linear dependencies between attributes(x) and output/outputs (y), an activation function must be used behind a linear layer in the model. This enables the network to approximate any given function, given the appropriate parameters.

Popular activation functions are Rectified Linear Unit (ReLU) and the Sigmoid function

$$Sigmoid(x) = (1 + e^{-x})^{-1}$$

[16].

3.4.5 Model evaluation

3.4.5.1 Holdout method

For model evaluation, a common technique is withhold parts of the data from training. This data can then be used to test the data. i.e. this data is only showed to a static version of the model, that does not adjust it's parameters based on the outcome of the predictions made on the data. This provides an unbiased assessment of the trained models ability to generalize to data it has not seen before.

3.4.5.2 F1 score

The F1 score metric for evaluating ML models is a technique that combines the metrics precision and recall.

Precision is the ratio of correctly predicted positives for a given label class with the total number of predicted positives. Equation 3.2

Recall, or sensitivity, is the rate of correctly predicted positives for a given class, divided by the total number of observations for the given class. Equation 3.3.

The F1 metric is the harmonic mean of precision and recall, and so it takes both false positives and false negatives into account. The F1 score ranges from 0 to 1. Equation 3.4, [6].

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.3}$$

$$F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{3.4}$$

3.4.5.3 K-fold stratified cross-validation

A k-fold cross validation splits the data into several folds, each with it's own test and training sets. Within each fold, it works exactly like the hold out method, in the way that the data is trained on one part of the data, and validated on another. Stratification of the folds simply mean that the folds and the test/train splits are done in a way that preserves the percentage of samples for each class. For a task like GI, this is appropriate [7].

3.5 Machine Learning architectures

3.5.1 LSTM - Long Short Term Memory network

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN). LSTM's are designed to overcome some of the limitations of traditional RNN's. The key difference is that LSTM's are able to hold information for long periods of time. LSTM utilize memory cells to achieve this effect. Another feature of LSTM is that they have gate mechanisms: Input gate, forget gate, and ouptut gate. These mechanisms control what information is kept and discarded, and allows these networks to yield good results for certain memory-demanding tasks [3].

3.5.2 CNN - Convolutional Neural Network

A class of deep neural networks, primarily used in analyzing visual imagery. Their ability to form abstractions of the input data is however very useful in other applications as well. The convolutional operation done can be seen as a feature extraction, in the way that it allows for quick sorting of the relevant features, or relevant combination of features [12].

3.5.3 TCN - Temoral Convolutional Network

Temporal convolution leverages the power of convolution along the temporal dimension. In this thesis, the use of TCN yields good results. This is not surprising, as looking at the input data from multiple levels of temporal resolution should provide good insights in the different characteristics of the features as time passes. For tasks where a sequence of data points along the temporal axis is fed to the ML model, this sort of arcitechture makes a lot of sense. In the scientific literature, TCN have been proved to perform well on prediction tasks with time-series data [20].

Chapter 4

Method

4.1 Data acquisition

4.1.1 UIA IMU dataset for Gait Identification

IMU Data was collected from 13 healthy adults, 12 male 1 female, as they performed a simple experiment consisting of walking up and down a hallway two times. The hallway's length was estimated by walking its entire span, measuring approximately 40 paces. The data collection was done at UIA Campus Grimstad. All the subjects agreed to their walking gait data being collected and used as part of research. No information about the subjects identity is carried past the point of data collection, i.e. the names of the subject are not used in the code for data processing or machine learning.

Figure 4.1: Walking Gait experiment protocol

The data was collected with the sensor positioned on the right thigh. Sensor was fastened with the elastic band included in the EmotiBit Essentials Kit [8]. Sensor orientation: The back of the EmotiBit board lay normal to the outside of the right thigh, i.e. the board in the upright position with the on/off button pointing down, and the face of the board pointing out, normal to the walking direction. The sensor position was not identical for all subjects, but the sensor orientation was similar for all recordings. The slight variations in sensor position is caused by variance in the subjects anatomy, as well as some variation due to the sensor moving during the walking experiment. These differences where however kept as small as possible, by re-adjusting the sensor placement if it slid down or otherwise deviated from the described position. The sensor used is an EmotiBit multi-sensor, with a high-quality IMU by Bosch, the BMI160, on its integrated circuit board. The EMotiBit is connected to a computer via Wifi for remote control of recording, and the data is recorded on a sd memory card onboard the EmotiBit board [2][8].

4.2 Data processing

4.2.1 UIA IMU dataset for GI

The data processing of the UIE IMU data can be viewed in it's entirety in Appendix A.1, A.2 and D.2.

The data collected from the EmotiBit board can be extrapolated from the onboard memory card by inserting the card in to a computer and running the recorded file in the EmotiBit dataparser [9]. The result is then a per-measurand file system containing measurand, timestamp, and other information. The 9 files containing information from the BMI160 accelometer, gyroscope, and magnetometer is then saved with an assigned number identifying the subject at the beginning of the file name. This process is done after each subject has undergone the experiment.

The process of combining all measurands for each subject in to a subject-specific Python Pandas Dataframe is included in it's entirety in Appendix A.1. The list of files and part of the code used to sort the file into subject-spesific dataframes are listed in listing 4.1.

Listing 4.1: Wrangling files from all 9 axis, all 13 subjects, to a list of pandas dataframes

```
    import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

/kaggle/input/uia-imu-gait-analysis-dataset/0_MX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/9_MZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/3_MZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/7_MX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/6_AZ.csv
..... List goes on, cut for convenience.

list_files  = [
'/kaggle/input/uia-imu-gait-analysis-dataset/0_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/2_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/4_MY.csv',

.... List goes on, list used in this function:

df_dict = {} # to store a list of subject-specific dataframes

for item in list_files:

    last_slash = item.split('/')[-1]

    indexer_s = last_slash.split('_')[0] #extrapolates subject number ...
       from file

    indexer = int(indexer_s)

    measurand_csv = last_slash.split('_')[1]

    # extrapolates measurand from file example:
    # 1_MZ = magnetic measuremnt in Z direction for subject 1)

    measurand = measurand_csv.split('.')[0]

    File = pd.read_csv(item)

    Time_Series = File.iloc[:, 0]
    Time_Series.name = f'{measurand}_timestamp'

    Measurand_Series = File.iloc[:, -1]
    Measurand_Series.name = measurand


    if f'df_{indexer}' not in df_dict:

        df_dict[f'df_{indexer}'] = pd.concat([Time_Series, ...
           Measurand_Series], axis=1)

    else:

        df_dict[f'df_{indexer}'][Time_Series.name] = Time_Series
        df_dict[f'df_{indexer}'][Measurand_Series.name] = Measurand_Series
```

This way of processing the data puts all the relevant measurands in a subject-specific pandas dataframe and store them in a Python dictionary. Given the potential for differences in sensor processing times, data readout intervals, and synchronization mechanisms within the Bosch BMI160 device, an analysis of timestamp synchronization across measurands was performed.

Listing 4.2: Timestamp synch analysis

```
Erroneous_Lines = {}

def Check_alignment(df_dict):
    for key, df in df_dict.items():
        time_cols = [column for column in df.columns if ...
            column.endswith('_timestamp')]

        for i in range(len(df)):
            timestamps = [df[column].iloc[i] for column in time_cols]

            if len(set(timestamps)) != 1:
                if key not in Erroneous_Lines:
                    Erroneous_Lines[key] = []
                Erroneous_Lines[key].append((i, timestamps))

    return Erroneous_Lines

# Run the check
Errors = Check_alignment(df_dict)

if Errors:
    for key, rows in Errors.items():
        print(f'Issues in {key}:')
        for row, timestamps in rows:
            print(f'Row {row}: {timestamps}')
else:
    print("All timestamps are aligned across all df's.")
```

The findings can then be printed, Listing 4.3

Listing 4.3: Timestamp synchronization errors

```
for key, rows in Errors.items():
    print(f'Issues in {key}:')
    for row, timestamps in rows:
        max_timestamp = max(timestamps)
        min_timestamp = min(timestamps)
        difference = max_timestamp - min_timestamp
        print(f'Row {row}: Difference = {difference} seconds')

Issues in df_0:
Row 670: Difference = 0.0003440380096435547 seconds
Row 671: Difference = 0.0006880760192871094 seconds
Row 672: Difference = 0.0006868839263916016 seconds
Row 673: Difference = 0.0003440380096435547 seconds
Issues in df_9:
Row 2020: Difference = 0.00017189979553222656 seconds
Row 2021: Difference = 0.0003440380096435547 seconds
Row 2022: Difference = 0.0003437995910644531 seconds
Row 2023: Difference = 0.00017189979553222656 seconds
Issues in df_4:
Row 803: Difference = 0.00017213821411132812 seconds
Row 804: Difference = 0.0003440380096435547 seconds
Row 1042: Difference = 0.0003440380096435547 seconds
Row 1043: Difference = 0.0006880760192871094 seconds
Row 1044: Difference = 0.0006880760192871094 seconds
Row 1045: Difference = 0.0003437995910644531 seconds
Row 2385: Difference = 0.00017213821411132812 seconds
Row 2386: Difference = 0.0003440380096435547 seconds
Issues in df_3:
Row 1372: Difference = 0.00017189979553222656 seconds
Row 1373: Difference = 0.0003440380096435547 seconds
Row 1374: Difference = 0.0006880760192871094 seconds
Row 1375: Difference = 0.0003440380096435547 seconds
Issues in df_8:
Row 466: Difference = 0.00017213821411132812 seconds
Row 467: Difference = 0.0003440380096435547 seconds
Row 468: Difference = 0.0006880760192871094 seconds
Row 469: Difference = 0.0003440380096435547 seconds
Issues in df_1:
Row 134: Difference = 0.00017189979553222656 seconds
Row 135: Difference = 0.0003437995910644531 seconds
Row 136: Difference = 0.0006880760192871094 seconds
Row 137: Difference = 0.0003440380096435547 seconds
Row 1576: Difference = 0.00034308433532714844 seconds
Row 1577: Difference = 0.0006880760192871094 seconds
Row 1578: Difference = 0.0006880760192871094 seconds
Row 1579: Difference = 0.0003440380096435547 seconds
Issues in df_10:
Row 1165: Difference = 0.0006880760192871094 seconds
Row 1166: Difference = 0.0003440380096435547 seconds
```

The discovered differences in timestamp for the different features in the same sample are small, and can safely be neglected for the purpose of this work. The differences are also repeated, which can mean that they are simply the result of some process on the BMI160 multi-sensor, or the EmotiBit board. The next step is to concatenate all the subject-specific dataframes into a combined dataframe, sorted using two criteria in an hierarchical manner: Primarily by Subject, and then by timestamp. In addition, we will create a column that signifies the subject number in the combined dataframe. This will be used as labels for the ML algorithm. For these steps, see listing 4.4.

Listing 4.4: Combining and sorting all data from all subjects, fix timestamp

```
    df_combined = pd.DataFrame()

for subject_key, subject_df in df_dict.items():

    subject_number = int(subject_key.split('_')[1]) #as in df_0 = subject 0

    subject_df['Subject_no'] = subject_number


    df_combined = pd.concat([df_combined, subject_df], ignore_index=False)


    # Timestamp reset for each subject
    df['Time_'] = df.groupby('Subject_no')['Timestamp'].transform(lambda ...
        x: x - x.min())




    ....
```

More processing of the data is necessary, but this is done in prepping the data for the ML models and in the generation of Quaternion and Euler angle expression for IMU sensor attitude.

4.2.1.1 Generating Quaternion and Euler angular expressions for IMU attitude on the UIA IMU dataset using the Madwick filter

The code for generating Quaterion and Euler angle expressions can be found in Appendix A.2.

The Madwick filter and the source for the code is described in section 3.2. The implementation of the filter is listed below. The attitude calculations are a multi-variable problem, and tuning the filter is considered part of the process. The beta variable regulates the balance between gyroscope and accelerometer readings, and finding a good value for this is done through experimentation. The initial value should however be based on some hard data from the datasheet [2]. The First lines of code in the filter implementation utilizes this information. The Madwick filter, listing 4.5

Listing 4.5: Implementing the Madwick filter in Python Language

```
'''
Madwick filter for generating quaternion expression for IMU attitude
'''


# Need to find constant for Gyro measurement error
pi = np.pi

BMI_Sensitivity_Rfs150 = 131.2 #LSB per deg/s, from BMI160 datasheet

BMI_Sens = BMI_Sensitivity_Rfs150 *(pi/180) #convert ~ 2.29 rad/s

Desired_Accuracy_Deg = 2.5
Desired_Accuracy = Desired_Accuracy_Deg * (pi/180) #convert ~ 0.0436 rad/s

Gyro_Meas_Error = BMI_Sens * Desired_Accuracy # 2.29 * 0.0436 ~ 0.099 r/s

GME = Gyro_Meas_Error #shorts are nice



'''
Global
'''

acc_X, acc_y, acc_Z = 0.0, 0.0, 0.0

gyr_X, gyr_Y, gyr_Z = 0.0, 0.0, 0.0

mag_X, mag_Y, mag_Z = 0.0, 0.0, 0.0

SE_q0, SE_q1, SE_q2, SE_q3 = 0.0, 0.0, 0.0, 0.0

beta = 1.5 * GME

#beta = 0.001 * GME


'''
Initialize the filter
'''


def Madwick_Init():

    global SE_q0, SE_q1, SE_q2, SE_q3
```

```
    SE_q0 = 1.0
    SE_q1 = 0.0
    SE_q2 = 0.0
    SE_q3 = 0.0


#Madwick_Init() # initialized


def Madwick_quat(dt, acc_X, acc_Y, acc_Z, gyr_X, gyr_Y, gyr_Z):

    global SE_q0, SE_q1, SE_q2, SE_q3 # this eliviates the need for return

    # normalize acc and mag measurements

    acc_N = math.sqrt(acc_X**2 + acc_Y**2 + acc_Z**2)

    if acc_N != 0:

        acc_X, acc_Y, acc_Z = acc_X / acc_N, acc_Y / acc_N, acc_Z / acc_N

    #mag_N = math.sqrt(mag_X**2 + mag_Y**2 + mag_Z**2)
    #mag_X, mag_Y, mag_Z = mag_X / mag_N, mag_Y / mag_N, mag_Z / mag_N

    # Variables for computational easing

    hf_q0 = 0.5 * SE_q0

    hf_q1 = 0.5 * SE_q1

    hf_q2 = 0.5 * SE_q2

    hf_q3 = 0.5 * SE_q3

    tw_q0 = 2.0 * SE_q0

    tw_q1 = 2.0 * SE_q1

    tw_q2 = 2.0 * SE_q2

    tw_q3 = 2.0 * SE_q3

    fr_q1 = 2.0 * tw_q1

    fr_q2 = 2.0 * tw_q2

     # Gradient descent algorithm corrective step
    s0 = tw_q1 * SE_q3 - tw_q0 * SE_q2 - acc_X
    s1 = tw_q0 * SE_q1 + tw_q2 * SE_q3 - acc_Y


    s2 = 1.0 - tw_q1 * SE_q1 - tw_q2 * SE_q2 - acc_Z
    # simplified 1.0 - 2.0 * SE_q1 * SE_q1 - 2.0 * SE_q2 * SE_q2 - acc_Z

    '''
    State Vector:
    | s0 |
    | s1 |
    | s2 |
```

```
'''

# Compute the gradient (Jacobians)
J_11or24 = tw_q2  # J_11 negated in matrix multiplication
J_12or23 = tw_q3 # replaced 2.0 * SE_q3 with a placeholder
J_13or22 = tw_q0  # raplaced 2.0 * SE_q0
J_14or21 = tw_q1  # J_13 negated in matrix multiplication
J_32 = fr_q1  # replaced 2.0 * J_14or21 with 2 * 2 * q1
J_33 = fr_q2  # replaced 2 * J_11or24 with 2 * 2 * q2

'''
    Jacobian Matrix:
| J_11or24 J_12or23 J_13or22 J_14or21 |
| 0        0        0        J_32     |
| 0        0        0        J_33     |

'''


'''
J_11or24 = two_q2  # J_11 negated in matrix multiplication
J_12or23 = 2.0 * SE_q3
J_13or22 = two_q0  # J_12 negated in matrix multiplication
J_14or21 = two_q1  # J_13 negated in matrix multiplication
J_32 = 2.0 * J_14or21  # negated in matrix multiplication
J_33 = 2.0 * J_11or24




'''









# Compute the gradient (matrix multiplication)
SEqHatDot_0 = J_14or21 * s1 - J_11or24 * s0
SEqHatDot_1 = J_12or23 * s0 + J_13or22 * s1 - J_32 * s2
SEqHatDot_2 = J_12or23 * s1 - J_33 * s2 - J_13or22 * s0
SEqHatDot_3 = J_14or21 * s0 + J_11or24 * s1

# Normalize the gradient
norm = math.sqrt(SEqHatDot_0**2 + SEqHatDot_1**2 + SEqHatDot_2**2 + ...
    SEqHatDot_3**2)

if norm != 0:

    SEqHatDot_0 /= norm
    SEqHatDot_1 /= norm
    SEqHatDot_2 /= norm
    SEqHatDot_3 /= norm
```

```
'''
 hf_q0 = 0.5 * SE_q0

hf_q1 = 0.5 * SE_q1

hf_q2 = 0.5 * SE_q2

hf_q3 = 0.5 * SE_q3
'''

# Compute the quaternion derivative measured by gyroscopes

SEqDot_omega_0 = -hf_q1 * gyr_X - hf_q2 * gyr_Y - hf_q3 * gyr_Z
#SEqDot_omega_1 = -halfSEq_2 * w_x - halfSEq_3 * w_y - halfSEq_4 * w_z;

SEqDot_omega_1 = hf_q0 * gyr_X + hf_q2 * gyr_Z - hf_q3 * gyr_Y
#SEqDot_omega_2 = halfSEq_1 * w_x + halfSEq_3 * w_z - halfSEq_4 * w_y;


SEqDot_omega_2 = hf_q0 * gyr_Y - hf_q1 * gyr_Z + hf_q3 * gyr_X
#SEqDot_omega_3 = halfSEq_1 * w_y - halfSEq_2 * w_z + halfSEq_4 * w_x;


SEqDot_omega_3 = hf_q0 * gyr_Z + hf_q1 * gyr_Y - hf_q2 * gyr_X
#SEqDot_omega_4 = halfSEq_1 * w_z + halfSEq_2 * w_y - halfSEq_3 * w_x;



# Compute and remove the gyroscope biases
SEqDot_omega_0 -= beta * SEqHatDot_0
SEqDot_omega_1 -= beta * SEqHatDot_1
SEqDot_omega_2 -= beta * SEqHatDot_2
SEqDot_omega_3 -= beta * SEqHatDot_3



# Integrate rate of change of quaternion to yield quaternion
SE_q0 += SEqDot_omega_0 * dt
SE_q1 += SEqDot_omega_1 * dt
SE_q2 += SEqDot_omega_2 * dt
SE_q3 += SEqDot_omega_3 * dt

# Normalize quaternion
norm = math.sqrt(SE_q0**2 + SE_q1**2 + SE_q2**2 + SE_q3**2)

if norm != 0: # precautionary measure for the case where the sum of ...
    the squares is zero
     SE_q0 /= norm
     SE_q1 /= norm
     SE_q2 /= norm
     SE_q3 /= norm
```

In order to gain correct naming of the axis in the global coordinate system, a simple experiment was conducted. A series of 90 degree rotations was done while recording the sensor output. The time for each rotation was noted so that it could be found through plotting the data post-recording. The axis was then re-defined to fit a Left Hand Coordinate System, X axis pointing in the walking direction, Y perpendicular to walking direction (Right positive), Z perpendicular to both axis with up defined as positive. For rotation, counter-clockwise direction is defined as positive for all three axis.

Listing 4.6: Re-mapping of the axis, UIA IMU Dataset

```
The following orientation is observed through experimentation with the ...
    Emotibit sensor

acc:

Y: Up +

X: FWD +

Z: R +

Gyr:

YAW = Y CCW +

PITCH = Z CCW +

ROLL = X CCW - (invert)

df_1['accX'] = df_1['AX'].copy()
df_1['accY'] = df_1['AZ'].copy()
df_1['accZ'] = -df_1['AY'].copy()+1


df_1['gyrX'] = -df_1['GX'].copy()*(math.pi/180)
df_1['gyrY'] = df_1['GZ'].copy()*(math.pi/180)
df_1['gyrZ'] = df_1['GY'].copy()*(math.pi/180)

# The magnetometer readings are not used in angle calculations
df_1['magX'] = -df_1['MX'].copy()
df_1['magY'] = df_1['MZ'].copy()
df_1['magZ'] = df_1['MY'].copy()

....
```

The filter is then applied to the data, through a loop that re-initiates the filter on a per-subject basis. This effectively ensures that the filter does not calculate across different recording sessions, as this would inevitably cause sudden jumps in the resulting calculations. The application of the filter can be seen in listing 4.7.

Listing 4.7: Applying the Madwick filter on the UIA dataset

```
    import math



'''Variable to keep track of the last subject number = last_subject_no, we ...
   re-initialize the filter for every subject,
so that there is no sudden jumps the values used in our calculations '''

last_subject_no = None

# Iterate through each row of the DataFrame
for index, row in df_1.iterrows():
    current_subject_no = row['Subject_no']

    # Check if the subject number has changed
    if current_subject_no != last_subject_no:

        # If so, reinitialize the filter for the new subject
        Madwick_Init()
        last_subject_no = current_subject_no

    # Extract the necessary values from the row
    dt = row['dt']
    acc_X = row['accX']
    acc_Y = row['accY']
    acc_Z = row['accZ']
    gyr_X = row['gyrX']
    gyr_Y = row['gyrY']
    gyr_Z = row['gyrZ']

    # Apply the filter
    Madwick_quat(dt, acc_X, acc_Y, acc_Z, gyr_X, gyr_Y, gyr_Z)

    # Store the updated quaternion values
    df_1.at[index, 'q0'] = SE_q0
    df_1.at[index, 'q1'] = SE_q1
    df_1.at[index, 'q2'] = SE_q2
    df_1.at[index, 'q3'] = SE_q3


    .....
```

The result can be analysed through some random samples from the pandas dataframe. This process in it's entirety is included in Appendix A.2, for simplicity only the interval index is included here:

```
df_subset = df_1.loc[5400:5600] #Random subset 1
df_subset = df_1.loc[8400:8600] #Random subset 2
df_subset = df_1.loc[33000:33200] #Random subset 3
```

These intervals are then plotted, see figures 4.2, 4.3 and 4.4.

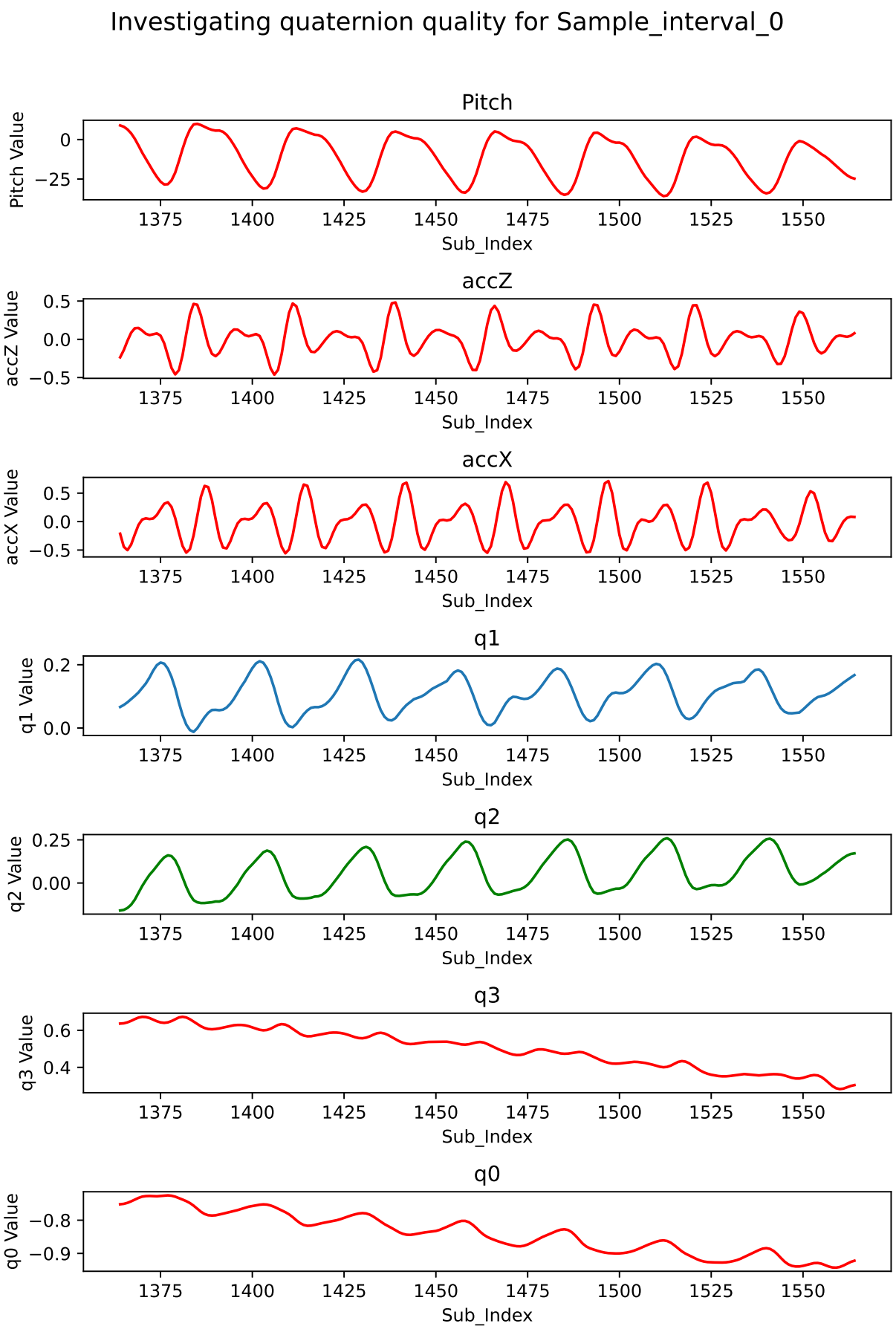


Figure 4.2: Plotting Quaterion and Euler for Random subset 1

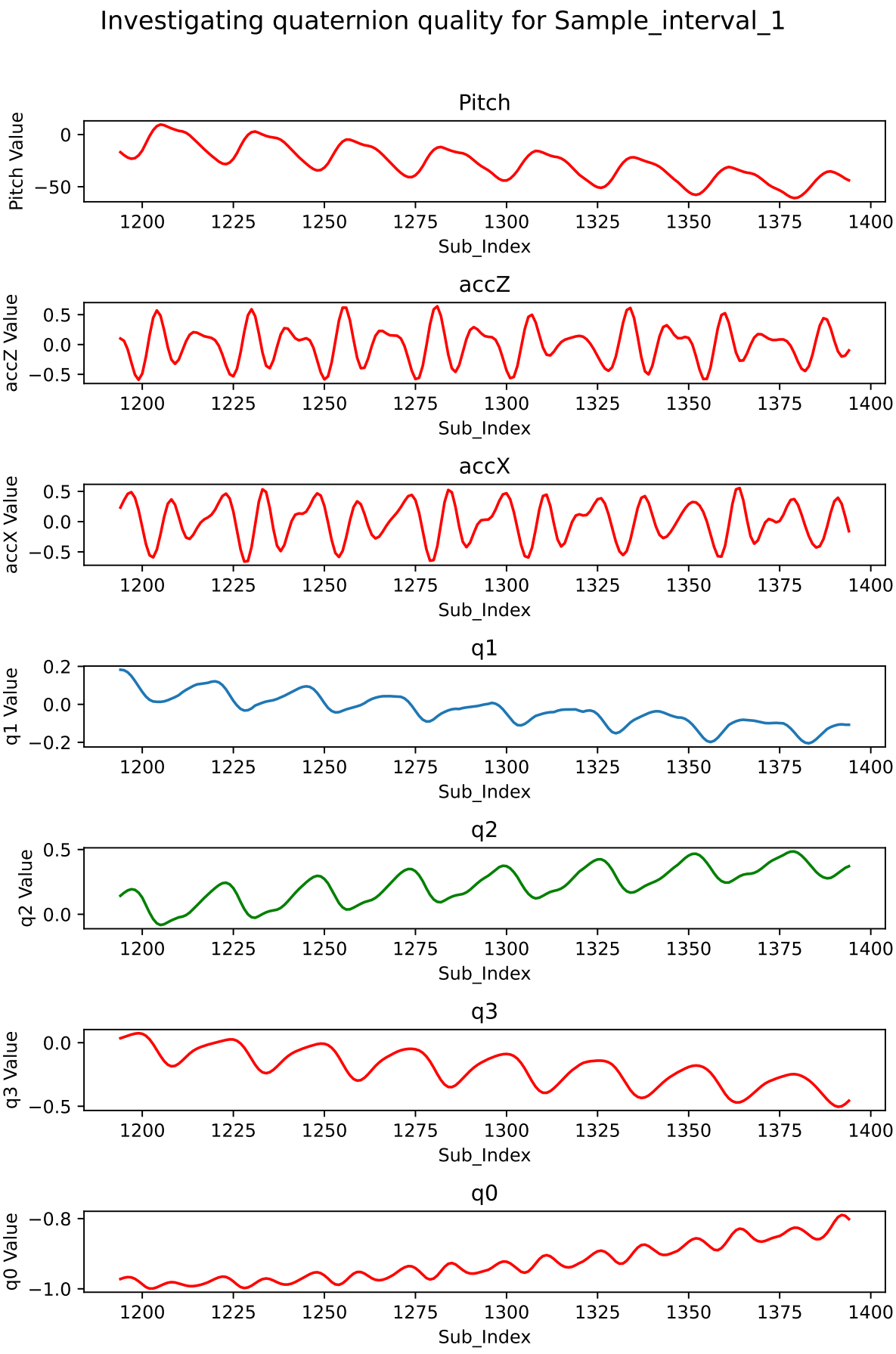


Figure 4.3: Plotting Quaterion and Euler for Random subset 2

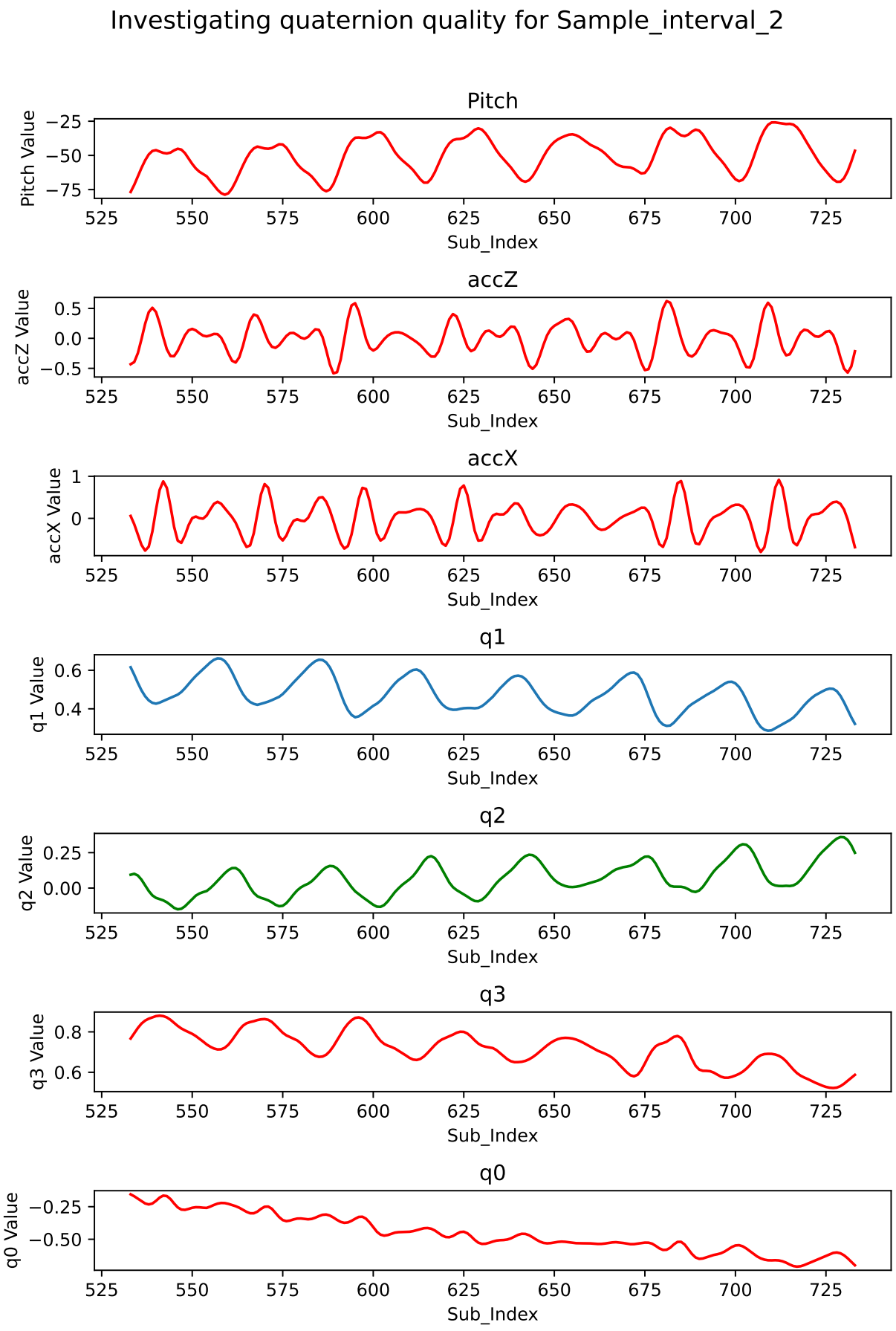


Figure 4.4: Plotting Quaterion and Euler for Random subset 3

The importance of these expressions vary, but in gait analytics the use of pitch angle is highly relevant. The impression from the sampled data is that the characteristics of the Pitch angle resembles what we expect to see from a hip or thigh -angle during walking [18][27], and so this result is deemed satisfactory.

It is important to note that even though the Pitch angle attitude of a sensor placed on the outside of the thigh will be related to the hip angle of the subject, the expression is in fact not the hip angle. Without other sensory input to map rotation around this axis directly to differences in thigh, shank and upper body -angles, this measurand can only be used as an indirect representation of hip angle. This observation is at the heart of the motivation of combining IMU and ML: The single-sensor data is lacking the complexity to explicitly describe complex gait characteristics, but through the use of Machine Learning we might be able to extrapolate this information implicitly. The level of detail on the information will depend on the types of labels available. One could potentially label many kinds of gait characteristics.

4.2.2 Topman et. al dataset for GI

The dataset can be accessed through an online database [25]. The dataset contains the files listed in listing 4.8

Listing 4.8: The Topman Dataset Content

```
leg_train_raw
Raw sensor data, columns are the data points provided by the sensor. Train ...
    set. Sensor attached to leg.


leg_test_raw
Raw sensor data, columns are the data points provided by the sensor. Test ...
    set. Sensor attached to leg.

leg_train_features
Windowed data with extracted features. Train set. Sensor attached to leg.

leg_test_features
Windowed data with extracted features. Train set. Sensor attached to leg.

arm_train_raw
Raw sensor data, columns are the data points provided by the sensor. Train ...
    set. Sensor attached to arm.

arm_test_raw
Raw sensor data, columns are the data points provided by the sensor. Test ...
    set. Sensor attached to arm.

arm_train_features
Windowed data with extracted features. Train set. Sensor attached to arm.

arm_test_features
Windowed data with extracted features. Train set. Sensor attached to arm.
```

For the work done by Topman and colleagues, they used a combination of feature engineering ([10]) and a thorough feature ranking- and selection process to identify a test group of 30 persons with 100% accuracy.

Their method seems scientifically viable and well executed, but is not repeated in this work due to the reliance of calculating statistical features based on the raw data. This method

should in theory be slow and computationally demanding, and not fit for a real-time system.

In addition, they used data from two sensors. One sensor on the arm, and one on the thigh. The sensor placement on the thigh is very similar to the UIA IMU dataset, and therefore very relevant for the proposed applications in this body of work.

To access the maximum amount of data, the test and train datasets
`leg_train_raw`
`leg_test_raw`
are concatenated, and sorted with a similar technique to that in 4.4, only this time a third hierarchical criteria is added: A column indicating whether the data came from the train or test set. This way the data from the `leg_train_raw` set for a given subject is listed in the correct temporal order, followed by the data in the `leg_test_raw` set, this also in the temporal order it was collected.

The data had some incidents of nan-values, these were handled using the technique displayed in Appendix C.1. After these steps, the raw data is prepared for the ML models as described in 4.3.1.

4.2.3 Vu et. al dataset for GD

The dataset used in this study was obtained from Vu and colleagues, who generously shared it upon an email request by the thesis author.

Given the nature in which this data was shared, with no additional information, and no naming of the features, some assumptions had to be made about the data.

The data is labeled for gait detection, through the use of signals form a force sensitive resistor (FSR) [26]. This approach is considered valid, as segmenting the data in to steps can be done by sampling the data between each IC (Initial Contact) event detected by the force sensor (IC: See figure 2 in [11] for reference). The labels are from the looks of things in three different levels of resolution:

$$[0, 3], [1, 10], [1, 100]$$

These label columns was named "*GaitPhase*", "*Desi*" and "*Percent*", respectively.

Some statistical tools were used to analyse the features in the dataset, see Appendix B.1. Problematic parts of the data, like NAN values, were removed.

Listing 4.9: Statistical analysis, features in Vu dataset

```
statistical_summary = df.describe(include='all')

print(statistical_summary)

            Sensor_0       Sensor_1       Sensor_2       Sensor_3  \
count  129159.000000  129159.000000  129159.000000  129159.000000
mean      145.375762    1144.768820     -97.515605      -4.377140
std       287.751470     545.118758     468.883905    2080.900275
min     -3416.000000   -1685.000000   -3397.000000   -6666.000000
25%        69.000000     925.000000    -178.000000    -213.000000
50%       161.000000    1020.000000     -36.000000     114.000000
75%       243.000000    1458.000000      48.000000     694.000000
max      5311.000000    5321.000000    3942.000000    6733.000000

            Sensor_4       Sensor_5       Sensor_6       Sensor_7  \
count  129159.000000  129159.000000  129159.000000  129159.000000
mean      -61.833206      66.911853    -350.821569    2111.428379
std       602.765886     864.597088     844.428760     663.933055
```

```
min     -4027.000000   -4884.000000   -5010.000000     -63.000000
25%      -220.000000    -276.000000    -550.000000    1934.000000
50%        18.000000     -27.000000    -285.000000    2048.000000
75%       179.000000     342.000000       5.000000    2337.000000
max      3398.000000    3730.000000    7544.000000    5321.000000

            Sensor_8       Sensor_9      Sensor_10      Sensor_11  \
count  129159.000000  129159.000000  129159.000000  129159.000000
mean     -422.422247      13.468183     -35.892543     -18.640435
std       354.337227     629.817378     526.257556    1976.545264
min     -5359.000000   -1601.000000   -3343.000000   -3966.000000
25%      -614.000000    -343.000000    -292.000000   -1067.000000
50%      -464.000000    -115.000000     -44.000000    -655.000000
75%      -262.000000     156.000000     248.000000     577.000000
max      2064.000000    2557.000000    3326.000000    5722.000000

           Sensor_12      Sensor_13      Sensor_14      Sensor_15  \
count  129159.000000  129159.000000  129159.000000  129159.000000
mean     -299.371844     337.864051    2027.746119      57.076712
std      1141.413767     715.887970    1111.972704     874.288824
min    -22026.000000  -10978.000000   -7064.000000   -7490.000000
25%      -471.000000     204.000000    1747.000000    -258.000000
50%      -171.000000     368.000000    1824.000000     -10.000000
75%        11.000000     522.000000    2114.000000     243.000000
max     18462.000000    9873.000000   23833.000000    8282.000000

           Sensor_16      Sensor_17      Sensor_18      Sensor_19  \
count  129159.000000  129159.000000  129159.000000  129159.000000
mean      -25.506399    -120.092916       0.002453      -0.004087
std      2507.278094     574.962212       0.707051       0.707152
min     -7502.000000   -7458.000000      -0.999507      -0.999507
25%      -729.500000    -190.000000      -0.707107      -0.707107
50%        51.000000     -58.000000      -0.031411      -0.031411
75%       545.000000      54.000000       0.707107       0.707107
max     18422.000000    2890.000000       0.999507       0.999507

          Gait_Phase     Time_Milli  Time_Delta_Milli  Time_Delta_Micro  \
count  129159.000000  129159.000000     129159.000000     129159.000000
mean        1.673108  140587.173353         10.385106      10385.106404
std         1.086281   96162.914805          1.753108       1753.107903
min         0.000000       8.835000          5.138000       5138.000000
25%         1.000000   53008.695500          9.099000       9099.000000
50%         2.000000  136179.615000         10.183000      10183.000000
75%         3.000000  219515.475500         10.983000      10983.000000
max         3.000000  361330.840000         21.207000      21207.000000

             Percent           Desi
count  129159.000000  129159.000000
mean       50.749673       5.566294
std        28.794576       2.864884
min         1.000000       1.000000
25%        26.000000       3.000000
50%        51.000000       6.000000
75%        76.000000       8.000000
max       100.000000      10.000000
```

The results listen in listing 4.9 led to the following conclusion (From python notbook, Appendix B.1):

"Suspect Columns 'Sensor 18' and 'Sensor 19' exhibit characteristics of a sinusoidal nature, given that they demonstrate a confined numerical scope [-1, 1]. This pattern may also be the result of a normalization process.

Given that the data was shared without any additional documentation, these columns are simply dropped. This is done to ensure that any successful machine learning implementation is not inadvertently a result of the columns being directly related to the label values, such as being representations of the label values in sinusoidal form."

This suspicion was then confirmed by plotting these features, in addition to plotting them up against one of the label columns. See figure 4.5.

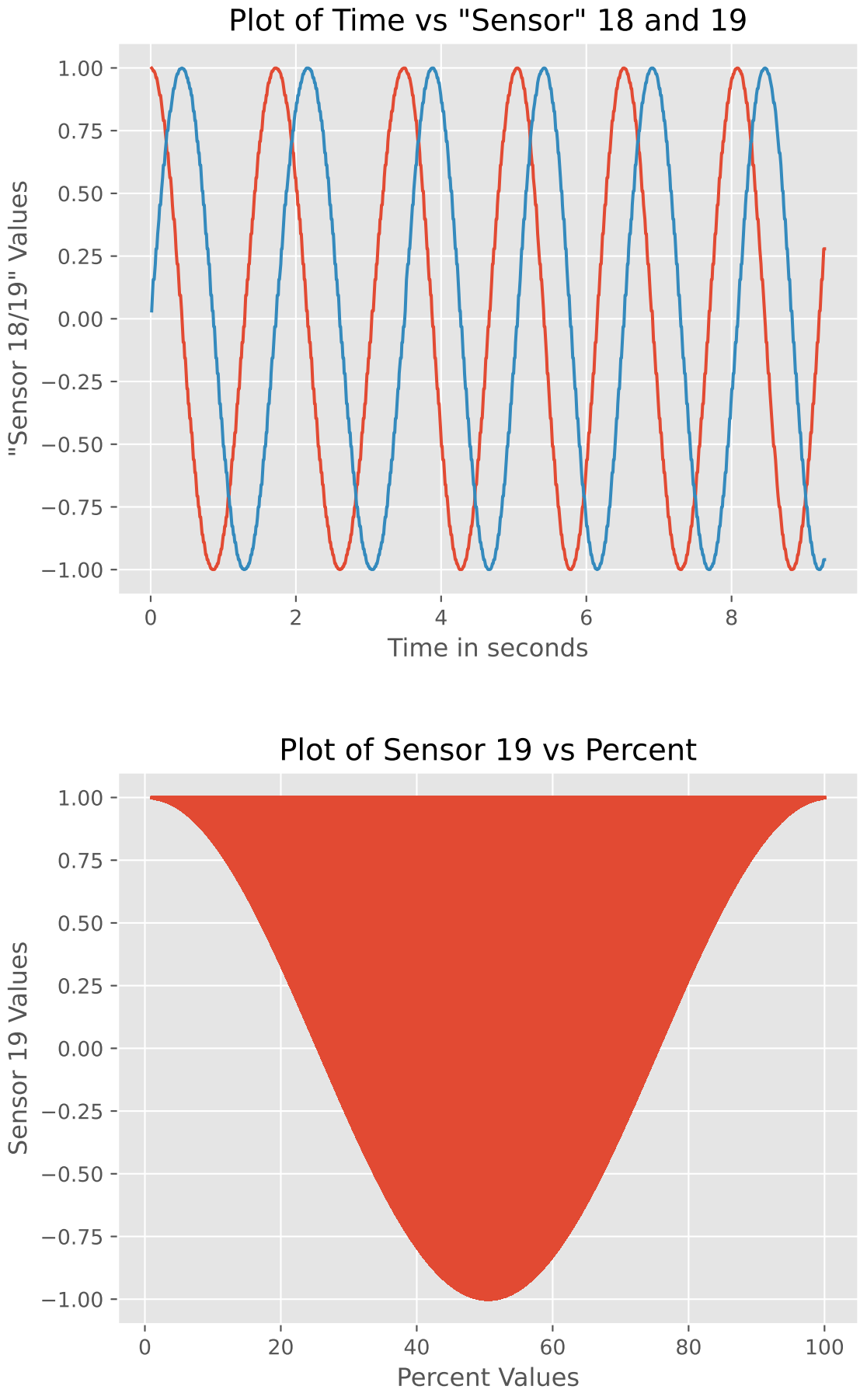


Figure 4.5: Features 18 and 19, and how feature 19 relates to the labels

4.3 Machine Learning

4.3.1 Data preparation for ML

In order to process any problem through a ML algorithm, additional machine learning -specific data preparation is necessary. Some aspects of this process is not discussed, as there is need to somewhat limit the scope of this thesis.

This process is included in the training of a TCN model for ID task on the UIA IMU data, Appendix D.2. The following examples are taken directly from the python environment, and may somewhat deviate from the code included in the appendix.

4.3.1.1 Normalizing the features

Feature scaling is an important step befre using many machine learning algorithms. The Z-score normalization, where the data is rescaled such that it has a standard deviation of 1 and a mean of 0, can be done by using the `sklearn.preprocessing` library in Python [23].

The selected features for the UIA IMU GI task, was normalized using the method in listing 4.10.

Listing 4.10: Normalizing features for ML task

```
    import pandas as pd

df = pd.read_csv('/kaggle/input/UIA_ID_Walking_Gait_Dataset_8_1_2024')

.....

import numpy as np
from sklearn.preprocessing import StandardScaler

.....

Feats = ['Pitch', 'Yaw', 'q1', 'magX', 'q2', 'magZ']
Label = ['Gait_Identification']
Columns = ['Gait_Identification', 'Pitch', 'Yaw', 'q1', 'magX', 'q2', 'magZ']

df = df[Columns]

# Scale
scaler = StandardScaler()
df[Feats] = scaler.fit_transform(df[Feats]).copy()
```

4.3.2 Feature selection

For feature selection, there are many available tools. The proposed solution is to run a dual feature selection process that involves analysing the features with both a Support Vector Machine (SVM) and a Random Forest (RF) algorithm. The philosophy behind this is to leverage the strength of both classifiers to detect both linear and non-linear relationships between the data and the label values, see chapter 3.3.

The proposed solution is used for the GI task on both the UIA and Topman data in Appendix D.1 and D.2 respectively.

The top fove for both classification algorithms is recorded. An operation is done on the two sets of features to make a third set, that includes all features that appear in either top 5. The results for running this algorithm on UIA and Topman datasets for GI are listed in listing 4.11, and plotted in figures 5.1 and 5.3.

Listing 4.11: Printing the result of the feature selection process for GI task

```
# Top 5 features from Random Forest for Identification task
top_rf_identification = [Feats_ID[i] for i in indices_rf_identification[:5]]

# Top 5 features from SVM for Identification task
top_svm_identification = [Feats_ID[i] for i in indices_svm_identification[:5]]

# Combine top features for Identification task
combined_top_identification = set(top_rf_identification) | ...
    set(top_svm_identification)

# For clearity: The operation above combines the two sets
# set(top_rf_identification) | set(top_svm_identification)
# This command creates a set to include all unique items,
# from both sets.

print("RF Top Features for Identification Task:", top_rf_identification)
print("SVM Top Features for Identification Task:", top_svm_identification)
print("\n")
print("Combined Top Features for Identification Task:", ...
    combined_top_identification)




UIA IMU dataset feature ranking:


RF Top Features for Identification Task:
['Yaw', 'magX', 'Pitch', 'q2', 'q1']
SVM Top Features for Identification Task:
['Pitch', 'magX', 'magZ', 'q1', 'q2']

Combined Top Features for Identification Task UIA IMU dataset:
{'magX', 'Yaw', 'q1', 'Pitch', 'q2', 'magZ'}


......


Topman dataset feature ranking:

RF Top Features for Identification Task:
['q2', 'q1', 'MotionDeg', 'q3', 'Yaw']
SVM Top Features for Identification Task:
['q2', 'q3', 'Yaw', 'q1', 'Pitch']

Combined Top Features for Identification Task Topman dataset:
{'q2', 'MotionDeg', 'q1', 'Pitch', 'Yaw', 'q3'}



......
```

UIA dataset Feature selection

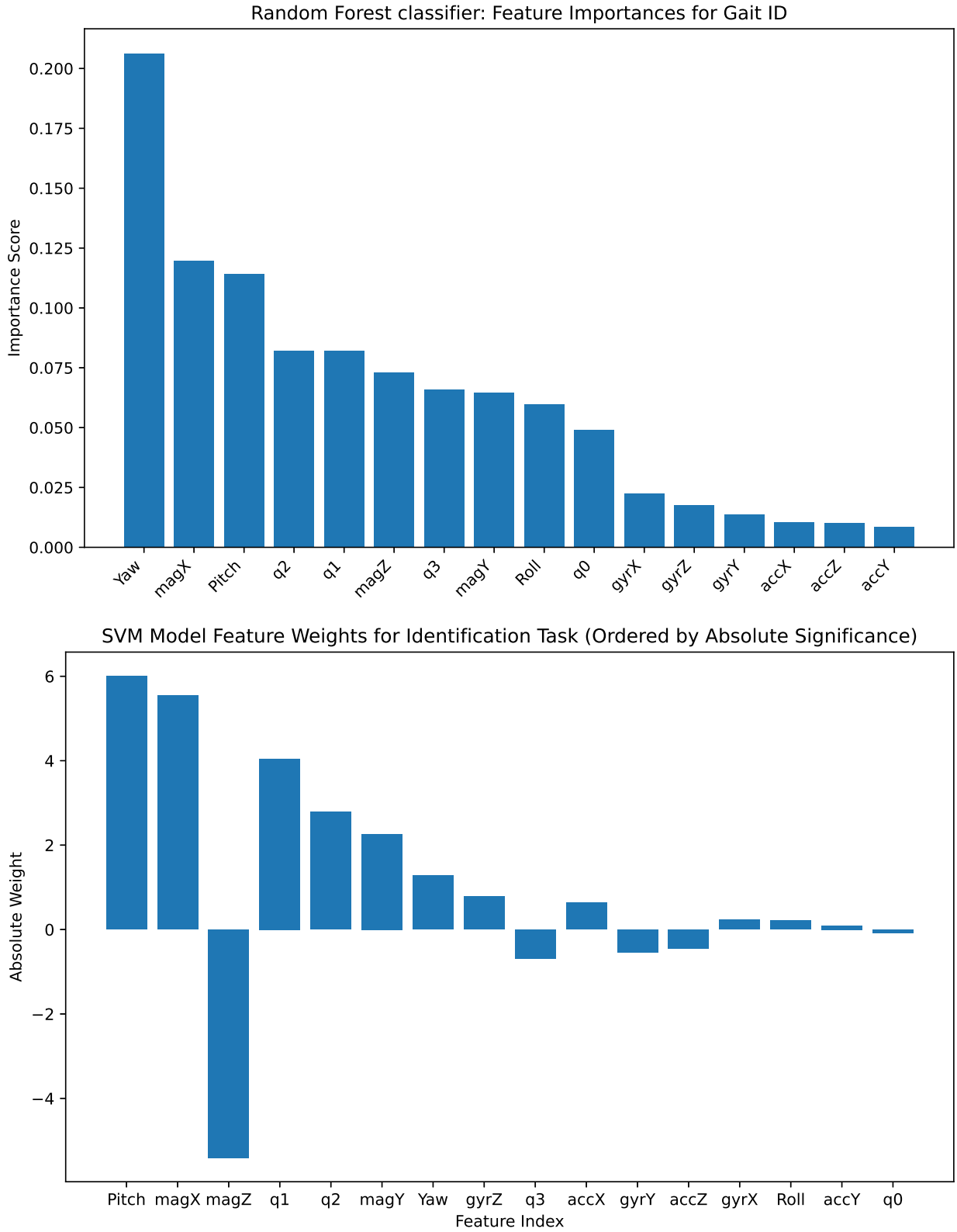


Figure 4.6: UIA dataset Feature selection

Topman dataset Feature selection

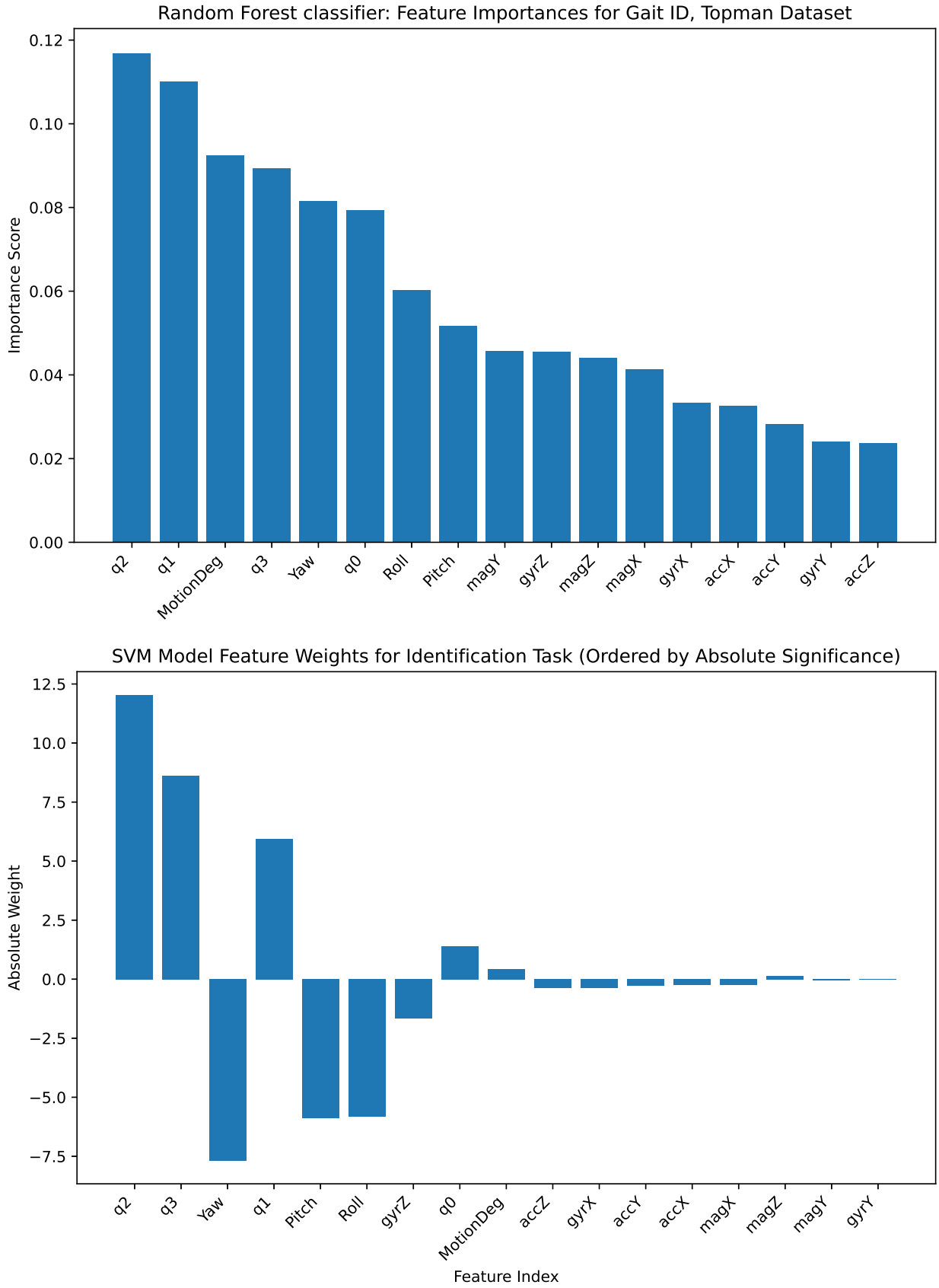


Figure 4.7: Topman dataset Feature selection

4.3.3 Training the ML models

Model training is displayed numerous times in the Appendix, for example D.2 and D.2.

In order to train, save, load, and train models further across multiple sessions, the .h5 format is utilized.

When training, a TensorFlow Keras ModelCheckpoint callback is utilized to continuously monitor, and save, the best performing version of the model. Performance can be measured in various ways, in this case the validation accuracy is chosen as out metric. The setup for the callback is listed here **??**, along with the call for training the model, in order to display how to call the fit (training function) in TensorFlow Keras to utilize this functionality [24].

Listing 4.12: Setting up callback for saving model during training

```
    from keras.callbacks import ModelCheckpoint

best_validation_callback = ModelCheckpoint(
    'TCN_Model.h5',  # Filename to save the model in the SavedModel format
    monitor='val_accuracy',         # Monitor the validation accuracy
    verbose=1,                      # Logging level
    save_best_only=True,            # Save only when the validation ...
       accuracy improves
    mode='max',                     # Mode 'max' because we are monitoring ...
       'val_accuracy'
    save_format='h5'                # Explicitly state to save in ...
       TensorFlow SavedModel format
)


# Fit model to data (training)
history = TCN_Model.fit(X_train, y_train,
                        epochs=300, batch_size=32,
                        validation_data=(X_test, y_test),
                       callbacks=[best_validation_callback])
```

The resulting file can be saved, and loaded in to the Python workspace.

This process is done for all trained models before running the Max F1-Averaged Stratified k-Fold validation. This is demonstrated in 4.13

Listing 4.13: Loading models for validation, Topman GI task

```
    from keras.models import load_model

# Define the custom object
custom_objects = {'ResidualBlock': ResidualBlock}

# Load the model
TCN_Model = load_model('/kaggle/input//TCN_Model_TopMan_ID.h5', ...
    custom_objects=custom_objects)


from keras.models import load_model

# Define the custom object
custom_objects = {'MyAttention': MyAttention}

# Load the model
#TCN_Model = load_model('/kaggle/input/TCN_Model_HH.h5', ...
    custom_objects=custom_objects)
```

```
A_C_LSTM_Model = load_model('/kaggle/input/A_C_LSTM_Topman_ID.h5', ...
    custom_objects=custom_objects)

CNN_Model = load_model('/kaggle/input/Strict_CNN_model_Topman_ID.h5')

#Check if the model weights loaded correctly:

test_loss, test_accuracy = TCN_Model.evaluate(X, y)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')
99/99 [==============================] - 3s 7ms/step - loss: 0.0350 - ...
    accuracy: 0.9914
Test Loss: 0.03497104346752167, Test Accuracy: 0.9914394617080688

test_loss, test_accuracy = A_C_LSTM_Model.evaluate(X, y)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')
99/99 [==============================] - 6s 48ms/step - loss: 0.5971 - ...
    accuracy: 0.8050
Test Loss: 0.5970556139945984, Test Accuracy: 0.8050094842910767

test_loss, test_accuracy = CNN_Model.evaluate(X, y)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')
99/99 [==============================] - 1s 4ms/step - loss: 1.7976 - ...
    accuracy: 0.8272
Test Loss: 1.7975622415542603, Test Accuracy: 0.8272035717964172

# Trained versions of models loaded.
```

4.3.4 ML model validation

For the model validation, a combination of stratified k-fold validation and F1 scores are utilized.

The technique of saving the best performing model is used, just like under training. This way, we collect the best F1 score per fold, and score each model by their average score for all folds.

The entire model evaluation process is shared in Appendix D.2 and D.2. The code to execute and plot the validation result is displayed in listing 4.14

Listing 4.14: The Max F1-Averaged Stratified k-Fold validation

```
    #The combined Average F1 - K-fold cross validation, Model Validation kit by Martin B Gresli. Made in 2024

### PS, only run once, as you will reset the dictonary holding model metrics if run again.

from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import f1_score
from tensorflow.keras.models import Sequential
from keras.optimizers import Adam
import tensorflow as tf
from keras.models import load_model
from keras.callbacks import ModelCheckpoint


import matplotlib.pyplot as plt
import matplotlib.colors as colors
import numpy as np
import tensorflow as tf
import os

def plot_training_history(training_history, title, filename):

    import matplotlib.pyplot as plt
    fig, ax1 = plt.subplots()

    # Plot training and validation loss on the left y-axis
    ax1.plot(training_history.history['loss'], 'b--', alpha=0.3, label='Training Loss')
    ax1.plot(training_history.history['val_loss'], 'b-', label='Validation Loss')
    ax1.set_xlabel('Epochs')
    ax1.set_ylabel('Loss', color='b')
    ax1.tick_params(axis='y', labelcolor='b')

    # Create a second y-axis
    ax2 = ax1.twinx()

    # Plot training and validation accuracy on the right y-axis
    ax2.plot(training_history.history['accuracy'], 'r--', alpha=0.3, label='Training Accuracy')
```

```
    ax2.plot(training_history.history['val_accuracy'], 'r-', label='Validation Accuracy')
    ax2.set_ylabel('Accuracy', color='r')
    ax2.tick_params(axis='y', labelcolor='r')


    lines, labels = ax1.get_legend_handles_labels()
    lines2, labels2 = ax2.get_legend_handles_labels()
    ax2.legend(lines + lines2, labels + labels2, loc='best')

    # Set the title
    plt.title(f'Training History for {title}')

    # Save the plot as a file
    fig.savefig(f'{filename}.eps', format='eps')
    plt.show()
    plt.close()

    import matplotlib.pyplot as plt
    import numpy as np

def save_f1_scores_grouped_as_eps(model_metrics, filename='f1_scores_grouped_performance.eps'):



    model_names = list(model_metrics.keys())
    n_models = len(model_names)
    n_folds = len(model_metrics[model_names[0]]['F1_Scores'])

    # Data preparation
    f1_scores = np.array([model_metrics[model_name]['F1_Scores'] for model_name in model_names]).T  # Transposed for ...
        grouping

    average_f1_scores = [model_metrics[model_name]['Average_F1'] for model_name in model_names]

    fig, ax = plt.subplots(figsize=(10 + n_models, 6))  # Dynamic width based on number of models

    # Calculate the width of each bar and the positions for the groups
    bar_width = 0.6 / n_folds  # The total width for each group is 0.8, leaving some space between groups
    indices = np.arange(n_models)
```

```
    # Plot each fold's F1 scores
    for i in range(n_folds):
        ax.bar(indices - 0.4 + i * bar_width, f1_scores[i], bar_width, label=f'Fold {i+1}')

    # Plot the average F1 score
    ax.bar(indices + 0.4, average_f1_scores, bar_width, label='Average', color='skyblue', edgecolor='black')

    # Add model names to the x-axis
    ax.set_xticks(indices)
    ax.set_xticklabels(model_names, rotation=45)
    ax.set_ylim(0.9, 1)



    ax.set_xlabel('Model Name')
    ax.set_ylabel('F1 Score')
    ax.set_title('F1 Scores per Fold and Average F1 Score of Models')


    ax.legend(loc='lower right')

    # Save the plot as a vector graphic file .eps
    plt.tight_layout()
    plt.savefig(filename, format='eps')
    plt.savefig(filename.replace('.eps', '.jpeg'), format='jpeg')
    plt.show()
    plt.close()



def save_f1_scores_table_as_eps(model_metrics, filename='f1_scores_table.eps'):

    '''

    This is a vector graphic file of the results in the form of a table.
    I you prefer to make an actual table, the data is available in the
    dictionary:
```

```
model_metrics = {}

'''



# Prepare data for the table
columns = ['Model Name'] + [f'Fold {i+1}' for i in range(len(next(iter(model_metrics.values()))['F1_Scores']))] + ...
    ['Average F1']
cell_text = []

for model_name, metrics in model_metrics.items():
    row = [model_name] + metrics['F1_Scores'] + [metrics['Average_F1']]
    cell_text.append(row)

column_width = 3
table_width = column_width * (len(columns))
table_height = 0.1 * len(cell_text)

fig, ax = plt.subplots(figsize=(table_width, table_height))
ax.axis('off')

the_table = ax.table(cellText=cell_text, colLabels=columns, loc='center', cellLoc='center', colLoc='center')

# Make the column names bold
for (i, j), cell in the_table.get_celld().items():
    if i == 0:
        cell.set_fontsize(12)
        cell.set_text_props(weight='bold')

fig.tight_layout()

# Save the plot as a vector graphic file .eps
plt.savefig(filename, format='eps', bbox_inches='tight')
plt.savefig(filename.replace('.eps', '.jpeg'), format='jpeg', bbox_inches='tight')
plt.show()
plt.close()
```

```
def plot_confusion_matrix_heatmap(model, model_name, X, y, title, filename_prop):

    '''

    This is a tweaked CM with a colormap mapped to the rate of correct predictions, but the numbers in
    the cells of the matrix displaying the actual count for predicitons.

    This way you can quickly grasp the distribution for the different classes, as well as
    per-class performance for the model.

    The heatmap is exponential at the very top of the scale, so that any sub-100% performance is
    clearly visible. Likse so:

    [(0, 'lightblue'), (0.5, 'skyblue'), (0.99, 'blue'), (1, 'darkblue')]

    '''



    predict = model.predict(X)
    predictions = np.argmax(predict, axis=1)

    output_dim = len(np.unique(y))

    confusion_matrix = tf.math.confusion_matrix(labels=y,
                                                predictions=predictions,
                                                num_classes=output_dim)

    # Calculate the correct prediction
    total_predictions_per_class = np.sum(confusion_matrix, axis=1)  # Sum over rows for total predictions
    correct_predictions = np.diag(confusion_matrix)
    correct_prediction_rate = correct_predictions / total_predictions_per_class.astype(float)

    # Create a matrix to hold the values for coloring the heatmap
```

```
    heatmap_data = np.full_like(confusion_matrix, np.nan, dtype=float)  # Fill with NaN
    np.fill_diagonal(heatmap_data, correct_prediction_rate)  # Set the diagonal with correct prediction rate

    # Create a colormap that is light blue for 0 and dark blue for high correct prediction rate
    cmap = colors.LinearSegmentedColormap.from_list(
        'custom blue',
        [(0, 'lightblue'), (0.5, 'skyblue'), (0.99, 'blue'), (1, 'darkblue')]
    )
    cmap.set_bad('lightgrey', 1.0)  # Color for NaN values

    # Plotting the heatmap
    fig, ax = plt.subplots()
    cax = ax.matshow(heatmap_data, cmap=cmap, vmin=0, vmax=1)

    # Add colorbar
    plt.colorbar(cax)

    # Annotate all cells with the actual count
    for (i, j), val in np.ndenumerate(confusion_matrix):
        if i == j:  # Diagonal: correct predictions
            text_color = 'white' if correct_prediction_rate[i] > 0.5 else 'black'
        else:  # Off-diagonal: incorrect predictions
            text_color = 'grey'
        ax.text(j, i, f'{val}', ha='center', va='center', color=text_color)

    plt.xlabel('Predicted Labels')
    plt.ylabel('True Labels')
    plt.title(f'{title}')
    plt.savefig(f'{filename_prop}.eps', format='eps')

    plt.show()

    plt.close()



# plot_training_history(training_history, 'training_history')
```

```
# plot_confusion_matrix_heatmap(confusion_matrix, 'confusion_matrix_heatmap')



model_metrics = {}



def validate_and_store_metrics(model_list, X, y, k=5, learning_rate=0.01, epochs=15, batch_size=32):
    # Ensure model_metrics is accessible within this function
    global model_metrics

    for model, model_name, custom_object in model_list:

        # Initialize StratifiedKFold

        skf = StratifiedKFold(n_splits=k, shuffle=True, random_state=55)

        # Initialize a list to store F1 scores for each fold
        f1_scores = []

        j = 0
        for train_index, test_index in skf.split(X, y):
            # Split data
            X_train, X_test = X[train_index], X[test_index]
            y_train, y_test = y[train_index], y[test_index]

            # Compile and fit the model
            model.compile(optimizer=Adam(lr=learning_rate), loss='sparse_categorical_crossentropy', metrics=['accuracy'])



            '''

            Dyneamic save/load model, to ensure the best result from each fold is the version used for F1-scoring
```

```
'''



checkpoint_filepath =  os.path.join('/kaggle/working/', f'{model_name}_Fold_{j}_Checkpoint.h5')

best_validation_callback = ModelCheckpoint(
    checkpoint_filepath,
    monitor='val_accuracy',
    verbose=1,
    save_best_only=True,
    mode='max',
    save_format='h5'
)


history = model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, validation_data=(X_test, ...
   y_test), callbacks=[best_validation_callback])


'''

If problems accessing the model file arises some conditions can be added.

Example:



if os.path.exists(checkpoint_filepath):
    print(f"Checkpoint file created: {checkpoint_filepath}")
    model_loaded = load_model(checkpoint_filepath)
else:
    print(f"Checkpoint file not found at: {checkpoint_filepath}. Saving the last model state manually.")
    model.save(checkpoint_filepath)
    model_loaded = model  # Use the current model state

'''

# Plot training history
```

```
plot_training_history(history, f'{model_name}, fold {j}', f'Training_History_fold{j}_{model_name}')



'''

In this line:

model = load_model(checkpoint_filepath, custom_object)

The custom object is allready loaded in the workspace, as this loop
extrapolates all three nested items in each list entry.



'''

model = load_model(checkpoint_filepath, custom_object)

# Evaluate the model
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f'Test Loss {model_name}, fold {j}: {test_loss}, Test Accuracy {model_name}, fold {j}: {test_accuracy}')
y_pred = np.argmax(model.predict(X_test), axis=1)
y_pred_classes = np.round(y_pred).astype(int)

# Calculate and store F1 score
f1 = f1_score(y_test, y_pred_classes, average='macro')
if model_name not in model_metrics:
    model_metrics[model_name] = {'F1_Scores': [], 'Average_F1': 0}
model_metrics[model_name]['F1_Scores'].append(f1)

print(f'F1 score for {model_name}, fold {j}: {f1}')
f1_scores.append(f1)

# Plot confusion matrix heatmap
plot_confusion_matrix_heatmap(model, model_name, X_test, y_test, f'Confusion matrix for {model_name}, fold ...
    {j}', f'Prop_Confusion_Matrix_fold_{j}_{model_name}')

j += 1
```

```
        average_f1_score = np.mean(model_metrics[model_name]['F1_Scores'])
        model_metrics[model_name]['Average_F1'] = average_f1_score

        print(f'RESULTS for {model_name}')

        print(f"Average F1 Score for {model_name}: {average_f1_score}")
        print(f'F1 scores for all folds for {model_name}:', model_metrics[model_name]['F1_Scores'])
        print('') # Make room


    save_f1_scores_table_as_eps(model_metrics, filename='f1_scores_table.eps')
    save_f1_scores_grouped_as_eps(model_metrics, filename='f1_scores_grouped_performance.eps')



'''

Please use this format:


model_list = [

    (Attentive_LSTM_Model, 'A_C_LSTM_Model', {'MyAttention': MyAttention}),
    (TCN_Model, 'TCN_Model', {'ResidualBlock': ResidualBlock}),
    (CNN_Model, 'CNN_Model', None)

]

The last entry for each line is a dictionary declaring any costum objects used in the models.
If costum objects are used, declaring the entire model before loading the model file might be necessary.
Please enter " None " for all models that stricktly contain objects that are in the libraries loaded in workspace.


validate_and_store_metrics(model_list, X, y, k=5, epochs=50) #Example

'''
```

4.3.5 TCN model simplification, UIA IMU GI task

The Temporal Convolutional network proved to be superior in the UIA IMU GI task. The model was then reduced to accomodate the scope of this thesis, which is smart gait applications for real-time systems. The reduction in complexity was simply done by reducing the number of filters in the residual blocks of the TCN, which repeats 14 times in the proposed model.

Training of the first iteration of the TCN can be seen in Appendix D.2. An overview of the model arcitechture and complexity can be called through `model.summary()` in the Python workspace, listing 4.15.

Through experimentation a minimum of model complexity that would still allow the model to reach 100% validation accuracy, and score a 1.0 on the Max F1-Averaged Stratified k-Fold validation was found. This process was in no means exhaustive, and further simplifications could be possible. The model did however not converge towards 100%validation accuracy for any lower level of complexity during testing than the proposed final model iteration, listing 4.16. The process of testing and comparing the simplified model against the other model architectures can be seen in Appendix D.2.

The result of this process was a reduction in trainable parameters in the order of 2.51. For reference we compare the simplified TCN with the 2nd best performing model, the CNN, which can be referenced in the model evaluation in Appendix D.2. We use the number of trainable parameters, and the memory allocation of said parameters (Param Memory (MB)), and the relative differences in memory allocation as measurands for model complexity. See table 4.1 for results.

Model	Trainable Params	Param Memory (MB)	Relative Size
TCN	347661	1.33	$\frac{1.33}{0.54075} \approx 2.46$
CNN	283437	1.08	$\frac{1.08}{0.54075} \approx 2.00$
TCN Simplified	138433	0.54075	1

Table 4.1: Results of TCN model simplification for the UIA IMU task

Listing 4.15: TCN first iteration, model complexity UIA IMU GI task

```
TCN_Model.summary()

        Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_1 (Conv1D)           (None, 30, 64)            448

 residual_block_12 (Residua  (None, 30, 64)            28864
 lBlock)

 residual_block_13 (Residua  (None, 30, 64)            28864
 lBlock)

 residual_block_14 (Residua  (None, 30, 64)            28864
 lBlock)

 residual_block_15 (Residua  (None, 30, 64)            28864
 lBlock)

 residual_block_16 (Residua  (None, 30, 64)            28864
 lBlock)

 residual_block_17 (Residua  (None, 30, 64)            28864
 lBlock)

 residual_block_18 (Residua  (None, 30, 64)            28864
 lBlock)

 residual_block_19 (Residua  (None, 30, 64)            28864
 lBlock)

 residual_block_20 (Residua  (None, 30, 64)            28864
 lBlock)

 residual_block_21 (Residua  (None, 30, 64)            28864
 lBlock)

 residual_block_22 (Residua  (None, 30, 64)            28864
 lBlock)

 residual_block_23 (Residua  (None, 30, 64)            28864
 lBlock)

 global_average_pooling1d_1  (None, 64)                0
  (GlobalAveragePooling1D)

 dense_1 (Dense)             (None, 13)                845

=================================================================
Total params: 347661 (1.33 MB)
Trainable params: 347661 (1.33 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Listing 4.16: TCN final iteration, simplified model complexity UIA IMU GI task

```
\begin{verbatim}
Model: "sequential_27"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_27 (Conv1D)          (None, 30, 36)            252

 residual_block_210 (Residu  (None, 30, 36)            9180
 alBlock)

 residual_block_211 (Residu  (None, 30, 36)            9180
 alBlock)

 residual_block_212 (Residu  (None, 30, 36)            9180
 alBlock)

 residual_block_213 (Residu  (None, 30, 36)            9180
 alBlock)

 residual_block_214 (Residu  (None, 30, 36)            9180
 alBlock)

 residual_block_215 (Residu  (None, 30, 36)            9180
 alBlock)

 residual_block_216 (Residu  (None, 30, 36)            9180
 alBlock)

 residual_block_217 (Residu  (None, 30, 36)            9180
 alBlock)

 residual_block_218 (Residu  (None, 30, 36)            9180
 alBlock)

 residual_block_219 (Residu  (None, 30, 36)            9180
 alBlock)

 residual_block_220 (Residu  (None, 30, 36)            9180
 alBlock)

 residual_block_221 (Residu  (None, 30, 36)            9180
 alBlock)

 residual_block_222 (Residu  (None, 30, 36)            9180
 alBlock)

 residual_block_223 (Residu  (None, 30, 36)            9180
 alBlock)

 residual_block_224 (Residu  (None, 30, 36)            9180
 alBlock)
 global_average_pooling1d_2  (None, 36)                0
 7 (GlobalAveragePooling1D)
 dense_27 (Dense)            (None, 13)                481
=================================================================
Total params: 138433 (540.75 KB)
Trainable params: 138433 (540.75 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Listing 4.17: CNN model complexity, UIA IMU GI task

```
CNN_Model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 32, 32)            224

 conv1d_1 (Conv1D)           (None, 32, 32)            2080

 max_pooling1d (MaxPooling1  (None, 32, 16)            0
 D)

 batch_normalization (Batch  (None, 32, 16)            64
 Normalization)

 conv1d_2 (Conv1D)           (None, 32, 64)            2112

 max_pooling1d_1 (MaxPoolin  (None, 32, 32)            0
 g1D)

 batch_normalization_1 (Bat  (None, 32, 32)            128
 chNormalization)

 conv1d_3 (Conv1D)           (None, 32, 64)            4160

 max_pooling1d_2 (MaxPoolin  (None, 32, 32)            0
 g1D)

 batch_normalization_2 (Bat  (None, 32, 32)            128
 chNormalization)

 conv1d_4 (Conv1D)           (None, 32, 64)            4160

 max_pooling1d_3 (MaxPoolin  (None, 32, 32)            0
 g1D)

 batch_normalization_3 (Bat  (None, 32, 32)            128
 chNormalization)

 conv1d_5 (Conv1D)           (None, 32, 64)            4160

 max_pooling1d_4 (MaxPoolin  (None, 32, 32)            0
 g1D)

 batch_normalization_4 (Bat  (None, 32, 32)            128
 chNormalization)

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 256)               262400

 dropout (Dropout)           (None, 256)               0

 batch_normalization_5 (Bat  (None, 256)               1024
 chNormalization)

 dense_1 (Dense)             (None, 13)                3341
```

```
=================================================================
Total params: 284237 (1.08 MB)
Trainable params: 283437 (1.08 MB)
Non-trainable params: 800 (3.12 KB)
_________________________________________________________________
```

4.4 Gait Detection Labeling and label analysis

4.4.1 Label analysis for the Vu et. al dataset

An analysis of the Gait Detection labels in the dataset from Vu and colleagues was excecuted in order to determine the complexity of the task and whether the task was fit for a ML model.

The analysis can be found in Appendix B.2.

4.4.2 Feature selection for Gait Detection on the Vu et. al dataset

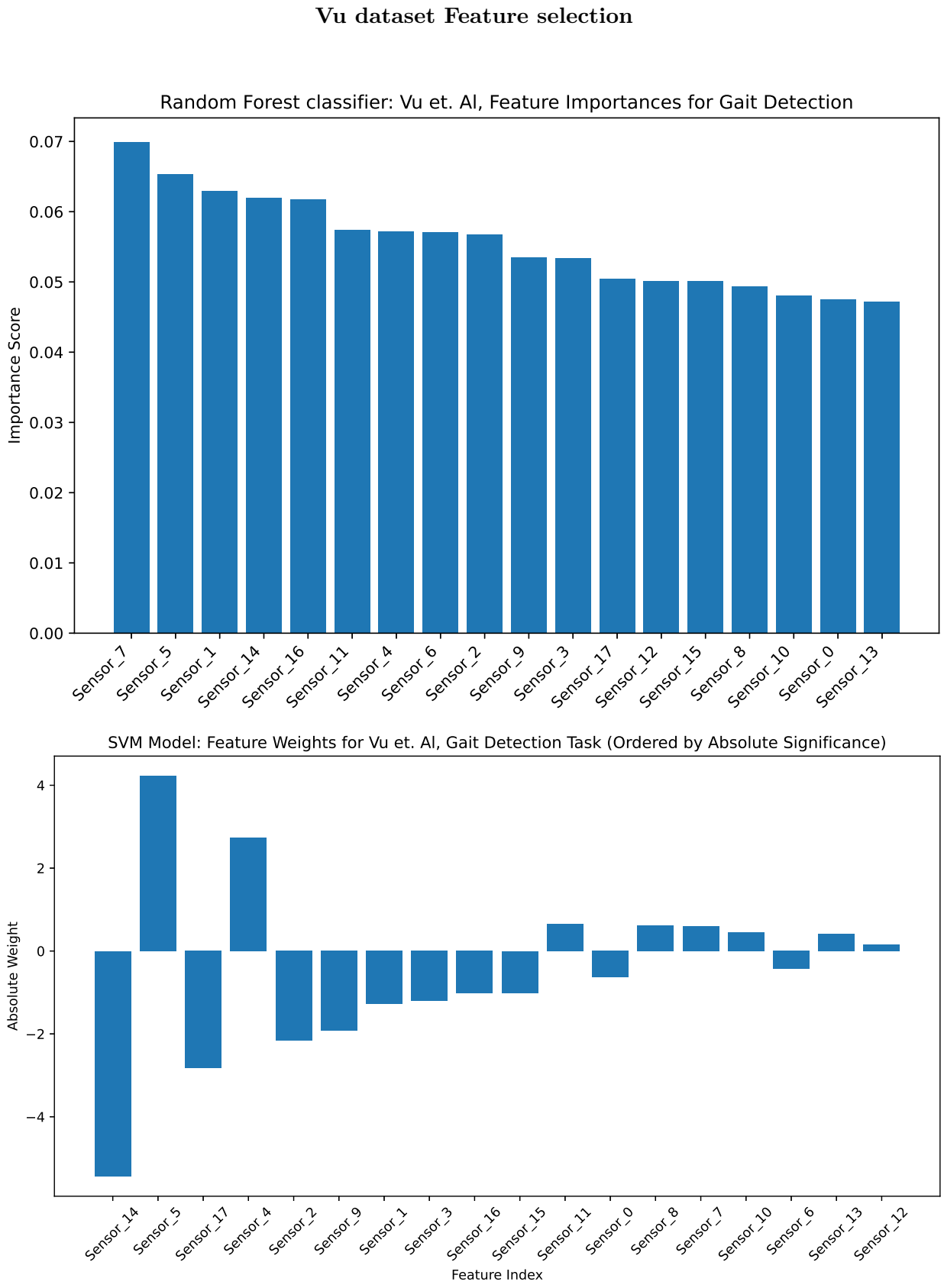


Figure 4.8: Vu dataset Feature selection

4.4.3 Gait Detection labeling algorithm for single-sensor IMU data

Labeling IMU data for is a complex task, and proved to be too time-consuming for the scope of this thesis. Significant work did however go into developing a code for labeling gait events, and so the code is included in listing 4.18. Using the algorithm on the Topman data is included in Appendix C.2

Listing 4.18: Gait Event labeling algorithm for Gait Detection applications

```
    '''

Gait event detection. Detects Heel strikes and Toe-Off's


'''



import matplotlib.pyplot as plt
import numpy as np
from scipy.signal import butter, filtfilt, argrelextrema

def butter_lowpass(data, cutoff_freq, fs, order=4):
    nyquist = 0.5 * fs
    normal_cutoff = cutoff_freq / nyquist
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    y = filtfilt(b, a, data, axis=0)
    return y

def normalize_window(window):
    # Normalize each column (feature) independently
    normalized_window = (window - window.mean(axis=0)) / window.std(axis=0)
    return normalized_window

def detect_local_minima(data, order):
    minima_indices = argrelextrema(data, np.less_equal, order=order)[0]
    return minima_indices

def detect_local_maxima(data, order):
    maxima_indices = argrelextrema(data, np.greater_equal, order=order)[0]
    return maxima_indices

def plot_around_event(event_type, event_pos_in_df, zoom, features, df=df):
    # Determine start and end indices based on the zoom and event position
    start = int(event_pos_in_df - zoom)
    end = int(event_pos_in_df + zoom)
```

```
# Extract the relevant portion of the DataFrame
df_plot = df.loc[start:end, :]

# Extract the subject, for use in plot name
# column name is in 'Gait_Identification'

name = df.loc[start, 'Gait_Identification']


# Create a figure and primary axis
fig, ax1 = plt.subplots(figsize=(10, 6))

# Plot each feature on the primary axis with twinx for secondary and tertiary axes
for i, feature in enumerate(features):
    ax = ax1.twinx() if i > 0 else ax1
    color = ['blue', 'orange', 'green'][i]
    ax.plot(df_plot[feature], label=feature, color=color)
    ax.set_ylabel(feature, color=color)
    ax.tick_params(axis='y', labelcolor=color)

    if i == 2:  # Make some space for y_ax for the third feature (accX)
        ax.spines['right'].set_position(('outward', 60))

# Add a vertical line at the event position
ax1.axvline(x=event_pos_in_df, color='red', linestyle='--', label=f'{event_type}')

# Add a title and legend based on the event type
if event_type == 'TO':
    plt.title(f'Original DataFrame around Toe Off at {start}:{end} in df, subject {name}')
elif event_type == 'HS':
    plt.title(f'Original DataFrame around Heel-strike at {start}:{end} in df, subject {name}')

fig.legend(loc='upper right', bbox_to_anchor=(0.85, 0.85))

plt.savefig(f'{event_type}_pos_{start}_{end}_subject_{name}_in_dataframe.eps', format='eps') #f strings are nice


plt.show()
```

```
def detect_heel_strike(Debug, Index_Loc_Max_Pitch, windowed_maxima_indices_dict, non_W_maxima_indices_dict,
                       windowed_minima_indices_dict, non_W_minima_indices_dict,
                       start_index_subject, start_index_window,
                       HS_lower, HS_upper_Max, HS_upper_Min, HS_Impact_Search_Lower, HS_Impact_Search_Upper,
                       W_zoom_lower, W_zoom_upper,
                       df_Labeling_Copy, df):
    if Debug == True:

        print('Running HS detection')

    Event_Detected = False



    Index_Loc_Min_accX = [index for index in windowed_minima_indices_dict['accX'] if
                          Index_Loc_Max_Pitch - HS_lower <= index <= Index_Loc_Max_Pitch + HS_upper_Min]


    if not any(Index_Loc_Min_accX):
        # If no candidates in windowed_maxima_indices_dict['accX'], check non_W_maxima_indices_dict['accX']
        Index_Loc_Min_accX_nonW = [index for index in non_W_minima_indices_dict['accX'] if
                                   Index_Loc_Max_Pitch - HS_lower <= index <= Index_Loc_Max_Pitch + HS_upper_Min]

        if any(Index_Loc_Min_accX_nonW):
            Window_Index_Loc_Min_accX = Index_Loc_Min_accX_nonW[0]

            if Debug == True:

                print('accX MIN component of HS found inside nonW:', Window_Index_Loc_Min_accX)
        else:
            if Debug == True:
                print('Not able to find local min accX near Pitch Max')
            return None

    else:
        Window_Index_Loc_Min_accX = Index_Loc_Min_accX[0]
```

```
        if Debug == True:
            print('accX MIN component of HS found inside WIN:', Window_Index_Loc_Min_accX)


    if Window_Index_Loc_Min_accX is not None:

        # Check for local maxima of accX within a range around Pitch maxima
        Index_Loc_Max_accX = [index for index in windowed_maxima_indices_dict['accX'] if
                              Window_Index_Loc_Min_accX <= index <= Index_Loc_Max_Pitch + HS_upper_Max]

        if not any(Index_Loc_Max_accX):
            # If no candidates in windowed_maxima_indices_dict['accX'], check non_W_maxima_indices_dict['accX']
            Index_Loc_Max_accX_nonW = [index for index in non_W_maxima_indices_dict['accX'] if
                                       Index_Loc_Max_Pitch <= index <= Index_Loc_Max_Pitch + HS_upper_Max]

            if any(Index_Loc_Max_accX_nonW):
                Window_Index_Loc_Max_accX = Index_Loc_Max_accX_nonW[0]

                if Debug == True:

                    print('accX MAX component of HS found inside nonW:', Window_Index_Loc_Max_accX)
            else:
                if Debug == True:
                    print('Not able to find local max accX near accX min')
                return None

        else:
            Window_Index_Loc_Max_accX = Index_Loc_Max_accX[0]

            if Debug == True:
                print('accX MAX component of HS found inside WIN:', Window_Index_Loc_Max_accX)


    if Window_Index_Loc_Max_accX is not None and Window_Index_Loc_Min_accX is not None:


        if Debug == True:
            print('Found both min and max for accX component near max for pitch. Probable HS. Search for accZ min.')
```

```
Cand_Loc_Min_accZ = [index for index in windowed_minima_indices_dict['accZ'] if
                 Window_Index_Loc_Max_accX - HS_Impact_Search_Lower <= index <= Window_Index_Loc_Max_accX + ...
                    HS_Impact_Search_Upper]

if Debug == True:

    print('these are accZ within range of accX max:', Cand_Loc_Min_accZ)

if any(Cand_Loc_Min_accZ):

    if Debug == True:

        print('Found candidates for accZ: ', Cand_Loc_Min_accZ)

    Cand_Loc_Min_accZ = Cand_Loc_Min_accZ[0]

    if Debug == True:

        print('Heel Strike at w index: ', Cand_Loc_Min_accZ)
        print('Heel Strike at Index: ', Cand_Loc_Min_accZ + start_index_subject + start_index_window)

    Heel_strike = Cand_Loc_Min_accZ + start_index_subject + start_index_window

    if not any(df_Labeling_Copy.loc[max(Heel_strike - 20, 0):min(Heel_strike + 20, len(df_Labeling_Copy) - 1), ...
       'Gait_Events'] == 'HS'):

        #Condition stops search from going out of bounds near start/end of labeling_df


            Zoom = 50

            plot_around_event('HS', Heel_strike, zoom=Zoom, features=['Pitch', 'accZ', 'accX'], df=df)



            df_Labeling_Copy.loc[Heel_strike, 'Gait_Events'] = 'HS'

            Event_Detected = True
```

```
            else:
                if Debug == True:

                    print("Heel Strike already exists within  20  positions; skip tag.")

        else:
            if Debug == True:

                print('Not able to find local min accZ near Pitch Max')


        if Event_Detected:

            return True

    return False


def detect_toe_off(Debug, Index_Loc_Min_Pitch, W_zoom_lower, W_zoom_upper, TO_lower, TO_upper, start_index_subject, ...
    start_index_window,
                   windowed_maxima_indices_dict, non_W_minima_indices_dict, df_Labeling_Copy, Event_Tagged,
                   df, Zoom=50):

    if Debug == True:
        print('Running Toe_Off detection')

    Event_Tagged = False

    if any(Index_Loc_Min_Pitch - TO_lower < index < Index_Loc_Min_Pitch + TO_upper for index in
           windowed_maxima_indices_dict['accZ']):

        matching_indices = np.where(
            (Index_Loc_Min_Pitch - TO_lower < windowed_maxima_indices_dict['accZ']) &
            (windowed_maxima_indices_dict['accZ'] < Index_Loc_Min_Pitch + TO_upper))[0]

        Index_Loc_Max_W_accZ = windowed_maxima_indices_dict['accZ'][matching_indices[0]]

        if Debug == True:
```

```
            print('all instances', windowed_maxima_indices_dict['accZ'])
            print('index in list that meets criteria: ', matching_indices)
            print('Toe off at: ',
                  Index_Loc_Max_W_accZ,
                  'convert to global coordinates:',
                  Index_Loc_Max_W_accZ + start_index_subject + start_index_window)

        Toe_Off = Index_Loc_Max_W_accZ + start_index_subject + start_index_window

        Event_Tagged = True

        if Debug == True:

            print('toe off at: ', Toe_Off)

        if not any(df_Labeling_Copy.loc[max(Toe_Off - 20, 0):min(Toe_Off + 20, len(df_Labeling_Copy) - 1), ...
            'Gait_Events'] == 'TO'):
            #Condition stops search from going out of bounds near start/end of labeling_df

            df_Labeling_Copy.loc[Toe_Off, 'Gait_Events'] = 'TO'

            plot_around_event('TO', Toe_Off, zoom=Zoom, features=['Pitch', 'accZ', 'accX'], df=df)

        else:
            if Debug == True:

                print("Toe_off already exists within  20  positions; skip tag.")

    else:
        if Debug == True:

            print(f'Toe_Off_Detection: Could not find accZ MAX in range [ {Index_Loc_Min_Pitch - TO_lower} , ...
                {Index_Loc_Min_Pitch + TO_upper} ]')

    if Event_Tagged:

        return True
```

```
    return False


def windowed_subject_data_plotter(df, df_Labeling_Copy, subject_column, subject_number, Debug=True, features=None, ...
   window_size=100,
                                  overlap=50, cutoff_freq=None, fs=None, order=None,
                                  non_windowed_features=None, NWF_minmax_orders=None,
                                  minmax_windowed_features=None):

    # Filter the DataFrame based on the specified subject and subject column
    subject_data = df[df[subject_column] == subject_number]

    start_index_subject = subject_data.index[0]

    # Extract the selected features
    selected_data = subject_data[features]

    # Apply Hamming windowing and Butterworth lowpass filtering to the data
    hamming_window = np.hamming(window_size)[:, None]  # Reshape to column vector
    windows = [
        butter_lowpass(selected_data[i:i+window_size].values * hamming_window, cutoff_freq, fs=fs, order=order)
        for i in range(0, len(selected_data)-window_size+1, overlap)
    ]

    # Normalize each window
    normalized_windows = [normalize_window(window) for window in windows]

    windowed_minima_indices_dict = {} #re-initialized for every window
    windowed_maxima_indices_dict = {} #re-initialized for every window

    Continue_Pitch_TH = False

    # Loop over non_windowed_features and NWF_minmax_orders simultaneously
    for j, (window, unfiltered_window) in enumerate(zip(normalized_windows, windows)):



        '''
        Event_Tagged is used to skip to next window as soon as either HS or TO is tagged
```

```
This mediates the problem of processing events to near the edge of the window,
where the characteristics of the measurands are dirstorted.

'''

Event_Tagged = False



start_index_window = j * (window_size - overlap)


fig, axs = plt.subplots(len(features) + len(non_windowed_features), 1, figsize=(10, 2*(len(features) + ...
    len(non_windowed_features))))



# Plot the normalized and filtered windowed data in separate plots for each feature and window
for i, feature in enumerate(features):
    axs[i].plot(range(len(window)), window[:, i], label=f'{feature} (Filtered and Normalized)')

    min_order, max_order = minmax_windowed_features[i] # fetch orders from function call list ...
        minmax_windowed_features=[[8,8],[35,9]]

    minima_indices = detect_local_minima(window[:, i], min_order)
    minima_indices_original = minima_indices + start_index_window

    axs[i].vlines(minima_indices, ymin=window[:, i].min(), ymax=window[:, i].max(), color='red', ...
        linestyle='dashed', label='accZ Minima')

    maxima_indices = detect_local_maxima(window[:, i], max_order)
    axs[i].vlines(maxima_indices, ymin=window[:, i].min(), ymax=window[:, i].max(), color='blue', ...
        linestyle='dashed', label='accZ Maxima')

    windowed_minima_indices_dict[feature] = minima_indices
    windowed_maxima_indices_dict[feature] = maxima_indices

    axs[i].set_xlabel('Index')
```

```
    axs[i].set_ylabel(f'{feature} Values')
    axs[i].set_title(f'Window {j+1}')
    axs[i].legend()

# Add a separate subplot for the features in non_windowed_features within the current window


print_statements = [] #re-initialized for every window

non_W_minima_indices_dict = {} #re-initialized for every window
non_W_maxima_indices_dict = {} #re-initialized for every window

for i, (non_windowed_feature, minmax_orders) in enumerate(zip(non_windowed_features, NWF_minmax_orders)):

    index_for_orders = i % len(NWF_minmax_orders)  # Ensure that the index wraps around for the length of ...
       NWF_minmax_orders
    globals()[f'{non_windowed_feature}_Minima_order_global'] = minmax_orders[0]
    globals()[f'{non_windowed_feature}_Maxima_order_global'] = minmax_orders[1]


    # Plot the non-windowed features
    feature_data = subject_data[non_windowed_feature].values[start_index_window:start_index_window + window_size]
    filtered_feature_data = butter_lowpass(feature_data, cutoff_freq=25, fs=100, order=2)
    axs[len(features) + i].plot(range(start_index_window,
                                      start_index_window + len(feature_data)),
                                filtered_feature_data, label=f'{non_windowed_feature} (Filtered)')

    axs[len(features) + i].set_xlabel('Index')
    axs[len(features) + i].set_ylabel(f'{non_windowed_feature} Values')
    axs[len(features) + i].set_title(f'Window {j + 1} - Filtered {non_windowed_feature}')
    axs[len(features) + i].legend()
    xticks = np.arange(start_index_window,
                       start_index_window + len(feature_data), 20)  # Adjust the step size as needed
    axs[len(features) + i].set_xticks(xticks)
    axs[len(features) + i].set_xticklabels(xticks - start_index_window)

    #axs[len(features) + i].set_xlim([0, len(feature_data)])
```

```
'''

# Check if the function call feeds the right arguments to argrelextrema


print(non_windowed_feature)
print(f'NWF_minmax_orders: {NWF_minmax_orders}')
print(f'minmax_orders: {minmax_orders}')


'''




# Detect local minima and maxima for the current non_windowed_feature
minima_indices = detect_local_minima(filtered_feature_data, minmax_orders[0])



minima_indices_original = minima_indices + start_index_window  # Use start_index_window
axs[len(features) + i].vlines(minima_indices_original,
                              ymin=filtered_feature_data.min(), ymax=filtered_feature_data.max(),
                              color='red', linestyle='dashed', label=f'{non_windowed_feature} Minima')

maxima_indices = detect_local_maxima(filtered_feature_data, minmax_orders[1])



maxima_indices_original = maxima_indices + start_index_window  # Use start_index_window
axs[len(features) + i].vlines(maxima_indices_original,
                              ymin=filtered_feature_data.min(), ymax=filtered_feature_data.max(),
                              color='blue', linestyle='dashed', label=f'{non_windowed_feature} Maxima')

axs[len(features) + i].legend()

'''
```

```
minima_indices_dict[non_windowed_feature] = minima_indices
maxima_indices_dict[non_windowed_feature] = maxima_indices
'''

non_W_minima_indices_dict[non_windowed_feature] = minima_indices
non_W_maxima_indices_dict[non_windowed_feature] = maxima_indices



'''

# append print statements to display where local minimas are (non-windowed feats)

txt = f'Window {j + 1} -
Minima indicies for Filtered {non_windowed_feature}: {minima_indices}'

print_statements.append(txt)

txt = f'Window {j + 1} -
Minima indicies for Filtered {non_windowed_feature} as they appear in original df index: ...
    {minima_indices+start_index_subject+start_index_window}'

print_statements.append(txt)

# append print statements to display where local maximas are (non-windowed feats)

txt = f'Window {j + 1} - Maxima indicies for Filtered {non_windowed_feature}: {maxima_indices}'

print_statements.append(txt)

txt = f'Window {j + 1} - Maxima indicies for Filtered {non_windowed_feature} as they appear in original df ...
    index: {maxima_indices+start_index_subject+start_index_window}'

print_statements.append(txt)

'''
```

```
    '''
    Setting threshold for pitch magnitude to rule out inactivity form the labeling process
    '''


print(f'Maxima indices for Pitch window {j+1}:', non_W_maxima_indices_dict['Pitch'])
print(f'Minima indices for Pitchwindow {j+1}:', non_W_minima_indices_dict['Pitch'])

Max_pitch_value = subject_data["Pitch"].iloc[start_index_window + non_W_maxima_indices_dict["Pitch"]].max()
Min_pitch_value = subject_data["Pitch"].iloc[start_index_window + non_W_minima_indices_dict["Pitch"]].min()

print(f'Max Pitch: {Max_pitch_value}')
print(f'Min Pitch: {Min_pitch_value}')

'''
print(f'Maxima values for Pitch window {j+1}:', pitch_values_at_maxima)
print(f'Minima values for Pitch window {j+1}:', pitch_values_at_minima)
'''

pitch_magnitude = np.abs(Max_pitch_value - Min_pitch_value)

print(f'Pitch Mag window {j+1} :',pitch_magnitude)

'''
pitch_magnitude_threshold = 20

if pitch_magnitude < pitch_magnitude_threshold:
    print(f'Skipping window {j+1} due to low pitch magnitude: {pitch_magnitude}')

    Continue_Pitch_TH = True

    break

if Continue_Pitch_TH == True:

    continue

    '''
```

```
plt.tight_layout()

#axs[i].set_title(f'Window {j+1}')

plt.savefig(f'TopMan_GaitEvent_Doc_Window_{j+1}.eps', format='eps')

plt.show()



#print('minimas for WIN Pitch:',windowed_minima_indices_dict['Pitch'])
#print('maximas for WIN Pitch:',windowed_maxima_indices_dict['Pitch'])


if Debug == True:


    print('HEEL_STRIKE parameters:')

    print('maximas for Pitch:',non_W_maxima_indices_dict['Pitch'])

    print('maximas for WIN accX:',windowed_maxima_indices_dict['accX'])
    print('maximas for accX:',non_W_maxima_indices_dict['accX'])

    print('minimas for WIN accX:',windowed_minima_indices_dict['accX'])
    print('minimas forr accX:',windowed_minima_indices_dict['accX'])

    print('minimas for WIN accZ:',windowed_minima_indices_dict['accZ'])

    print('maximas for WIN accZ:',windowed_maxima_indices_dict['accZ'])
```

```
        '''
        print('minimas for Pitch:',minima_indices_dict['Pitch']+start_index_subject+start_index_window)
        print('maximas for Pitch:',maxima_indices_dict['Pitch']+start_index_subject+start_index_window)
        '''

        '''
        non_W_minima_indices_dict[non_windowed_feature] = minima_indices
        non_W_maxima_indices_dict[non_windowed_feature] = maxima_indices
        '''

        print('minimas for Pitch:',non_W_minima_indices_dict['Pitch'])

        print('maximas for WIN accZ:',windowed_maxima_indices_dict['accZ'])

        print('maximas for accZ:',non_W_maxima_indices_dict['accZ'])


        print('minimas for accZ:',non_W_minima_indices_dict['accZ'])



        print(f'Window {j+1} - Start Index subject: {start_index_subject}')
        print(f'Window {j+1} - Start Index window: {start_index_window}')


W_zoom_lower = 10
W_zoom_upper = 70

TO_lower = 5
TO_upper = 20

HS_accX_Max = 30
HS_accX_Min = 20

HS_lower = 4
HS_upper_Max = HS_accX_Max
HS_upper_Min = HS_accX_Min
```

```
HS_Impact_accZ_Min = 20



HS_Impact_Search_Lower = 0
HS_Impact_Search_Upper = HS_Impact_accZ_Min

Index_Loc_Max_Pitch_Bool = False

Index_Loc_Min_Pitch_Bool = False


 # HEEL STRIKE incidents
Pitch_Maxs = [index for index in non_W_maxima_indices_dict['Pitch'] if W_zoom_lower < index < W_zoom_upper]

if any(Pitch_Maxs):

    if Debug == True:
        print(f'Condition met: There are indices greater than {W_zoom_lower} and less than {W_zoom_upper} for ...
           loc MAX pitch. Possible Heel-strike!')

    Index_Loc_Max_Pitch = Pitch_Maxs[0]  # if multiple instances, handle the first one in the window first
    Index_Loc_Max_Pitch_Bool = True

    if Debug == True:

        print('index loc max pitch:', Index_Loc_Max_Pitch)

else:

    if Debug == True:

        print(f'No indices found for loc MAX pitch in the specified range {W_zoom_lower}-{W_zoom_upper}.')

 # Toe Off incidents
Pitch_Mins = [index for index in non_W_minima_indices_dict['Pitch'] if W_zoom_lower < index < W_zoom_upper]

if any(Pitch_Mins):
```

```
        if Debug == True:

            print(f'Condition met: There are indices greater than {W_zoom_lower} and less than {W_zoom_upper} for ...
               loc MIN pitch. Possible toe-off')

        Index_Loc_Min_Pitch = Pitch_Mins[0]
        Index_Loc_Min_Pitch_Bool = True

        if Debug == True:
            print('index loc min pitch:', Index_Loc_Min_Pitch)

    else:
        if Debug == True:
            print(f'No indices found for loc MIN pitch in the specified range {W_zoom_lower}-{W_zoom_upper}.')



    if Index_Loc_Min_Pitch_Bool == True or Index_Loc_Max_Pitch_Bool == True:


        if Index_Loc_Min_Pitch_Bool == False and Index_Loc_Max_Pitch_Bool == True:


            HS_Detected = detect_heel_strike(Debug, Index_Loc_Max_Pitch, windowed_maxima_indices_dict, ...
               non_W_maxima_indices_dict,
                   windowed_minima_indices_dict, non_W_minima_indices_dict,
                   start_index_subject, start_index_window,
                   HS_lower, HS_upper_Max, HS_upper_Min, HS_Impact_Search_Lower, HS_Impact_Search_Upper,
                   W_zoom_lower, W_zoom_upper,
                   df_Labeling_Copy, df)







        if Index_Loc_Min_Pitch_Bool == True and Index_Loc_Max_Pitch_Bool == False:
```

```
        TO_Detected = detect_toe_off(Debug, Index_Loc_Min_Pitch, W_zoom_lower, W_zoom_upper, TO_lower, ...
            TO_upper, start_index_subject, start_index_window,
         windowed_maxima_indices_dict, non_W_minima_indices_dict, df_Labeling_Copy, Event_Tagged, df)



    if (Index_Loc_Min_Pitch_Bool == True and Index_Loc_Max_Pitch_Bool == True) and Index_Loc_Min_Pitch < ...
        Index_Loc_Max_Pitch:

        if Debug == True:

            print ('Possibility of both TO and HS, in that order. Running TO first:')


        TO_Detected = detect_toe_off(Debug, Index_Loc_Min_Pitch, W_zoom_lower, W_zoom_upper, TO_lower, ...
            TO_upper, start_index_subject, start_index_window,
              windowed_maxima_indices_dict, non_W_minima_indices_dict, df_Labeling_Copy, Event_Tagged, df)

        if Debug == True:
            print ('TO finished, now we want HS to run: ')

        HS_Detected = detect_heel_strike(Debug, Index_Loc_Max_Pitch, windowed_maxima_indices_dict, ...
            non_W_maxima_indices_dict,
                windowed_minima_indices_dict, non_W_minima_indices_dict,
                start_index_subject, start_index_window,
                HS_lower, HS_upper_Max, HS_upper_Min, HS_Impact_Search_Lower, HS_Impact_Search_Upper,
                W_zoom_lower, W_zoom_upper,
                df_Labeling_Copy, df)


    if (Index_Loc_Min_Pitch_Bool == True and Index_Loc_Max_Pitch_Bool == True) and Index_Loc_Min_Pitch > ...
        Index_Loc_Max_Pitch:

        HS_Detected = detect_heel_strike(Debug, Index_Loc_Max_Pitch, windowed_maxima_indices_dict, ...
            non_W_maxima_indices_dict,
                windowed_minima_indices_dict, non_W_minima_indices_dict,
                start_index_subject, start_index_window,
                HS_lower, HS_upper_Max, HS_upper_Min, HS_Impact_Search_Lower, HS_Impact_Search_Upper,
                W_zoom_lower, W_zoom_upper,
```

```
                        df_Labeling_Copy, df)

                TO_Detected = detect_toe_off(Debug, Index_Loc_Min_Pitch, W_zoom_lower, W_zoom_upper, TO_lower, ...
                    TO_upper, start_index_subject, start_index_window,
                      windowed_maxima_indices_dict, non_W_minima_indices_dict, df_Labeling_Copy, Event_Tagged, df)

    return df_Labeling_Copy



        #for subject in range (0 , Subjects) :

        # Go through subjects [0,29], adjust parameters to detect as many TO and HS as possible.


W_zoom_lower = 10
W_zoom_upper = 70

TO_lower = 5
TO_upper = 10

HS_accX_Max = 30
HS_accX_Min = 20

HS_lower = 4
HS_upper_Max = HS_accX_Max
HS_upper_Min = HS_accX_Min

HS_Impact_accZ_Min = 20



HS_Impact_Search_Lower = 0
HS_Impact_Search_Upper = HS_Impact_accZ_Min

Index_Loc_Max_Pitch_Bool = False

Index_Loc_Min_Pitch_Bool = False
```

```
'''
Subjects = 30

for subject in range (0 , Subjects) :

    print(f'Subject: {subject}')
'''

windowed_subject_data_plotter(df, df_Labeling_Copy, 'Gait_Identification', 1, Debug=True, features=['accZ', 'accX'],
                              minmax_windowed_features=[[12,15],[5,5]],
                              cutoff_freq=15, fs=100, order=3,
                              non_windowed_features=['Pitch', 'accZ', 'accX'],
                              NWF_minmax_orders=[[35, 30], [int(HS_Impact_accZ_Min*0.8), 8], [9, 10]])
```

Chapter 5

Results

5.1 Results for the UIA IMU dataset GI task

13 subjects was recorded by means of single sensor IMU in a simple walking experiment at the UIA campus. The data was processed as described in section 4.2.1, which included filtering and running a Madwick filter on the acellerometer- and gyroscope data in order to generate Quaternion and Euler angle expressions in 3 dimensions.

These six features, along with the 9 axis accelerometer, gyroscope and magnetometer data, was ranked in order of importance by running a Random Forrest classifier and a Support Vector Machine linear classification model. These classifiers were chosen on the basis of the RF algorithm being known for it's ability to determine non-linear relationships between data and label values, and the SVM for it's ability to detect linear relationships in the data.

The results from the combined feature selection and dimensionality reduction process are presented in Figure 5.1.

The analysis clearly shows that any feature related to sensor attitude: Magnetometer readings, quaternion expressions and Euler angles have the tightest relative relationships with the label values.

The results of the combined SVM and RF analysis clearly shows that features associated with sensor attitude, including magnetometer readings, quaternion expressions, and Euler angles, exhibit the most significant relationships with the label values. It might be interesting to note that the magnetometer readings was not as prominent in the same analysis for the Topman data, which might be explained by the high quality of, and the low level of noise exhibited in, the UIA IMU data.

For the machine learning done on the UIA IMU GI task, two models exhibited excellent ability to classify sequences of data to identify the subject based on the IMU walking gait data. The Temporal Convolutional Network described in **??** quickly converged to 100% validation accuracy during training, and scored a perfect 1.0 on the Max F1-Averaged Stratified k-Fold model validation. Due to the good results and ease of training the model, efforts where done to reduce the complexity of the model to reduce computational demand. This is in line with the imagined real-time application. This resulted in a successful reduction in TCN model complexity, as can be seen in table 4.1.

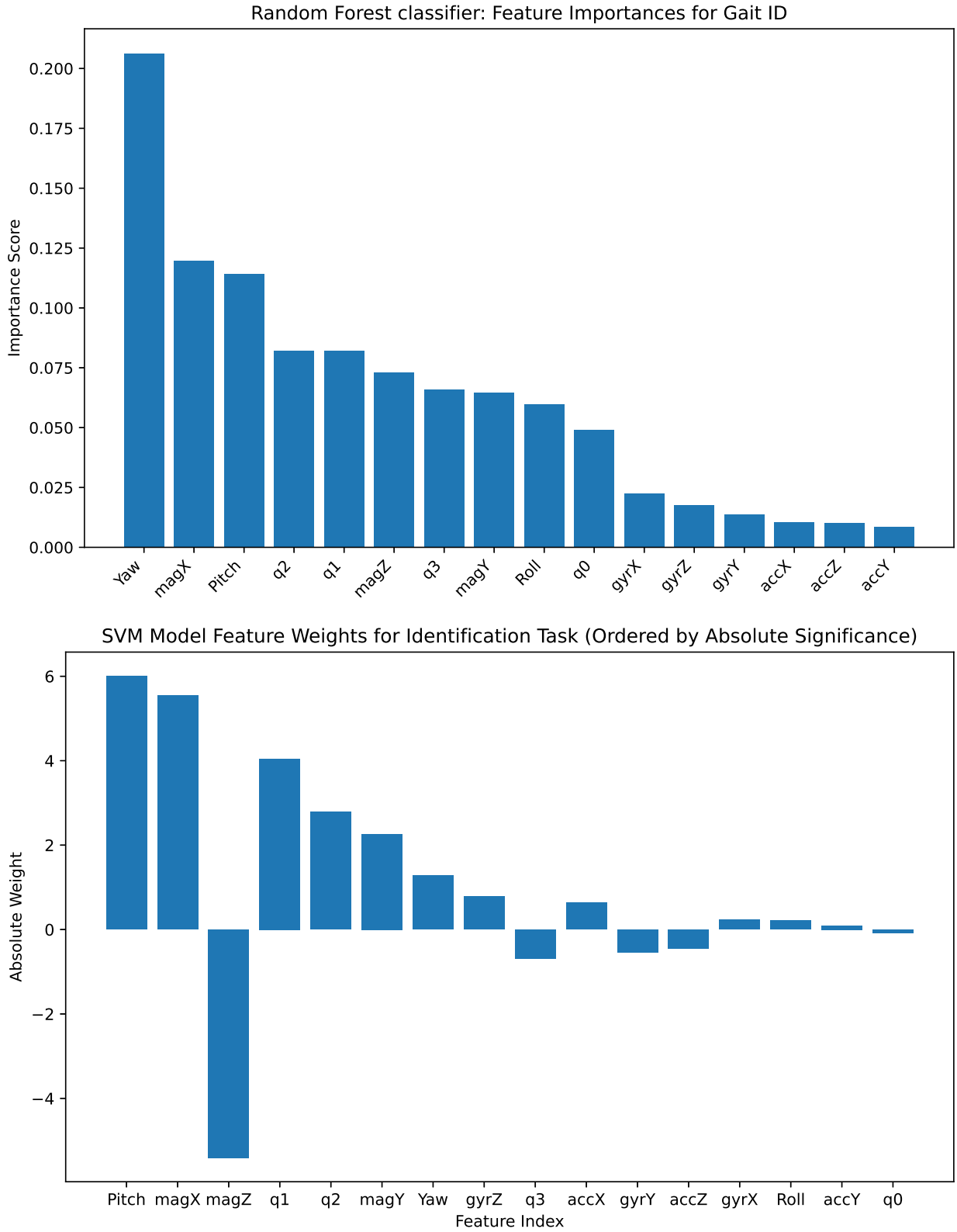
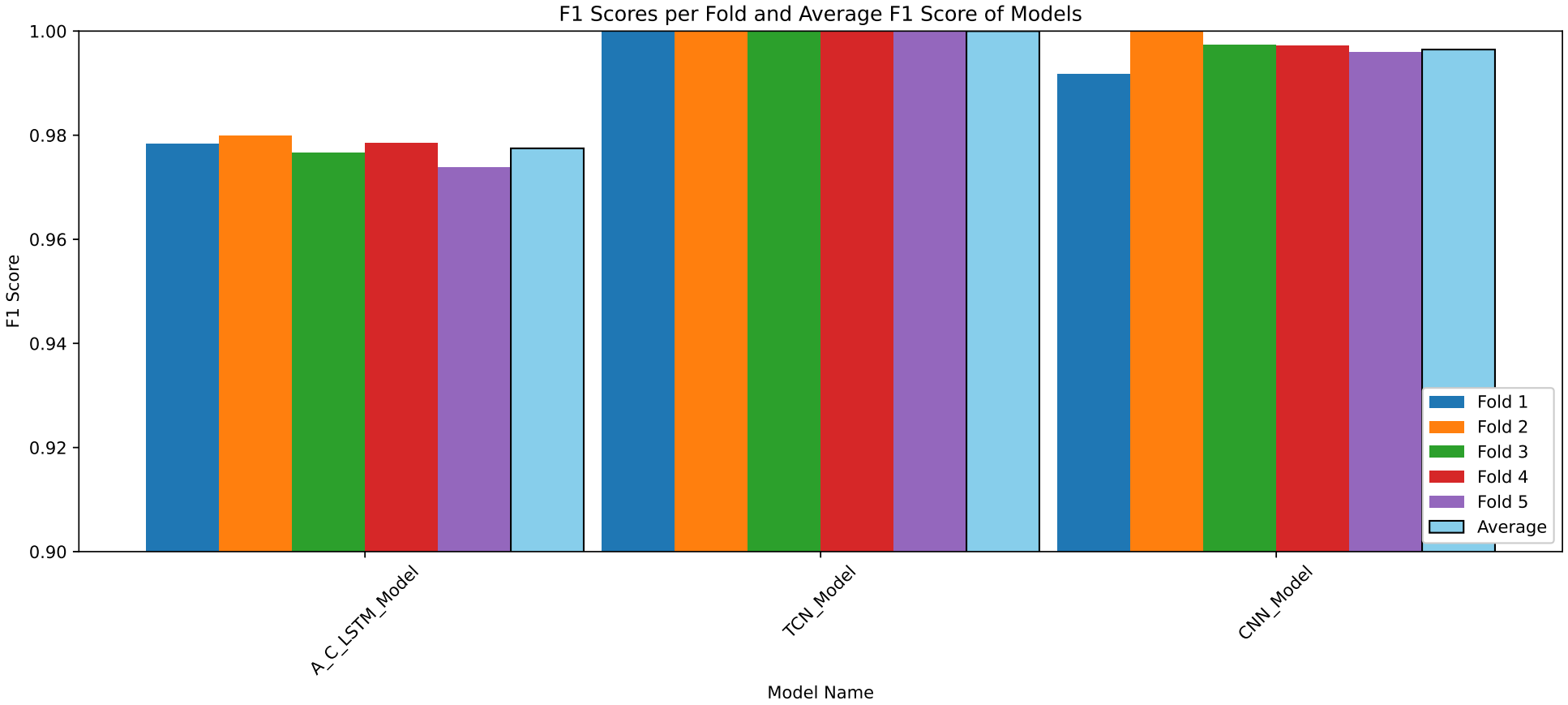


Figure 5.1: UIA dataset Feature selection

UIA IMU dataset GI Task: Max F1-Averaged Stratified k-Fold model validation



Model Name	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average F1
A_C_LSTM_Model	0.9783559218526776	0.9799765088683448	0.9766877451617154	0.9785225793293548	0.9738313786126882	0.9774748267649562
TCN_Model	1.0	1.0	1.0	1.0	1.0	1.0
CNN_Model	0.9917991477966913	1.0	0.9973233926453009	0.9971861742420361	0.9959602882250264	0.996453800581811

Figure 5.2: UIA IMU dataset GI Task: Max F1-Averaged Stratified k-Fold model validation

5.2 Results for the Topman dataset GI task

On the publicly available dataset by Topman and colleagues, a through feature selection process wad executed in order to reduce the dimensionality of the Gait identification problem. The available features, among others, were quaternion expressions for IMU sensor attitude, along the standard 9-axis IMU features. The same classifiers where used as in the UIA IMU GI task, and the results can be seen in figure 5.3.

Just like for the UIA IMU dataset GI task, the feature analysis shows a clear dominance of the attitude expressions in the data. The magnetometer readings for the Topman data did however not achieve a significant score, unlike in the UIA IMU data. The machine learning application on the Topman GI task yielded good results for the TCN model, not unlike the results in the UIA IMU GI task. The problem was significantly more difficult to train the models for however, and a perfect score was never reached. A score of 0.98 on the Max F1-Averaged Stratified k-Fold model validation is however considered acceptable given the high number of subjects in the dataset (30 subjects). The fact that a perfect score was not reached, did however result in the dropping of model simplification. This time was instead spent training the model for further improvement in accuracy. The results from the Max F1-Averaged Stratified k-Fold model validation can be seen in figure 5.4.

Topman dataset Feature selection

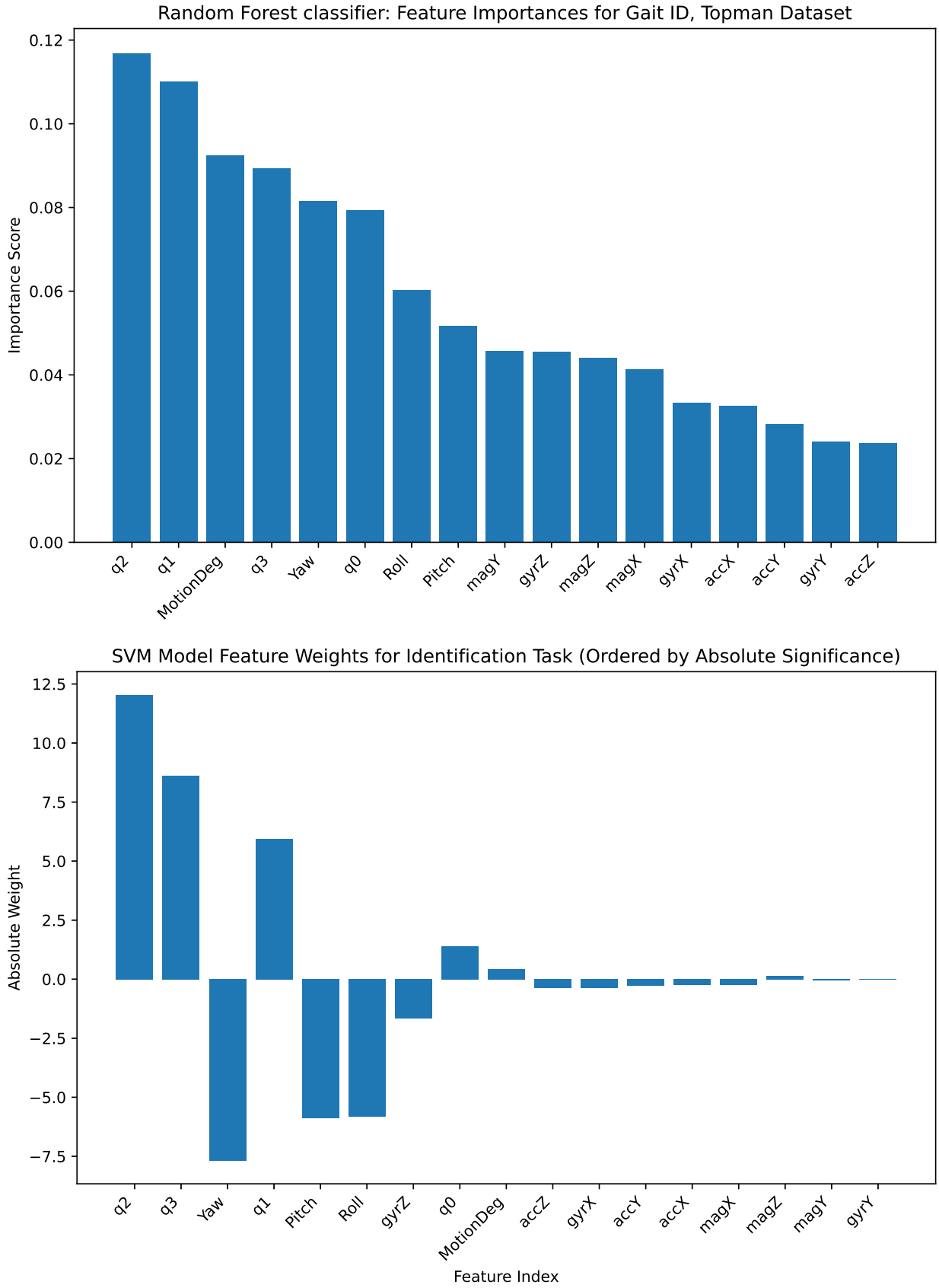
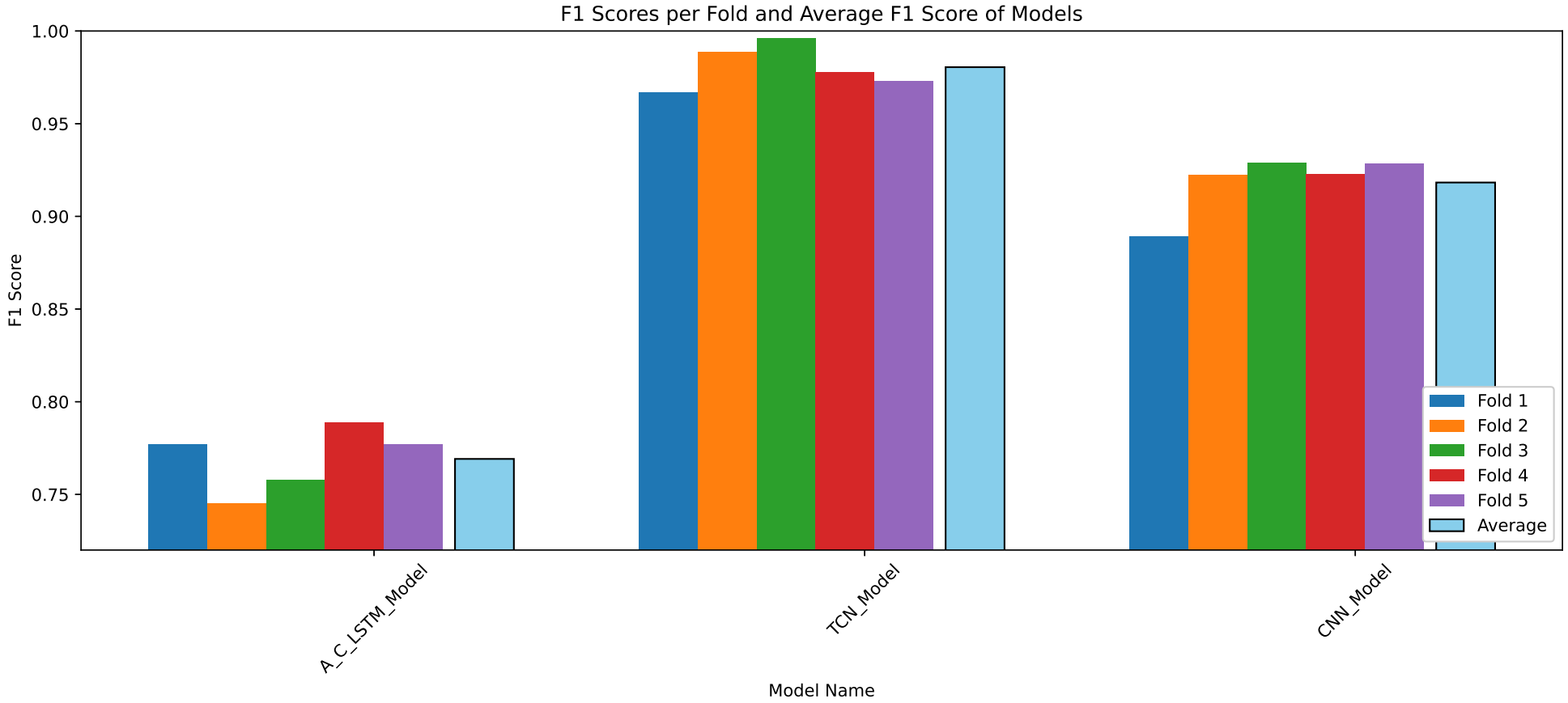


Figure 5.3: Topman dataset Feature selection

Topman dataset GI Task: Max F1-Averaged Stratified k-Fold model validation



Model Name	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average F1
A_C_LSTM_Model	0.7771861305935289	0.7451250906981052	0.7576101883479243	0.7887851274015556	0.777085410226929	0.7691583894536087
TCN_Model	0.9668776343899625	0.9887455502368246	0.9960427719473249	0.9777520753583643	0.9730095256876878	0.980485511524033
CNN_Model	0.8890784909088278	0.922339560707998	0.9286956831388417	0.922642182646403	0.9285938282825523	0.9182699491369245

Figure 5.4: Topman dataset GI Task: Max F1-Averaged Stratified k-Fold model validation

Chapter 6

Future Work

The scope of this thesis proved to be large, and the development of gait detection algorithms and gait detection labeling algorithms had to be put on pause for the completion of the thesis.

In future works, the implementation of the highly-accurate TCN model for GI could be implemented on a system running in real-time on a micro controller or a mobile device.

The GD labeling algorithm could be further developed, and validated through the use of other gait measuring modalities such as the optical motion capture system Qualisys.

Chapter 7

Conclusions

In this body of work, the integration of IMU data and ML algorithms was successfully demonstrated in the context of Gait Identification on two datasets.

The research notably achieved perfect and near-perfect scores in Gait Identity tasks using a Temporal Convolutional Network (TCN) model.

The techniques used for feature selection and dimensionality reduction in ML tasks was successfully implemented in order to make the final product more feasible on resource-constrained platforms like microcontrollers or mobile devices.

This exploration into the synergy between wearable sensor technology and machine learning highlights the possibility for further developing smart healthcare solutions, personal identification systems, and control systems for prosthetic devices.

Appendix A

Appendix: UIA IMU Dataset

A.1 Data wrangling

Initial data wrangling for the UIA IMU walking gait data. The data is stored on a per-measurand basis on the memory card of the sensor board. Need to extrapolate all data, gather all measurands for all users, and make sure all measurands for all users are synchronized (temporal alignment). The data is then gathered in a .csv file for further processing.

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/uia-imu-gait-analysis-dataset/34_GY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/0_MX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/9_MZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/7_GZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/2_GX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/4_MY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/4_MZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/6_MY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/34_AX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/11_MX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/6_MZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/3_MZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/33_GX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/0_AZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/7_GY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/7_MX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/6_AZ.csv
```

```
/kaggle/input/uia-imu-gait-analysis-dataset/33_AZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/9_GZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/4_GZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/6_GY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/12_AX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/2_GZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/9_AY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/8_GZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/3_GZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/5_GX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/1_AX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/2_AZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/33_AY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/8_MZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/11_GX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/33_MX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/9_GX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/12_AZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/2_MY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/10_AX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/11_AY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/9_MY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/11_AZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/3_GX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/2_MX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/10_GY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/6_AY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/4_GY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/7_AZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/4_AZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/10_AZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/6_GX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/0_AY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/35_GY.csv
```

```
/kaggle/input/uia-imu-gait-analysis-dataset/10_MZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/12_MZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/34_GZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/5_GY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/9_AZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/1_MX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/8_GX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/5_MZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/10_MX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/34_MX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/11_AX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/0_AX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/3_AZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/5_MX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/9_GY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/0_MZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/11_MZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/34_MY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/10_GZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/8_AY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/10_GX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/12_GZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/7_GX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/35_AY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/6_GZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/UIA_IMU_9ax_WG_Dataset.csv
/kaggle/input/uia-imu-gait-analysis-dataset/1_MY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/4_MX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/4_AX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/7_AY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/5_GZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/1_GX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/35_MZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/12_GX.csv
```

```
/kaggle/input/uia-imu-gait-analysis-dataset/35_MY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/11_GZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/3_MX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/33_GY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/35_AX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/0_GX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/33_MY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/1_AY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/34_MZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/5_AY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/12_MY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/3_AX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/8_MX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/33_AX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/2_GY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/3_GY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/11_GY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/11_MY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/8_AZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/8_MY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/2_AX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/5_AZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/5_MY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/2_AY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/4_GX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/8_GY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/35_GZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/5_AX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/1_GZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/35_GX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/35_AZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/12_AY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/35_MX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/33_MZ.csv
```

```
/kaggle/input/uia-imu-gait-analysis-dataset/0_GY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/7_MY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/12_MX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/12_GY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/1_AZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/6_AX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/3_AY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/34_AZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/34_GX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/0_GZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/0_MY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/9_MX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/6_MX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/9_AX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/8_AX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/34_AY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/7_AX.csv
/kaggle/input/uia-imu-gait-analysis-dataset/2_MZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/10_MY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/10_AY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/7_MZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/3_MY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/1_MZ.csv
/kaggle/input/uia-imu-gait-analysis-dataset/4_AY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/1_GY.csv
/kaggle/input/uia-imu-gait-analysis-dataset/33_GZ.csv
```

```
list_files  = [
    '/kaggle/input/uia-imu-gait-analysis-dataset/0_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/9_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/7_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/2_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/4_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/4_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/6_MY.csv',
```

```
'/kaggle/input/uia-imu-gait-analysis-dataset/11_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/6_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/3_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/0_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/7_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/7_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/6_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/9_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/4_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/6_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/12_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/2_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/9_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/8_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/3_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/5_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/1_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/2_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/8_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/11_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/9_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/12_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/2_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/10_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/11_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/9_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/11_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/3_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/2_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/10_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/6_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/4_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/7_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/4_AZ.csv',
```

```
'/kaggle/input/uia-imu-gait-analysis-dataset/10_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/6_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/0_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/10_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/12_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/5_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/9_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/1_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/8_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/5_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/10_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/11_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/0_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/3_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/5_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/9_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/0_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/11_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/10_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/8_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/10_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/12_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/7_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/6_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/1_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/4_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/4_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/7_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/5_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/1_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/12_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/11_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/3_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/0_GX.csv',
```

```
'/kaggle/input/uia-imu-gait-analysis-dataset/1_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/5_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/12_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/3_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/8_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/2_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/3_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/11_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/11_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/8_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/8_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/2_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/5_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/5_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/2_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/4_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/8_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/5_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/1_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/12_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/0_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/7_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/12_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/12_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/1_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/6_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/3_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/0_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/0_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/9_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/6_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/9_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/8_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/7_AX.csv',
```

```
'/kaggle/input/uia-imu-gait-analysis-dataset/2_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/10_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/10_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/7_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/3_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/1_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/4_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/1_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/33_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/33_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/33_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/33_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/33_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/33_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/33_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/33_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/33_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/34_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/34_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/34_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/34_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/34_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/34_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/34_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/34_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/34_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/35_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/35_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/35_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/35_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/35_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/35_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/35_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/35_MY.csv',
```

```
'/kaggle/input/uia-imu-gait-analysis-dataset/35_MZ.csv'


]
```

```
list_files
```

```
['/kaggle/input/uia-imu-gait-analysis-dataset/0_MX.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/9_MZ.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/7_GZ.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/2_GX.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/4_MY.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/4_MZ.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/6_MY.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/11_MX.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/6_MZ.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/3_MZ.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/0_AZ.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/7_GY.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/7_MX.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/6_AZ.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/9_GZ.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/4_GZ.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/6_GY.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/12_AX.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/2_GZ.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/9_AY.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/8_GZ.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/3_GZ.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/5_GX.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/1_AX.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/2_AZ.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/8_MZ.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/11_GX.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/9_GX.csv',
```

```
'/kaggle/input/uia-imu-gait-analysis-dataset/12_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/2_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/10_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/11_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/9_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/11_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/3_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/2_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/10_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/6_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/4_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/7_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/4_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/10_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/6_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/0_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/10_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/12_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/5_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/9_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/1_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/8_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/5_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/10_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/11_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/0_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/3_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/5_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/9_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/0_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/11_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/10_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/8_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/10_GX.csv',
```

```
'/kaggle/input/uia-imu-gait-analysis-dataset/12_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/7_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/6_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/1_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/4_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/4_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/7_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/5_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/1_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/12_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/11_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/3_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/0_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/1_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/5_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/12_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/3_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/8_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/2_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/3_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/11_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/11_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/8_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/8_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/2_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/5_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/5_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/2_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/4_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/8_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/5_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/1_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/12_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/0_GY.csv',
```

```
'/kaggle/input/uia-imu-gait-analysis-dataset/7_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/12_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/12_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/1_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/6_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/3_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/0_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/0_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/9_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/6_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/9_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/8_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/7_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/2_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/10_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/10_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/7_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/3_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/1_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/4_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/1_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/33_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/33_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/33_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/33_GX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/33_GY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/33_GZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/33_MX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/33_MY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/33_MZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/34_AX.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/34_AY.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/34_AZ.csv',
'/kaggle/input/uia-imu-gait-analysis-dataset/34_GX.csv',
```

```
 '/kaggle/input/uia-imu-gait-analysis-dataset/34_GY.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/34_GZ.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/34_MX.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/34_MY.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/34_MZ.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/35_AX.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/35_AY.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/35_AZ.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/35_GX.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/35_GY.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/35_GZ.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/35_MX.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/35_MY.csv',
 '/kaggle/input/uia-imu-gait-analysis-dataset/35_MZ.csv']

df_dict = {}



for item in list_files:

    last_slash = item.split('/')[-1]

    indexer_s = last_slash.split('_')[0] #extrapolates subject number from file

    indexer = int(indexer_s)



    measurand_csv = last_slash.split('_')[1] #extrapolates measurand from file example: 1_MZ = magnetic measuremnt in Z direction for subj

    measurand = measurand_csv.split('.')[0]
```

```
    File = pd.read_csv(item)


    Time_Series = File.iloc[:, 0]
    Time_Series.name = f'{measurand}_timestamp'

    Measurand_Series = File.iloc[:, -1]
    Measurand_Series.name = measurand




    if f'df_{indexer}' not in df_dict:

        df_dict[f'df_{indexer}'] = pd.concat([Time_Series, Measurand_Series], axis=1)

    else:

        df_dict[f'df_{indexer}'][Time_Series.name] = Time_Series
        df_dict[f'df_{indexer}'][Measurand_Series.name] = Measurand_Series
```



```
df_dict
```

```
{'df_0':       MX_timestamp  MX  AZ_timestamp     AZ  AY_timestamp     AY  \
 0     1.702897e+09  72  1.702897e+09 -0.254  1.702897e+09  0.962
 1     1.702897e+09  72  1.702897e+09 -0.248  1.702897e+09  0.953
 2     1.702897e+09  72  1.702897e+09 -0.244  1.702897e+09  0.956
 3     1.702897e+09  72  1.702897e+09 -0.245  1.702897e+09  0.967
 4     1.702897e+09  72  1.702897e+09 -0.250  1.702897e+09  0.963
```

```
...             ...  ..            ...    ...            ...    ...
4040  1.702897e+09  76  1.702897e+09 -0.240  1.702897e+09  0.972
4041  1.702897e+09  76  1.702897e+09 -0.248  1.702897e+09  0.968
4042  1.702897e+09  76  1.702897e+09 -0.238  1.702897e+09  0.965
4043  1.702897e+09  76  1.702897e+09 -0.277  1.702897e+09  0.979
4044  1.702897e+09  76  1.702897e+09 -0.259  1.702897e+09  0.970

      AX_timestamp     AX  MZ_timestamp  MZ  GX_timestamp     GX  \
0     1.702897e+09 -0.039  1.702897e+09 -43  1.702897e+09  3.174
1     1.702897e+09 -0.033  1.702897e+09 -43  1.702897e+09  3.357
2     1.702897e+09 -0.046  1.702897e+09 -43  1.702897e+09  3.052
3     1.702897e+09 -0.073  1.702897e+09 -42  1.702897e+09  2.563
4     1.702897e+09 -0.075  1.702897e+09 -42  1.702897e+09  1.892
...            ...    ...           ...  ..           ...    ...
4040  1.702897e+09 -0.027  1.702897e+09 -27  1.702897e+09  5.585
4041  1.702897e+09  0.003  1.702897e+09 -27  1.702897e+09  1.373
4042  1.702897e+09  0.007  1.702897e+09 -27  1.702897e+09  0.580
4043  1.702897e+09 -0.010  1.702897e+09 -27  1.702897e+09  1.282
4044  1.702897e+09 -0.067  1.702897e+09 -28  1.702897e+09  3.082

      GY_timestamp      GY  GZ_timestamp      GZ  MY_timestamp  MY
0     1.702897e+09  -8.148  1.702897e+09  -2.899  1.702897e+09   6
1     1.702897e+09  -6.927  1.702897e+09  -3.479  1.702897e+09   5
2     1.702897e+09  -1.678  1.702897e+09  -3.784  1.702897e+09   8
3     1.702897e+09   2.228  1.702897e+09  -2.960  1.702897e+09   7
4     1.702897e+09   2.563  1.702897e+09  -2.319  1.702897e+09   7
...            ...     ...           ...     ...           ...  ..
4040  1.702897e+09  90.088  1.702897e+09 -41.748  1.702897e+09  27
4041  1.702897e+09  87.830  1.702897e+09 -39.734  1.702897e+09  23
4042  1.702897e+09  92.377  1.702897e+09 -35.553  1.702897e+09  21
4043  1.702897e+09  98.724  1.702897e+09 -28.748  1.702897e+09  19
4044  1.702897e+09  96.405  1.702897e+09 -20.264  1.702897e+09  20

[4045 rows x 18 columns],
```

'df_9':   MZ_timestamp MZ GZ_timestamp   GZ AY_timestamp   AY \
0   1.702896e+09 -37 1.702896e+09   0.031 1.702896e+09 1.002
1   1.702896e+09 -38 1.702896e+09   1.129 1.702896e+09 0.973
2   1.702896e+09 -38 1.702896e+09   2.655 1.702896e+09 0.965
3   1.702896e+09 -37 1.702896e+09   4.028 1.702896e+09 0.955
4   1.702896e+09 -38 1.702896e+09   5.066 1.702896e+09 0.992
...   .
3249 1.702896e+09 -45 1.702896e+09   1.984 1.702896e+09 1.073
3250 1.702896e+09 -46 1.702896e+09   8.575 1.702896e+09 1.107
3251 1.702896e+09 -47 1.702896e+09 16.541 1.702896e+09 0.851
3252 1.702896e+09 -48 1.702896e+09 12.970 1.702896e+09 1.932
3253 1.702896e+09 -47 1.702896e+09 -30.579 1.702896e+09 1.204

      GX_timestamp   GX MY_timestamp MY AZ_timestamp   AZ \
0   1.702896e+09   2.533 1.702896e+09 12 1.702896e+09 0.013
1   1.702896e+09 -0.336 1.702896e+09 12 1.702896e+09 -0.016
2   1.702896e+09 -0.519 1.702896e+09 14 1.702896e+09 -0.021
3   1.702896e+09 -0.458 1.702896e+09 15 1.702896e+09 -0.020
4   1.702896e+09 -1.404 1.702896e+09 14 1.702896e+09 -0.022
...   .
3249 1.702896e+09 22.217 1.702896e+09 23 1.702896e+09 0.195
3250 1.702896e+09 37.659 1.702896e+09 23 1.702896e+09 0.379
3251 1.702896e+09 33.112 1.702896e+09 26 1.702896e+09 0.352
3252 1.702896e+09 28.900 1.702896e+09 25 1.702896e+09 -0.153
3253 1.702896e+09   9.430 1.702896e+09 25 1.702896e+09 -0.916

      GY_timestamp   GY MX_timestamp MX AX_timestamp   AX
0   1.702896e+09   4.913 1.702896e+09 88 1.702896e+09 -0.026
1   1.702896e+09   3.754 1.702896e+09 88 1.702896e+09 -0.030
2   1.702896e+09   -0.793 1.702896e+09 87 1.702896e+09 -0.014
3   1.702896e+09   -3.693 1.702896e+09 87 1.702896e+09 0.008
4   1.702896e+09   -3.479 1.702896e+09 88 1.702896e+09 0.038
...   .
3249 1.702896e+09 129.181 1.702896e+09 84 1.702896e+09 0.572

```
3250  1.702896e+09   88.135  1.702896e+09  84  1.702896e+09  0.499
3251  1.702896e+09   30.884  1.702896e+09  83  1.702896e+09  0.223
3252  1.702896e+09  -33.905  1.702896e+09  82  1.702896e+09 -0.348
3253  1.702896e+09  -54.077  1.702896e+09  82  1.702896e+09  0.024

[3254 rows x 18 columns],
'df_7':       GZ_timestamp     GZ  GY_timestamp     GY  MX_timestamp  MX  \
0     1.702889e+09 -7.385  1.702889e+09  2.625  1.702889e+09  58
1     1.702889e+09 -6.714  1.702889e+09  1.709  1.702889e+09  59
2     1.702889e+09 -6.561  1.702889e+09 -2.747  1.702889e+09  58
3     1.702889e+09 -7.050  1.702889e+09 -6.104  1.702889e+09  59
4     1.702889e+09 -5.310  1.702889e+09 -2.411  1.702889e+09  58
...            ...    ...           ...    ...           ...  ..
2734  1.702889e+09 -2.289  1.702889e+09 -6.592  1.702889e+09  60
2735  1.702889e+09 -1.282  1.702889e+09 -2.014  1.702889e+09  60
2736  1.702889e+09 -0.610  1.702889e+09  5.493  1.702889e+09  61
2737  1.702889e+09 -0.580  1.702889e+09  9.216  1.702889e+09  60
2738  1.702889e+09 -1.068  1.702889e+09  9.796  1.702889e+09  60

      AZ_timestamp     AZ  GX_timestamp     GX  AY_timestamp     AY  \
0     1.702889e+09 -0.063  1.702889e+09  1.221  1.702889e+09  1.001
1     1.702889e+09 -0.099  1.702889e+09 -0.092  1.702889e+09  0.995
2     1.702889e+09 -0.094  1.702889e+09 -1.068  1.702889e+09  0.983
3     1.702889e+09 -0.094  1.702889e+09 -2.594  1.702889e+09  0.999
4     1.702889e+09 -0.098  1.702889e+09 -5.127  1.702889e+09  0.971
...            ...    ...           ...    ...           ...    ...
2734  1.702889e+09 -0.027  1.702889e+09 -1.160  1.702889e+09  0.999
2735  1.702889e+09 -0.062  1.702889e+09 -1.892  1.702889e+09  0.989
2736  1.702889e+09 -0.050  1.702889e+09 -0.763  1.702889e+09  0.987
2737  1.702889e+09 -0.070  1.702889e+09  0.244  1.702889e+09  0.995
2738  1.702889e+09 -0.044  1.702889e+09  0.580  1.702889e+09  0.998

      MY_timestamp  MY  AX_timestamp     AX  MZ_timestamp  MZ
0     1.702889e+09  26  1.702889e+09 -0.019  1.702889e+09 -49
```

```
1     1.702889e+09  26  1.702889e+09 -0.023  1.702889e+09 -49
2     1.702889e+09  25  1.702889e+09 -0.005  1.702889e+09 -49
3     1.702889e+09  25  1.702889e+09  0.035  1.702889e+09 -49
4     1.702889e+09  26  1.702889e+09  0.076  1.702889e+09 -49
...            ...  ..           ...    ...           ...  ..
2734  1.702889e+09  48  1.702889e+09 -0.039  1.702889e+09 -54
2735  1.702889e+09  46  1.702889e+09 -0.032  1.702889e+09 -53
2736  1.702889e+09  47  1.702889e+09 -0.060  1.702889e+09 -53
2737  1.702889e+09  47  1.702889e+09 -0.066  1.702889e+09 -54
2738  1.702889e+09  49  1.702889e+09 -0.103  1.702889e+09 -53

[2739 rows x 18 columns],
'df_2':       GX_timestamp      GX  GZ_timestamp      GZ  AZ_timestamp     AZ  \
0     1.702893e+09  32.227  1.702893e+09 -41.107  1.702893e+09 -0.194
1     1.702893e+09  29.938  1.702893e+09 -51.270  1.702893e+09 -0.143
2     1.702893e+09  33.630  1.702893e+09 -38.879  1.702893e+09 -0.282
3     1.702893e+09  43.060  1.702893e+09 -27.924  1.702893e+09 -0.097
4     1.702893e+09  22.095  1.702893e+09 -36.591  1.702893e+09 -0.137
...            ...     ...           ...     ...           ...    ...
2840  1.702894e+09  29.510  1.702894e+09 -50.720  1.702894e+09 -0.390
2841  1.702894e+09  34.546  1.702894e+09 -38.696  1.702894e+09 -0.213
2842  1.702894e+09  21.393  1.702894e+09 -25.391  1.702894e+09 -0.229
2843  1.702894e+09  10.284  1.702894e+09 -16.235  1.702894e+09 -0.216
2844  1.702894e+09   5.096  1.702894e+09 -13.458  1.702894e+09 -0.378

      MY_timestamp  MY  MX_timestamp  MX  GY_timestamp       GY  AX_timestamp  \
0     1.702893e+09  30  1.702893e+09  61  1.702893e+09   39.062  1.702893e+09
1     1.702893e+09  29  1.702893e+09  60  1.702893e+09   -2.533  1.702893e+09
2     1.702893e+09  26  1.702893e+09  60  1.702893e+09  -30.334  1.702893e+09
3     1.702893e+09  27  1.702893e+09  59  1.702893e+09   -4.608  1.702893e+09
4     1.702893e+09  25  1.702893e+09  59  1.702893e+09   24.872  1.702893e+09
...            ...  ..           ...  ..           ...      ...           ...
2840  1.702894e+09  28  1.702894e+09  64  1.702894e+09  149.017  1.702894e+09
2841  1.702894e+09  25  1.702894e+09  65  1.702894e+09  114.594  1.702894e+09
```

```
2842  1.702894e+09  25  1.702894e+09  65  1.702894e+09   57.831  1.702894e+09
2843  1.702894e+09  24  1.702894e+09  66  1.702894e+09   31.586  1.702894e+09
2844  1.702894e+09  25  1.702894e+09  65  1.702894e+09   42.725  1.702894e+09

         AX  AY_timestamp     AY  MZ_timestamp  MZ
0    -0.266  1.702893e+09  0.988  1.702893e+09 -55
1     0.159  1.702893e+09  1.089  1.702893e+09 -57
2     0.011  1.702893e+09  0.992  1.702893e+09 -56
3     0.024  1.702893e+09  1.144  1.702893e+09 -57
4    -0.012  1.702893e+09  0.947  1.702893e+09 -58
...     ...           ...    ...           ...  ..
2840 -0.190  1.702894e+09  0.895  1.702894e+09 -42
2841 -0.269  1.702894e+09  1.081  1.702894e+09 -43
2842 -0.059  1.702894e+09  1.003  1.702894e+09 -44
2843  0.012  1.702894e+09  0.950  1.702894e+09 -45
2844 -0.073  1.702894e+09  0.892  1.702894e+09 -45

[2845 rows x 18 columns],
'df_4':       MY_timestamp  MY  MZ_timestamp  MZ  GZ_timestamp      GZ  GY_timestamp  \
0     1.702898e+09  11  1.702898e+09 -31  1.702898e+09  38.635  1.702898e+09
1     1.702898e+09  11  1.702898e+09 -33  1.702898e+09  25.482  1.702898e+09
2     1.702898e+09  12  1.702898e+09 -35  1.702898e+09  27.710  1.702898e+09
3     1.702898e+09  15  1.702898e+09 -36  1.702898e+09  33.844  1.702898e+09
4     1.702898e+09  15  1.702898e+09 -36  1.702898e+09  29.449  1.702898e+09
...            ...  ..           ...  ..           ...     ...           ...
2690  1.702898e+09  31  1.702898e+09 -22  1.702898e+09 -52.216  1.702898e+09
2691  1.702898e+09  27  1.702898e+09 -21  1.702898e+09 -64.972  1.702898e+09
2692  1.702898e+09  25  1.702898e+09 -19  1.702898e+09 -63.293  1.702898e+09
2693  1.702898e+09  23  1.702898e+09 -17  1.702898e+09 -15.778  1.702898e+09
2694  1.702898e+09  25  1.702898e+09 -18  1.702898e+09  55.786  1.702898e+09

           GY  AZ_timestamp     AZ  MX_timestamp  MX  AX_timestamp     AX  \
0      97.290  1.702898e+09  0.029  1.702898e+09  81  1.702898e+09  0.009
1     150.024  1.702898e+09  0.123  1.702898e+09  81  1.702898e+09 -0.014
```

```
2     136.047  1.702898e+09  0.084  1.702898e+09  82  1.702898e+09  0.132
3      86.090  1.702898e+09  0.068  1.702898e+09  81  1.702898e+09  0.336
4     106.750  1.702898e+09 -0.076  1.702898e+09  81  1.702898e+09  0.432
...       ...           ...    ...           ...  ..           ...    ...
2690  173.187  1.702898e+09  0.000  1.702898e+09  79  1.702898e+09  0.011
2691  209.717  1.702898e+09  0.181  1.702898e+09  80  1.702898e+09  0.145
2692  226.715  1.702898e+09  0.334  1.702898e+09  80  1.702898e+09  0.393
2693  218.079  1.702898e+09 -0.075  1.702898e+09  80  1.702898e+09  0.084
2694   54.260  1.702898e+09 -0.345  1.702898e+09  79  1.702898e+09 -0.264

      GX_timestamp      GX  AY_timestamp     AY
0     1.702898e+09  44.342  1.702898e+09  0.958
1     1.702898e+09  40.436  1.702898e+09  1.012
2     1.702898e+09  31.921  1.702898e+09  0.960
3     1.702898e+09  13.977  1.702898e+09  0.926
4     1.702898e+09   4.761  1.702898e+09  0.934
...            ...     ...           ...    ...
2690  1.702898e+09  26.428  1.702898e+09  1.096
2691  1.702898e+09  18.158  1.702898e+09  1.110
2692  1.702898e+09 -18.555  1.702898e+09  1.257
2693  1.702898e+09 -26.489  1.702898e+09  1.016
2694  1.702898e+09  16.388  1.702898e+09  0.600

[2695 rows x 18 columns],
'df_6':       MY_timestamp  MY  MZ_timestamp  MZ  AZ_timestamp     AZ  GY_timestamp  \
0     1.702898e+09  24  1.702898e+09 -18  1.702898e+09 -0.138  1.702898e+09
1     1.702898e+09  25  1.702898e+09 -19  1.702898e+09 -0.212  1.702898e+09
2     1.702898e+09  25  1.702898e+09 -21  1.702898e+09 -0.184  1.702898e+09
3     1.702898e+09  23  1.702898e+09 -21  1.702898e+09 -0.132  1.702898e+09
4     1.702898e+09  21  1.702898e+09 -23  1.702898e+09 -0.685  1.702898e+09
...            ...  ..           ...  ..           ...    ...           ...
3019  1.702898e+09  43  1.702898e+09 -24  1.702898e+09 -0.263  1.702898e+09
3020  1.702898e+09  41  1.702898e+09 -24  1.702898e+09 -0.103  1.702898e+09
3021  1.702898e+09  41  1.702898e+09 -22  1.702898e+09  0.091  1.702898e+09
```

```
3022  1.702898e+09  37  1.702898e+09 -22  1.702898e+09  0.058  1.702898e+09
3023  1.702898e+09  35  1.702898e+09 -20  1.702898e+09 -0.127  1.702898e+09

           GY  AY_timestamp     AY  GX_timestamp      GX  GZ_timestamp  \
0     156.952  1.702898e+09  1.024  1.702898e+09  66.406  1.702898e+09
1      99.701  1.702898e+09  0.886  1.702898e+09  50.781  1.702898e+09
2      31.830  1.702898e+09  0.907  1.702898e+09  58.685  1.702898e+09
3      62.317  1.702898e+09  0.736  1.702898e+09  34.698  1.702898e+09
4     107.330  1.702898e+09  0.628  1.702898e+09  38.269  1.702898e+09
...       ...           ...    ...           ...     ...           ...
3019  156.860  1.702898e+09  0.862  1.702898e+09  65.796  1.702898e+09
3020  113.190  1.702898e+09  0.997  1.702898e+09  72.693  1.702898e+09
3021   90.454  1.702898e+09  1.016  1.702898e+09  64.331  1.702898e+09
3022  154.327  1.702898e+09  1.051  1.702898e+09  56.152  1.702898e+09
3023  177.979  1.702898e+09  0.933  1.702898e+09  34.241  1.702898e+09

          GZ  AX_timestamp     AX  MX_timestamp  MX
0     14.679  1.702898e+09  0.174  1.702898e+09  81
1     20.325  1.702898e+09  0.131  1.702898e+09  81
2      0.153  1.702898e+09  0.235  1.702898e+09  81
3    -21.240  1.702898e+09  0.702  1.702898e+09  81
4    -18.707  1.702898e+09  0.430  1.702898e+09  82
...      ...           ...    ...           ...  ..
3019  16.357  1.702898e+09 -0.066  1.702898e+09  78
3020  -5.829  1.702898e+09  0.000  1.702898e+09  79
3021 -32.501  1.702898e+09  0.171  1.702898e+09  79
3022 -53.711  1.702898e+09  0.306  1.702898e+09  81
3023 -53.345  1.702898e+09  0.321  1.702898e+09  82

[3024 rows x 18 columns],
'df_11':       MX_timestamp  MX  GX_timestamp       GX  AY_timestamp     AY  \
0     1.702896e+09  77  1.702896e+09   -8.392  1.702896e+09  0.923
1     1.702896e+09  78  1.702896e+09  -16.846  1.702896e+09  1.265
2     1.702896e+09  77  1.702896e+09  -89.996  1.702896e+09  1.532
```

```
3     1.702896e+09  76  1.702896e+09 -114.197  1.702896e+09  1.432
4     1.702896e+09  76  1.702896e+09  -82.977  1.702896e+09  1.192
...            ...  ..           ...      ...           ...    ...
2976  1.702897e+09  99  1.702897e+09   16.449  1.702897e+09  0.836
2977  1.702897e+09  99  1.702897e+09    9.094  1.702897e+09  0.902
2978  1.702897e+09  99  1.702897e+09    9.155  1.702897e+09  0.931
2979  1.702897e+09  99  1.702897e+09    3.143  1.702897e+09  1.004
2980  1.702897e+09  99  1.702897e+09   -7.996  1.702897e+09  1.220

      AZ_timestamp     AZ  AX_timestamp     AX  MZ_timestamp  MZ  \
0     1.702896e+09 -0.313  1.702896e+09  0.649  1.702896e+09 -45
1     1.702896e+09  0.576  1.702896e+09  0.157  1.702896e+09 -45
2     1.702896e+09 -0.267  1.702896e+09  0.011  1.702896e+09 -44
3     1.702896e+09 -0.197  1.702896e+09  0.082  1.702896e+09 -42
4     1.702896e+09 -0.081  1.702896e+09  0.167  1.702896e+09 -42
...            ...    ...           ...    ...           ...  ..
2976  1.702897e+09 -0.131  1.702897e+09  0.284  1.702897e+09 -32
2977  1.702897e+09 -0.181  1.702897e+09  0.562  1.702897e+09 -33
2978  1.702897e+09 -0.141  1.702897e+09  0.708  1.702897e+09 -32
2979  1.702897e+09  0.049  1.702897e+09  0.569  1.702897e+09 -33
2980  1.702897e+09 -0.010  1.702897e+09  0.677  1.702897e+09 -33

      GZ_timestamp       GZ  GY_timestamp       GY  MY_timestamp  MY
0     1.702896e+09   10.956  1.702896e+09  -87.891  1.702896e+09  28
1     1.702896e+09   49.469  1.702896e+09  -71.930  1.702896e+09  28
2     1.702896e+09  108.185  1.702896e+09 -143.616  1.702896e+09  31
3     1.702896e+09  159.607  1.702896e+09 -166.748  1.702896e+09  36
4     1.702896e+09  174.622  1.702896e+09 -112.701  1.702896e+09  40
...            ...      ...           ...      ...           ...  ..
2976  1.702897e+09  -80.109  1.702897e+09   35.309  1.702897e+09  31
2977  1.702897e+09  -71.411  1.702897e+09   51.727  1.702897e+09  27
2978  1.702897e+09  -58.502  1.702897e+09   90.210  1.702897e+09  27
2979  1.702897e+09  -46.753  1.702897e+09  119.873  1.702897e+09  25
2980  1.702897e+09  -31.616  1.702897e+09  121.429  1.702897e+09  25
```

```
[2981 rows x 18 columns],
'df_3':       MZ_timestamp  MZ  GZ_timestamp      GZ  GX_timestamp      GX  \
0     1.702890e+09 -55  1.702890e+09  -6.073  1.702890e+09   5.341
1     1.702890e+09 -53  1.702890e+09  -5.432  1.702890e+09   3.815
2     1.702890e+09 -54  1.702890e+09  -3.479  1.702890e+09   1.373
3     1.702890e+09 -55  1.702890e+09   0.275  1.702890e+09  -0.763
4     1.702890e+09 -53  1.702890e+09   2.869  1.702890e+09  -3.204
...            ...  ..           ...     ...           ...     ...
2688  1.702890e+09 -44  1.702890e+09 -15.686  1.702890e+09 -65.491
2689  1.702890e+09 -44  1.702890e+09 -19.501  1.702890e+09 -58.289
2690  1.702890e+09 -44  1.702890e+09 -23.468  1.702890e+09 -53.802
2691  1.702890e+09 -46  1.702890e+09 -21.454  1.702890e+09 -38.391
2692  1.702890e+09 -46  1.702890e+09 -21.973  1.702890e+09 -13.794

      AZ_timestamp     AZ  MX_timestamp  MX  AX_timestamp     AX  \
0     1.702890e+09 -0.100  1.702890e+09  71  1.702890e+09  0.026
1     1.702890e+09 -0.098  1.702890e+09  70  1.702890e+09 -0.030
2     1.702890e+09 -0.092  1.702890e+09  71  1.702890e+09 -0.089
3     1.702890e+09 -0.082  1.702890e+09  69  1.702890e+09 -0.093
4     1.702890e+09 -0.098  1.702890e+09  71  1.702890e+09 -0.037
...            ...    ...           ...  ..           ...    ...
2688  1.702890e+09  0.187  1.702890e+09  80  1.702890e+09  0.307
2689  1.702890e+09  0.306  1.702890e+09  80  1.702890e+09  0.391
2690  1.702890e+09  0.288  1.702890e+09  79  1.702890e+09  0.336
2691  1.702890e+09  0.249  1.702890e+09  80  1.702890e+09  0.333
2692  1.702890e+09  0.207  1.702890e+09  80  1.702890e+09  0.349

      GY_timestamp       GY  AY_timestamp     AY  MY_timestamp  MY
0     1.702890e+09   16.632  1.702890e+09  1.008  1.702890e+09  33
1     1.702890e+09   18.555  1.702890e+09  1.021  1.702890e+09  34
2     1.702890e+09   12.207  1.702890e+09  0.993  1.702890e+09  33
3     1.702890e+09   -1.312  1.702890e+09  0.991  1.702890e+09  34
4     1.702890e+09  -19.287  1.702890e+09  1.035  1.702890e+09  32
```

```
...             ...      ...           ...    ...           ...  ..
2688  1.702890e+09 -102.264  1.702890e+09  0.815  1.702890e+09  54
2689  1.702890e+09 -119.537  1.702890e+09  0.895  1.702890e+09  53
2690  1.702890e+09 -125.000  1.702890e+09  0.807  1.702890e+09  54
2691  1.702890e+09 -113.037  1.702890e+09  0.720  1.702890e+09  51
2692  1.702890e+09  -93.903  1.702890e+09  0.753  1.702890e+09  51

[2693 rows x 18 columns],
'df_12':       AX_timestamp     AX  AZ_timestamp     AZ  MZ_timestamp  MZ  \
0     1.702892e+09  0.130  1.702892e+09  0.027  1.702892e+09 -44
1     1.702892e+09  0.128  1.702892e+09  0.046  1.702892e+09 -44
2     1.702892e+09  0.141  1.702892e+09  0.017  1.702892e+09 -44
3     1.702892e+09  0.133  1.702892e+09  0.045  1.702892e+09 -44
4     1.702892e+09  0.140  1.702892e+09  0.038  1.702892e+09 -45
...            ...    ...           ...    ...           ...  ..
3313  1.702892e+09  0.108  1.702892e+09  0.072  1.702892e+09 -27
3314  1.702892e+09  0.107  1.702892e+09  0.062  1.702892e+09 -27
3315  1.702892e+09  0.138  1.702892e+09  0.022  1.702892e+09 -28
3316  1.702892e+09 -0.170  1.702892e+09 -0.428  1.702892e+09 -30
3317  1.702892e+09 -0.529  1.702892e+09  0.341  1.702892e+09 -31

      GZ_timestamp      GZ  GX_timestamp      GX  MY_timestamp  MY  \
0     1.702892e+09   1.251  1.702892e+09  -1.160  1.702892e+09  32
1     1.702892e+09   1.038  1.702892e+09   0.549  1.702892e+09  33
2     1.702892e+09   0.763  1.702892e+09  -0.854  1.702892e+09  33
3     1.702892e+09   0.885  1.702892e+09  -0.031  1.702892e+09  33
4     1.702892e+09   1.068  1.702892e+09  -0.092  1.702892e+09  32
...            ...     ...           ...     ...           ...  ..
3313  1.702892e+09  -3.876  1.702892e+09   2.777  1.702892e+09  41
3314  1.702892e+09   8.179  1.702892e+09  -3.235  1.702892e+09  40
3315  1.702892e+09  20.264  1.702892e+09  32.562  1.702892e+09  41
3316  1.702892e+09  23.132  1.702892e+09  49.622  1.702892e+09  39
3317  1.702892e+09   5.646  1.702892e+09  84.442  1.702892e+09  36
```

```
      AY_timestamp     AY  MX_timestamp  MX  GY_timestamp       GY
0     1.702892e+09  0.983  1.702892e+09  55  1.702892e+09    1.251
1     1.702892e+09  0.989  1.702892e+09  56  1.702892e+09    0.702
2     1.702892e+09  0.985  1.702892e+09  55  1.702892e+09   -0.549
3     1.702892e+09  0.988  1.702892e+09  56  1.702892e+09   -3.143
4     1.702892e+09  0.989  1.702892e+09  57  1.702892e+09   -0.519
...            ...    ...           ...  ..           ...      ...
3313  1.702892e+09  0.880  1.702892e+09  56  1.702892e+09  218.170
3314  1.702892e+09  0.849  1.702892e+09  55  1.702892e+09  181.396
3315  1.702892e+09  0.812  1.702892e+09  55  1.702892e+09  104.889
3316  1.702892e+09  0.953  1.702892e+09  56  1.702892e+09   88.623
3317  1.702892e+09  1.102  1.702892e+09  56  1.702892e+09   87.830

[3318 rows x 18 columns],
'df_8':       GZ_timestamp      GZ  MZ_timestamp  MZ  GX_timestamp      GX  \
0     1.702891e+09   4.181  1.702891e+09 -31  1.702891e+09  -2.899
1     1.702891e+09   3.845  1.702891e+09 -31  1.702891e+09  -2.625
2     1.702891e+09   3.601  1.702891e+09 -30  1.702891e+09  -2.411
3     1.702891e+09   3.510  1.702891e+09 -30  1.702891e+09  -1.526
4     1.702891e+09   2.747  1.702891e+09 -30  1.702891e+09  -1.251
...            ...     ...           ...  ..           ...     ...
2826  1.702891e+09 -37.903  1.702891e+09 -32  1.702891e+09  85.327
2827  1.702891e+09 -34.760  1.702891e+09 -32  1.702891e+09  73.517
2828  1.702891e+09 -48.798  1.702891e+09 -34  1.702891e+09  52.948
2829  1.702891e+09 -54.657  1.702891e+09 -35  1.702891e+09  58.289
2830  1.702891e+09 -65.735  1.702891e+09 -37  1.702891e+09  40.588

      AY_timestamp     AY  MX_timestamp  MX  AZ_timestamp     AZ  \
0     1.702891e+09  1.000  1.702891e+09  43  1.702891e+09  0.007
1     1.702891e+09  0.989  1.702891e+09  44  1.702891e+09  0.011
2     1.702891e+09  0.989  1.702891e+09  43  1.702891e+09  0.011
3     1.702891e+09  0.991  1.702891e+09  43  1.702891e+09  0.010
4     1.702891e+09  0.994  1.702891e+09  43  1.702891e+09  0.015
...            ...    ...           ...  ..           ...    ...
```

```
2826  1.702891e+09  1.219  1.702891e+09  59  1.702891e+09  0.143
2827  1.702891e+09  1.094  1.702891e+09  60  1.702891e+09 -0.287
2828  1.702891e+09  1.064  1.702891e+09  60  1.702891e+09  0.045
2829  1.702891e+09  0.983  1.702891e+09  61  1.702891e+09  0.282
2830  1.702891e+09  1.009  1.702891e+09  61  1.702891e+09 -0.349

      MY_timestamp  MY  GY_timestamp       GY  AX_timestamp     AX
0     1.702891e+09  12  1.702891e+09   -7.233  1.702891e+09  0.093
1     1.702891e+09  14  1.702891e+09   -5.402  1.702891e+09  0.097
2     1.702891e+09  14  1.702891e+09   -2.472  1.702891e+09  0.110
3     1.702891e+09  14  1.702891e+09   -1.465  1.702891e+09  0.100
4     1.702891e+09  14  1.702891e+09   -2.808  1.702891e+09  0.103
...            ...  ..           ...      ...           ...    ...
2826  1.702891e+09  43  1.702891e+09  345.978  1.702891e+09 -1.236
2827  1.702891e+09  41  1.702891e+09  205.444  1.702891e+09 -0.515
2828  1.702891e+09  40  1.702891e+09  -21.332  1.702891e+09  0.400
2829  1.702891e+09  38  1.702891e+09  -74.615  1.702891e+09  0.724
2830  1.702891e+09  34  1.702891e+09  165.955  1.702891e+09 -0.096

[2831 rows x 18 columns],
'df_5':       GX_timestamp      GX  GY_timestamp       GY  MZ_timestamp  MZ  \
0     1.702890e+09   3.418  1.702890e+09  -12.268  1.702890e+09 -59
1     1.702890e+09  -2.014  1.702890e+09  -11.719  1.702890e+09 -59
2     1.702890e+09  -7.629  1.702890e+09  -17.487  1.702890e+09 -59
3     1.702890e+09  -4.852  1.702890e+09  -19.409  1.702890e+09 -59
4     1.702890e+09  -1.709  1.702890e+09   -7.050  1.702890e+09 -58
...            ...     ...           ...      ...           ...  ..
2724  1.702891e+09  29.419  1.702891e+09  138.153  1.702891e+09 -46
2725  1.702891e+09   6.348  1.702891e+09   93.933  1.702891e+09 -46
2726  1.702891e+09   0.214  1.702891e+09   12.939  1.702891e+09 -46
2727  1.702891e+09   4.517  1.702891e+09  -26.245  1.702891e+09 -47
2728  1.702891e+09   3.143  1.702891e+09  -10.590  1.702891e+09 -45

      MX_timestamp  MX  GZ_timestamp      GZ  AY_timestamp     AY  \
```

```
0     1.702890e+09  79  1.702890e+09   2.533  1.702890e+09  0.987
1     1.702890e+09  78  1.702890e+09   3.143  1.702890e+09  1.003
2     1.702890e+09  77  1.702890e+09   3.082  1.702890e+09  0.945
3     1.702890e+09  78  1.702890e+09  -0.488  1.702890e+09  0.951
4     1.702890e+09  78  1.702890e+09  -1.343  1.702890e+09  0.982
...            ...  ..           ...     ...           ...    ...
2724  1.702891e+09  80  1.702891e+09   1.709  1.702891e+09  1.036
2725  1.702891e+09  80  1.702891e+09  12.360  1.702891e+09  1.061
2726  1.702891e+09  79  1.702891e+09   7.446  1.702891e+09  1.051
2727  1.702891e+09  80  1.702891e+09   1.617  1.702891e+09  0.968
2728  1.702891e+09  79  1.702891e+09  -0.854  1.702891e+09  1.018

      AZ_timestamp     AZ  MY_timestamp  MY  AX_timestamp     AX
0     1.702890e+09 -0.191  1.702890e+09  37  1.702890e+09  0.030
1     1.702890e+09 -0.132  1.702890e+09  36  1.702890e+09  0.062
2     1.702890e+09 -0.086  1.702890e+09  37  1.702890e+09  0.149
3     1.702890e+09 -0.094  1.702890e+09  35  1.702890e+09  0.190
4     1.702890e+09 -0.081  1.702890e+09  37  1.702890e+09  0.143
...            ...    ...           ...  ..           ...    ...
2724  1.702891e+09  0.023  1.702891e+09  29  1.702891e+09 -0.145
2725  1.702891e+09 -0.122  1.702891e+09  30  1.702891e+09 -0.287
2726  1.702891e+09 -0.116  1.702891e+09  30  1.702891e+09  0.018
2727  1.702891e+09 -0.034  1.702891e+09  30  1.702891e+09  0.125
2728  1.702891e+09  0.090  1.702891e+09  32  1.702891e+09  0.064

[2729 rows x 18 columns],
'df_1':       AX_timestamp     AX  MX_timestamp  MX  MY_timestamp  MY  GX_timestamp  \
0     1.702892e+09  0.081  1.702892e+09  68  1.702892e+09  28  1.702892e+09
1     1.702892e+09  0.157  1.702892e+09  66  1.702892e+09  29  1.702892e+09
2     1.702892e+09  0.099  1.702892e+09  67  1.702892e+09  30  1.702892e+09
3     1.702892e+09  0.086  1.702892e+09  68  1.702892e+09  29  1.702892e+09
4     1.702892e+09  0.117  1.702892e+09  66  1.702892e+09  30  1.702892e+09
...            ...    ...           ...  ..           ...  ..           ...
3172  1.702893e+09  0.164  1.702893e+09  76  1.702893e+09  29  1.702893e+09
```

```
3173  1.702893e+09  0.217  1.702893e+09  75  1.702893e+09  28  1.702893e+09
3174  1.702893e+09  0.293  1.702893e+09  76  1.702893e+09  29  1.702893e+09
3175  1.702893e+09  0.306  1.702893e+09  75  1.702893e+09  30  1.702893e+09
3176  1.702893e+09  0.298  1.702893e+09  76  1.702893e+09  28  1.702893e+09

          GX  AY_timestamp     AY  GZ_timestamp      GZ  AZ_timestamp     AZ  \
0    -13.885  1.702892e+09  0.940  1.702892e+09  -3.143  1.702892e+09 -0.131
1      0.244  1.702892e+09  0.980  1.702892e+09   0.122  1.702892e+09  0.053
2      6.256  1.702892e+09  0.991  1.702892e+09  -3.204  1.702892e+09  0.052
3      1.770  1.702892e+09  1.000  1.702892e+09  -3.784  1.702892e+09 -0.021
4     -0.610  1.702892e+09  0.992  1.702892e+09  -2.625  1.702892e+09 -0.012
...      ...           ...    ...           ...     ...           ...    ...
3172 -17.609  1.702893e+09  1.014  1.702893e+09  -9.613  1.702893e+09  0.116
3173 -27.893  1.702893e+09  0.989  1.702893e+09 -10.437  1.702893e+09  0.051
3174 -36.102  1.702893e+09  0.965  1.702893e+09  -5.890  1.702893e+09  0.110
3175 -33.966  1.702893e+09  0.975  1.702893e+09   0.580  1.702893e+09  0.133
3176 -26.733  1.702893e+09  0.929  1.702893e+09   9.216  1.702893e+09  0.095

      MZ_timestamp  MZ  GY_timestamp       GY
0     1.702892e+09 -41  1.702892e+09  -35.034
1     1.702892e+09 -41  1.702892e+09  -29.724
2     1.702892e+09 -40  1.702892e+09   -5.280
3     1.702892e+09 -41  1.702892e+09   -0.061
4     1.702892e+09 -41  1.702892e+09   -9.827
...            ...  ..           ...      ...
3172  1.702893e+09 -50  1.702893e+09 -127.625
3173  1.702893e+09 -48  1.702893e+09 -119.598
3174  1.702893e+09 -46  1.702893e+09 -119.751
3175  1.702893e+09 -44  1.702893e+09 -107.361
3176  1.702893e+09 -44  1.702893e+09  -96.527

[3177 rows x 18 columns],
'df_10':       AX_timestamp     AX  GY_timestamp      GY  AZ_timestamp     AZ  \
0     1.702898e+09 -0.044  1.702898e+09   7.111  1.702898e+09 -0.161
```

```
1     1.702898e+09 -0.011  1.702898e+09   3.052  1.702898e+09 -0.101
2     1.702898e+09 -0.088  1.702898e+09   0.244  1.702898e+09 -0.174
3     1.702898e+09 -0.107  1.702898e+09   5.646  1.702898e+09 -0.074
4     1.702898e+09 -0.071  1.702898e+09  14.465  1.702898e+09 -0.152
...            ...    ...           ...     ...           ...    ...
2501  1.702898e+09  0.224  1.702898e+09 -21.088  1.702898e+09 -0.108
2502  1.702898e+09  0.343  1.702898e+09 -36.743  1.702898e+09 -0.160
2503  1.702898e+09  0.139  1.702898e+09 -19.623  1.702898e+09 -0.082
2504  1.702898e+09  0.120  1.702898e+09 -10.376  1.702898e+09 -0.336
2505  1.702898e+09 -0.032  1.702898e+09  15.747  1.702898e+09 -0.031

      MZ_timestamp  MZ  MX_timestamp  MX  GZ_timestamp      GZ  GX_timestamp  \
0     1.702898e+09 -36  1.702898e+09  85  1.702898e+09  -4.120  1.702898e+09
1     1.702898e+09 -37  1.702898e+09  85  1.702898e+09  -2.777  1.702898e+09
2     1.702898e+09 -36  1.702898e+09  84  1.702898e+09  -4.486  1.702898e+09
3     1.702898e+09 -37  1.702898e+09  84  1.702898e+09  -7.507  1.702898e+09
4     1.702898e+09 -37  1.702898e+09  84  1.702898e+09  -8.698  1.702898e+09
...            ...  ..           ...  ..           ...     ...           ...
2501  1.702898e+09 -42  1.702898e+09  79  1.702898e+09 -34.485  1.702898e+09
2502  1.702898e+09 -41  1.702898e+09  79  1.702898e+09 -13.763  1.702898e+09
2503  1.702898e+09 -38  1.702898e+09  78  1.702898e+09  22.156  1.702898e+09
2504  1.702898e+09 -37  1.702898e+09  82  1.702898e+09  64.514  1.702898e+09
2505  1.702898e+09 -35  1.702898e+09  83  1.702898e+09  99.304  1.702898e+09

          GX  MY_timestamp  MY  AY_timestamp     AY
0     -1.556  1.702898e+09   3  1.702898e+09  0.975
1      1.831  1.702898e+09   3  1.702898e+09  0.972
2     -2.289  1.702898e+09   4  1.702898e+09  0.975
3     -5.951  1.702898e+09   2  1.702898e+09  0.998
4     -4.883  1.702898e+09   3  1.702898e+09  0.999
...      ...           ...  ..           ...    ...
2501   9.644  1.702898e+09  -2  1.702898e+09  0.948
2502 -17.059  1.702898e+09  -3  1.702898e+09  0.807
2503 -45.624  1.702898e+09  -4  1.702898e+09  1.359
```

```
2504 -72.723  1.702898e+09  -1  1.702898e+09  1.170
2505 -73.730  1.702898e+09   0  1.702898e+09  1.358

[2506 rows x 18 columns],
'df_33':      AX_timestamp     AX  AY_timestamp     AY  AZ_timestamp     AZ  \
0    1.703156e+09 -0.039  1.703156e+09  1.004  1.703156e+09  0.055
1    1.703156e+09 -0.044  1.703156e+09  1.003  1.703156e+09  0.055
2    1.703156e+09 -0.042  1.703156e+09  1.010  1.703156e+09  0.050
3    1.703156e+09 -0.047  1.703156e+09  0.995  1.703156e+09  0.062
4    1.703156e+09 -0.050  1.703156e+09  1.000  1.703156e+09  0.049
..            ...    ...           ...    ...           ...    ...
867  1.703156e+09 -0.043  1.703156e+09  1.002  1.703156e+09  0.032
868  1.703156e+09 -0.045  1.703156e+09  1.002  1.703156e+09  0.029
869  1.703156e+09 -0.045  1.703156e+09  1.002  1.703156e+09  0.024
870  1.703156e+09 -0.042  1.703156e+09  1.005  1.703156e+09  0.023
871  1.703156e+09 -0.045  1.703156e+09  1.002  1.703156e+09  0.038

     GX_timestamp     GX  GY_timestamp     GY  GZ_timestamp     GZ  \
0    1.703156e+09  1.953  1.703156e+09 -0.366  1.703156e+09  0.153
1    1.703156e+09  0.793  1.703156e+09 -0.305  1.703156e+09  0.031
2    1.703156e+09  2.319  1.703156e+09 -0.061  1.703156e+09  0.092
3    1.703156e+09  1.282  1.703156e+09 -0.244  1.703156e+09  0.458
4    1.703156e+09  0.824  1.703156e+09 -0.458  1.703156e+09  0.458
..            ...    ...           ...    ...           ...    ...
867  1.703156e+09  1.404  1.703156e+09 -0.458  1.703156e+09  0.031
868  1.703156e+09  1.678  1.703156e+09 -0.397  1.703156e+09  0.092
869  1.703156e+09  2.075  1.703156e+09 -0.397  1.703156e+09  0.092
870  1.703156e+09  1.404  1.703156e+09 -0.488  1.703156e+09  0.153
871  1.703156e+09 -1.556  1.703156e+09 -0.488  1.703156e+09  0.092

     MX_timestamp  MX  MY_timestamp  MY  MZ_timestamp  MZ
0    1.703156e+09  67  1.703156e+09  24  1.703156e+09 -24
1    1.703156e+09  67  1.703156e+09  26  1.703156e+09 -25
2    1.703156e+09  66  1.703156e+09  26  1.703156e+09 -25
```

```
3    1.703156e+09  66  1.703156e+09  28  1.703156e+09 -26
4    1.703156e+09  66  1.703156e+09  27  1.703156e+09 -24
..            ...  ..           ...  ..           ...  ..
867  1.703156e+09  67  1.703156e+09  29  1.703156e+09 -27
868  1.703156e+09  69  1.703156e+09  28  1.703156e+09 -27
869  1.703156e+09  69  1.703156e+09  29  1.703156e+09 -27
870  1.703156e+09  69  1.703156e+09  26  1.703156e+09 -28
871  1.703156e+09  68  1.703156e+09  29  1.703156e+09 -27

[872 rows x 18 columns],
'df_34':      AX_timestamp     AX  AY_timestamp     AY  AZ_timestamp     AZ  \
0    1.703156e+09 -0.041  1.703156e+09  1.000  1.703156e+09  0.021
1    1.703156e+09 -0.033  1.703156e+09  1.003  1.703156e+09  0.033
2    1.703156e+09 -0.033  1.703156e+09  1.012  1.703156e+09  0.052
3    1.703156e+09 -0.044  1.703156e+09  0.995  1.703156e+09  0.025
4    1.703156e+09 -0.036  1.703156e+09  1.009  1.703156e+09 -0.015
..            ...    ...           ...    ...           ...    ...
697  1.703156e+09 -0.043  1.703156e+09  1.007  1.703156e+09  0.046
698  1.703156e+09 -0.040  1.703156e+09  1.006  1.703156e+09  0.047
699  1.703156e+09 -0.041  1.703156e+09  0.996  1.703156e+09  0.052
700  1.703156e+09 -0.047  1.703156e+09  1.003  1.703156e+09  0.045
701  1.703156e+09 -0.042  1.703156e+09  0.998  1.703156e+09  0.055

     GX_timestamp      GX  GY_timestamp     GY  GZ_timestamp     GZ  \
0    1.703156e+09   1.740  1.703156e+09 -0.305  1.703156e+09 -0.366
1    1.703156e+09  -3.662  1.703156e+09 -0.854  1.703156e+09  0.122
2    1.703156e+09  -3.387  1.703156e+09 -0.732  1.703156e+09  0.061
3    1.703156e+09   8.759  1.703156e+09 -0.244  1.703156e+09 -1.068
4    1.703156e+09  12.451  1.703156e+09 -0.275  1.703156e+09  0.488
..            ...     ...           ...    ...           ...    ...
697  1.703156e+09   1.770  1.703156e+09 -0.488  1.703156e+09 -0.397
698  1.703156e+09   0.641  1.703156e+09 -0.458  1.703156e+09  0.549
699  1.703156e+09  -0.427  1.703156e+09 -0.427  1.703156e+09  1.068
700  1.703156e+09   0.702  1.703156e+09 -0.427  1.703156e+09 -0.092
```

```
701  1.703156e+09   0.214  1.703156e+09 -0.397  1.703156e+09  0.000

     MX_timestamp  MX  MY_timestamp  MY  MZ_timestamp  MZ
0    1.703156e+09  68  1.703156e+09  27  1.703156e+09 -29
1    1.703156e+09  67  1.703156e+09  28  1.703156e+09 -27
2    1.703156e+09  67  1.703156e+09  28  1.703156e+09 -29
3    1.703156e+09  68  1.703156e+09  29  1.703156e+09 -29
4    1.703156e+09  69  1.703156e+09  28  1.703156e+09 -28
..            ...  ..           ...  ..           ...  ..
697  1.703156e+09  67  1.703156e+09  30  1.703156e+09 -27
698  1.703156e+09  68  1.703156e+09  27  1.703156e+09 -27
699  1.703156e+09  67  1.703156e+09  29  1.703156e+09 -27
700  1.703156e+09  68  1.703156e+09  29  1.703156e+09 -27
701  1.703156e+09  69  1.703156e+09  25  1.703156e+09 -28

[702 rows x 18 columns],
'df_35':      AX_timestamp     AX  AY_timestamp     AY  AZ_timestamp     AZ  \
0    1.703156e+09 -0.041  1.703156e+09  1.000  1.703156e+09  0.045
1    1.703156e+09 -0.042  1.703156e+09  1.000  1.703156e+09  0.043
2    1.703156e+09 -0.035  1.703156e+09  1.006  1.703156e+09  0.034
3    1.703156e+09 -0.038  1.703156e+09  1.003  1.703156e+09  0.043
4    1.703156e+09 -0.043  1.703156e+09  1.000  1.703156e+09  0.049
..            ...    ...           ...    ...           ...    ...
844  1.703156e+09 -0.189  1.703156e+09  0.072  1.703156e+09  0.999
845  1.703156e+09 -0.255  1.703156e+09  0.031  1.703156e+09  1.004
846  1.703156e+09 -0.244  1.703156e+09  0.052  1.703156e+09  1.010
847  1.703156e+09 -0.274  1.703156e+09  0.077  1.703156e+09  1.010
848  1.703156e+09 -0.254  1.703156e+09  0.031  1.703156e+09  1.010

     GX_timestamp      GX  GY_timestamp      GY  GZ_timestamp      GZ  \
0    1.703156e+09   1.312  1.703156e+09  -0.427  1.703156e+09   0.061
1    1.703156e+09   1.404  1.703156e+09  -0.427  1.703156e+09  -0.244
2    1.703156e+09   2.380  1.703156e+09  -0.519  1.703156e+09   0.031
3    1.703156e+09   0.580  1.703156e+09  -0.458  1.703156e+09   0.214
```

```
4    1.703156e+09  -2.106  1.703156e+09  -0.397  1.703156e+09   0.031
..            ...     ...           ...     ...           ...     ...
844  1.703156e+09  13.672  1.703156e+09  37.079  1.703156e+09 -21.240
845  1.703156e+09 -17.303  1.703156e+09  29.327  1.703156e+09  -9.979
846  1.703156e+09 -23.804  1.703156e+09 -11.383  1.703156e+09   3.906
847  1.703156e+09   0.366  1.703156e+09 -21.576  1.703156e+09  11.383
848  1.703156e+09   6.989  1.703156e+09   9.460  1.703156e+09  13.062

     MX_timestamp  MX  MY_timestamp  MY  MZ_timestamp  MZ
0    1.703156e+09  68  1.703156e+09  28  1.703156e+09 -28
1    1.703156e+09  68  1.703156e+09  29  1.703156e+09 -27
2    1.703156e+09  69  1.703156e+09  29  1.703156e+09 -27
3    1.703156e+09  68  1.703156e+09  29  1.703156e+09 -27
4    1.703156e+09  67  1.703156e+09  28  1.703156e+09 -28
..            ...  ..           ...  ..           ...  ..
844  1.703156e+09  30  1.703156e+09  21  1.703156e+09  14
845  1.703156e+09  31  1.703156e+09  18  1.703156e+09  14
846  1.703156e+09  29  1.703156e+09  19  1.703156e+09  14
847  1.703156e+09  30  1.703156e+09  21  1.703156e+09  14
848  1.703156e+09  29  1.703156e+09  20  1.703156e+09  14

[849 rows x 18 columns]}
```

The sensor board has multiple sensors with their own internal clock. Keeping track of the timestamp for eachmeasurand, and comparing them to see if data is synchronized across the different sensors. (Acellerometer, Gyroscope, Magnetometer).

```
desired_order = ['AX', 'AX_timestamp', 'AY', 'AY_timestamp', 'AZ', 'AZ_timestamp',
                 'GX', 'GX_timestamp', 'GY', 'GY_timestamp', 'GZ', 'GZ_timestamp',
                 'MX', 'MX_timestamp', 'MY', 'MY_timestamp', 'MZ', 'MZ_timestamp']


def reorder(df, desired_order):

    columns =  [column for column in desired_order if column in df.columns]
    return df[columns]
```

```
#access the dictionary

for key, df in df_dict.items():

    df_dict[key] = reorder(df, desired_order)


df_dict['df_33']
```

```
        AX  AX_timestamp     AY  AY_timestamp     AZ  AZ_timestamp     GX  \
0   -0.039  1.703156e+09  1.004  1.703156e+09  0.055  1.703156e+09  1.953
1   -0.044  1.703156e+09  1.003  1.703156e+09  0.055  1.703156e+09  0.793
2   -0.042  1.703156e+09  1.010  1.703156e+09  0.050  1.703156e+09  2.319
3   -0.047  1.703156e+09  0.995  1.703156e+09  0.062  1.703156e+09  1.282
4   -0.050  1.703156e+09  1.000  1.703156e+09  0.049  1.703156e+09  0.824
..     ...           ...    ...           ...    ...           ...    ...
867 -0.043  1.703156e+09  1.002  1.703156e+09  0.032  1.703156e+09  1.404
868 -0.045  1.703156e+09  1.002  1.703156e+09  0.029  1.703156e+09  1.678
869 -0.045  1.703156e+09  1.002  1.703156e+09  0.024  1.703156e+09  2.075
870 -0.042  1.703156e+09  1.005  1.703156e+09  0.023  1.703156e+09  1.404
871 -0.045  1.703156e+09  1.002  1.703156e+09  0.038  1.703156e+09 -1.556

     GX_timestamp     GY  GY_timestamp     GZ  GZ_timestamp  MX  MX_timestamp  \
0    1.703156e+09 -0.366  1.703156e+09  0.153  1.703156e+09  67  1.703156e+09
1    1.703156e+09 -0.305  1.703156e+09  0.031  1.703156e+09  67  1.703156e+09
2    1.703156e+09 -0.061  1.703156e+09  0.092  1.703156e+09  66  1.703156e+09
3    1.703156e+09 -0.244  1.703156e+09  0.458  1.703156e+09  66  1.703156e+09
4    1.703156e+09 -0.458  1.703156e+09  0.458  1.703156e+09  66  1.703156e+09
..            ...    ...           ...    ...           ...  ..           ...
867  1.703156e+09 -0.458  1.703156e+09  0.031  1.703156e+09  67  1.703156e+09
868  1.703156e+09 -0.397  1.703156e+09  0.092  1.703156e+09  69  1.703156e+09
869  1.703156e+09 -0.397  1.703156e+09  0.092  1.703156e+09  69  1.703156e+09
870  1.703156e+09 -0.488  1.703156e+09  0.153  1.703156e+09  69  1.703156e+09
871  1.703156e+09 -0.488  1.703156e+09  0.092  1.703156e+09  68  1.703156e+09
```

```
     MY  MY_timestamp  MZ  MZ_timestamp
0    24  1.703156e+09 -24  1.703156e+09
1    26  1.703156e+09 -25  1.703156e+09
2    26  1.703156e+09 -25  1.703156e+09
3    28  1.703156e+09 -26  1.703156e+09
4    27  1.703156e+09 -24  1.703156e+09
..   ..           ...  ..           ...
867  29  1.703156e+09 -27  1.703156e+09
868  28  1.703156e+09 -27  1.703156e+09
869  29  1.703156e+09 -27  1.703156e+09
870  26  1.703156e+09 -28  1.703156e+09
871  29  1.703156e+09 -27  1.703156e+09

[872 rows x 18 columns]
```

```
df_dict['df_0'].iloc[-1]['AY_timestamp']
```

```
1702897317.060722
```

```
df_dict['df_0'].iloc[0]['AY_timestamp']
```

```
1702897155.583326
```

Now all data for all subjects are gathered in separate df's.

Need to check if the measurands have identical timestamp for each row in all df's, before concatinating.

```
Erroneous_Lines = {}

def Check_alignment(df_dict):
    for key, df in df_dict.items():
        time_cols = [column for column in df.columns if column.endswith('_timestamp')]

        for i in range(len(df)):
            timestamps = [df[column].iloc[i] for column in time_cols]

            if len(set(timestamps)) != 1:
```

```
                if key not in Erroneous_Lines:
                    Erroneous_Lines[key] = []
                Erroneous_Lines[key].append((i, timestamps))

    return Erroneous_Lines

# Run the check
Errors = Check_alignment(df_dict)

if Errors:
    for key, rows in Errors.items():
        print(f'Issues in {key}:')
        for row, timestamps in rows:
            print(f'Row {row}: {timestamps}')
else:
    print("All timestamps are aligned across all df's.")
```

```
Issues in df_0:
Row 670: [1702897182.204687, 1702897182.204687, 1702897182.204687, 1702897182.204343, 1702897182.204343, ...
   1702897182.204343, 1702897182.204343, 1702897182.204343, 1702897182.204343]
Row 671: [1702897182.244935, 1702897182.244935, 1702897182.244935, 1702897182.244247, 1702897182.244247, ...
   1702897182.244247, 1702897182.244247, 1702897182.244247, 1702897182.244247]
Row 672: [1702897182.284838, 1702897182.284838, 1702897182.284838, 1702897182.284151, 1702897182.284151, ...
   1702897182.284151, 1702897182.284151, 1702897182.284151, 1702897182.284151]
Row 673: [1702897182.324742, 1702897182.324742, 1702897182.324742, 1702897182.324398, 1702897182.324398, ...
   1702897182.324398, 1702897182.324398, 1702897182.324398, 1702897182.324398]
Issues in df_9:
Row 2020: [1702896119.684585, 1702896119.684585, 1702896119.684585, 1702896119.684585, 1702896119.684757, ...
   1702896119.684757, 1702896119.684757, 1702896119.684757, 1702896119.684757]
Row 2021: [1702896119.724305, 1702896119.724305, 1702896119.724305, 1702896119.724305, 1702896119.724649, ...
   1702896119.724649, 1702896119.724649, 1702896119.724649, 1702896119.724649]
Row 2022: [1702896119.764197, 1702896119.764197, 1702896119.764197, 1702896119.764197, 1702896119.764541, ...
   1702896119.764541, 1702896119.764541, 1702896119.764541, 1702896119.764541]
Row 2023: [1702896119.80409, 1702896119.80409, 1702896119.80409, 1702896119.80409, 1702896119.804262, ...
   1702896119.804262, 1702896119.804262, 1702896119.804262, 1702896119.804262]
```

```
Issues in df_4:
Row 803: [1702897919.308827, 1702897919.308827, 1702897919.308827, 1702897919.308827, 1702897919.308827, ...
    1702897919.308827, 1702897919.308999, 1702897919.308999, 1702897919.308999]
Row 804: [1702897919.348547, 1702897919.348547, 1702897919.348547, 1702897919.348547, 1702897919.348547, ...
    1702897919.348547, 1702897919.348891, 1702897919.348891, 1702897919.348891]
Row 1042: [1702897928.858675, 1702897928.858675, 1702897928.858675, 1702897928.858675, 1702897928.858331, ...
    1702897928.858331, 1702897928.858331, 1702897928.858331, 1702897928.858331]
Row 1043: [1702897928.898911, 1702897928.898911, 1702897928.898911, 1702897928.898911, 1702897928.898223, ...
    1702897928.898223, 1702897928.898223, 1702897928.898223, 1702897928.898223]
Row 1044: [1702897928.938803, 1702897928.938803, 1702897928.938803, 1702897928.938803, 1702897928.938115, ...
    1702897928.938115, 1702897928.938115, 1702897928.938115, 1702897928.938115]
Row 1045: [1702897928.978695, 1702897928.978695, 1702897928.978695, 1702897928.978695, 1702897928.978351, ...
    1702897928.978351, 1702897928.978351, 1702897928.978351, 1702897928.978351]
Row 2385: [1702897982.520537, 1702897982.520537, 1702897982.520537, 1702897982.520709, 1702897982.520709, ...
    1702897982.520709, 1702897982.520709, 1702897982.520709, 1702897982.520709]
Row 2386: [1702897982.560257, 1702897982.560257, 1702897982.560257, 1702897982.560601, 1702897982.560601, ...
    1702897982.560601, 1702897982.560601, 1702897982.560601, 1702897982.560601]
Issues in df_3:
Row 1372: [1702889676.588383, 1702889676.588383, 1702889676.588383, 1702889676.588383, 1702889676.588383, ...
    1702889676.588383, 1702889676.588383, 1702889676.588383, 1702889676.588211]
Row 1373: [1702889676.628114, 1702889676.628114, 1702889676.628114, 1702889676.628114, 1702889676.628114, ...
    1702889676.628114, 1702889676.628114, 1702889676.628114, 1702889676.62777]
Row 1374: [1702889676.668018, 1702889676.668018, 1702889676.668018, 1702889676.668018, 1702889676.668018, ...
    1702889676.668018, 1702889676.668018, 1702889676.668018, 1702889676.66733]
Row 1375: [1702889676.707921, 1702889676.707921, 1702889676.707921, 1702889676.707921, 1702889676.707921, ...
    1702889676.707921, 1702889676.707921, 1702889676.707921, 1702889676.707577]
Issues in df_8:
Row 466: [1702891390.18587, 1702891390.18587, 1702891390.18587, 1702891390.186042, 1702891390.186042, ...
    1702891390.186042, 1702891390.186042, 1702891390.186042, 1702891390.186042]
Row 467: [1702891390.225587, 1702891390.225587, 1702891390.225587, 1702891390.225931, 1702891390.225931, ...
    1702891390.225931, 1702891390.225931, 1702891390.225931, 1702891390.225931]
Row 468: [1702891390.265132, 1702891390.265132, 1702891390.265132, 1702891390.26582, 1702891390.26582, ...
    1702891390.26582, 1702891390.26582, 1702891390.26582, 1702891390.26582]
Row 469: [1702891390.304677, 1702891390.304677, 1702891390.304677, 1702891390.305021, 1702891390.305021, ...
    1702891390.305021, 1702891390.305021, 1702891390.305021, 1702891390.305021]
Issues in df_1:
Row 134: [1702892384.067103, 1702892384.067103, 1702892384.067103, 1702892384.067103, 1702892384.067103, ...
    1702892384.067103, 1702892384.066931, 1702892384.066931, 1702892384.066931]
Row 135: [1702892384.10683, 1702892384.10683, 1702892384.10683, 1702892384.10683, 1702892384.10683, 1702892384.10683, ...
```

```
    1702892384.106486, 1702892384.106486, 1702892384.106486]
Row 136: [1702892384.146729, 1702892384.146729, 1702892384.146729, 1702892384.146729, 1702892384.146729, ...
    1702892384.146729, 1702892384.146041, 1702892384.146041, 1702892384.146041]
Row 137: [1702892384.186628, 1702892384.186628, 1702892384.186628, 1702892384.186628, 1702892384.186628, ...
    1702892384.186628, 1702892384.186284, 1702892384.186284, 1702892384.186284]
Row 1576: [1702892441.701734, 1702892441.701734, 1702892441.701734, 1702892441.701391, 1702892441.701391, ...
    1702892441.701391, 1702892441.701391, 1702892441.701391, 1702892441.701391]
Row 1577: [1702892441.741977, 1702892441.741977, 1702892441.741977, 1702892441.741289, 1702892441.741289, ...
    1702892441.741289, 1702892441.741289, 1702892441.741289, 1702892441.741289]
Row 1578: [1702892441.781876, 1702892441.781876, 1702892441.781876, 1702892441.781188, 1702892441.781188, ...
    1702892441.781188, 1702892441.781188, 1702892441.781188, 1702892441.781188]
Row 1579: [1702892441.821775, 1702892441.821775, 1702892441.821775, 1702892441.821431, 1702892441.821431, ...
    1702892441.821431, 1702892441.821431, 1702892441.821431, 1702892441.821431]
Issues in df_10:
Row 1165: [1702898298.376473, 1702898298.376473, 1702898298.376473, 1702898298.377161, 1702898298.377161, ...
    1702898298.377161, 1702898298.377161, 1702898298.377161, 1702898298.377161]
Row 1166: [1702898298.416032, 1702898298.416032, 1702898298.416032, 1702898298.416376, 1702898298.416376, ...
    1702898298.416376, 1702898298.416376, 1702898298.416376, 1702898298.416376]
```



```python
for key, rows in Errors.items():
    print(f'Issues in {key}:')
    for row, timestamps in rows:
        max_timestamp = max(timestamps)
        min_timestamp = min(timestamps)
        difference = max_timestamp - min_timestamp
        print(f'Row {row}: Difference = {difference} seconds')
```

```
Issues in df_0:
Row 670: Difference = 0.0003440380096435547 seconds
Row 671: Difference = 0.0006880760192871094 seconds
Row 672: Difference = 0.0006868839263916016 seconds
Row 673: Difference = 0.0003440380096435547 seconds
Issues in df_9:
Row 2020: Difference = 0.00017189979553222656 seconds
Row 2021: Difference = 0.0003440380096435547 seconds
Row 2022: Difference = 0.0003437995910644531 seconds
Row 2023: Difference = 0.00017189979553222656 seconds
```

```
Issues in df_4:
Row 803: Difference = 0.00017213821411132812 seconds
Row 804: Difference = 0.0003440380096435547 seconds
Row 1042: Difference = 0.0003440380096435547 seconds
Row 1043: Difference = 0.0006880760192871094 seconds
Row 1044: Difference = 0.0006880760192871094 seconds
Row 1045: Difference = 0.0003437995910644531 seconds
Row 2385: Difference = 0.00017213821411132812 seconds
Row 2386: Difference = 0.0003440380096435547 seconds
Issues in df_3:
Row 1372: Difference = 0.00017189979553222656 seconds
Row 1373: Difference = 0.0003440380096435547 seconds
Row 1374: Difference = 0.0006880760192871094 seconds
Row 1375: Difference = 0.0003440380096435547 seconds
Issues in df_8:
Row 466: Difference = 0.00017213821411132812 seconds
Row 467: Difference = 0.0003440380096435547 seconds
Row 468: Difference = 0.0006880760192871094 seconds
Row 469: Difference = 0.0003440380096435547 seconds
Issues in df_1:
Row 134: Difference = 0.00017189979553222656 seconds
Row 135: Difference = 0.0003437995910644531 seconds
Row 136: Difference = 0.0006880760192871094 seconds
Row 137: Difference = 0.0003440380096435547 seconds
Row 1576: Difference = 0.00034308433532714844 seconds
Row 1577: Difference = 0.0006880760192871094 seconds
Row 1578: Difference = 0.0006880760192871094 seconds
Row 1579: Difference = 0.0003440380096435547 seconds
Issues in df_10:
Row 1165: Difference = 0.0006880760192871094 seconds
Row 1166: Difference = 0.0003440380096435547 seconds
```

Found variations in timestamp on several rows in the df's. The differences are however small, and we can probably neglect these. Using the timestamp from one of the acc measurands for all measurands.

```
df_combined = pd.DataFrame()

for subject_key, subject_df in df_dict.items():

    subject_number = int(subject_key.split('_')[1]) #as in df_0 = subject 0

    subject_df['Subject_no'] = subject_number


    df_combined = pd.concat([df_combined, subject_df], ignore_index=False)
```

```
/tmp/ipykernel_43/3828879222.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: ...
   https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  subject_df['Subject_no'] = subject_number
```

```
df_combined
```

	AX	AX_timestamp	AY	AY_timestamp	AZ	AZ_timestamp	GX	\
0	-0.039	1.702897e+09	0.962	1.702897e+09	-0.254	1.702897e+09	3.174	
1	-0.033	1.702897e+09	0.953	1.702897e+09	-0.248	1.702897e+09	3.357	
2	-0.046	1.702897e+09	0.956	1.702897e+09	-0.244	1.702897e+09	3.052	
3	-0.073	1.702897e+09	0.967	1.702897e+09	-0.245	1.702897e+09	2.563	
4	-0.075	1.702897e+09	0.963	1.702897e+09	-0.250	1.702897e+09	1.892	
..	...	...	...	...	...	...	...	
844	-0.189	1.703156e+09	0.072	1.703156e+09	0.999	1.703156e+09	13.672	
845	-0.255	1.703156e+09	0.031	1.703156e+09	1.004	1.703156e+09	-17.303	
846	-0.244	1.703156e+09	0.052	1.703156e+09	1.010	1.703156e+09	-23.804	
847	-0.274	1.703156e+09	0.077	1.703156e+09	1.010	1.703156e+09	0.366	
848	-0.254	1.703156e+09	0.031	1.703156e+09	1.010	1.703156e+09	6.989	

```
     GX_timestamp      GY  GY_timestamp      GZ  GZ_timestamp  MX  \
0    1.702897e+09  -8.148  1.702897e+09  -2.899  1.702897e+09  72
1    1.702897e+09  -6.927  1.702897e+09  -3.479  1.702897e+09  72
2    1.702897e+09  -1.678  1.702897e+09  -3.784  1.702897e+09  72
3    1.702897e+09   2.228  1.702897e+09  -2.960  1.702897e+09  72
4    1.702897e+09   2.563  1.702897e+09  -2.319  1.702897e+09  72
..            ...     ...           ...     ...           ...  ..
844  1.703156e+09  37.079  1.703156e+09 -21.240  1.703156e+09  30
845  1.703156e+09  29.327  1.703156e+09  -9.979  1.703156e+09  31
846  1.703156e+09 -11.383  1.703156e+09   3.906  1.703156e+09  29
847  1.703156e+09 -21.576  1.703156e+09  11.383  1.703156e+09  30
848  1.703156e+09   9.460  1.703156e+09  13.062  1.703156e+09  29

     MX_timestamp  MY  MY_timestamp  MZ  MZ_timestamp  Subject_no
0    1.702897e+09   6  1.702897e+09 -43  1.702897e+09           0
1    1.702897e+09   5  1.702897e+09 -43  1.702897e+09           0
2    1.702897e+09   8  1.702897e+09 -43  1.702897e+09           0
3    1.702897e+09   7  1.702897e+09 -42  1.702897e+09           0
4    1.702897e+09   7  1.702897e+09 -42  1.702897e+09           0
..            ...  ..           ...  ..           ...         ...
844  1.703156e+09  21  1.703156e+09  14  1.703156e+09          35
845  1.703156e+09  18  1.703156e+09  14  1.703156e+09          35
846  1.703156e+09  19  1.703156e+09  14  1.703156e+09          35
847  1.703156e+09  21  1.703156e+09  14  1.703156e+09          35
848  1.703156e+09  20  1.703156e+09  14  1.703156e+09          35

[41260 rows x 19 columns]
```

```
df_combined['Subject_no'] = df_combined['Subject_no'].astype(int)
```

```
df_combined['AX_timestamp'].dtype
```

```
dtype('float64')
```

```
df_combined = df_combined.sort_values(by=['Subject_no', 'AX_timestamp'], ascending=[True, True])
```

```
df_combined

        AX  AX_timestamp     AY  AY_timestamp     AZ  AZ_timestamp      GX  \
0   -0.039  1.702897e+09  0.962  1.702897e+09 -0.254  1.702897e+09   3.174
1   -0.033  1.702897e+09  0.953  1.702897e+09 -0.248  1.702897e+09   3.357
2   -0.046  1.702897e+09  0.956  1.702897e+09 -0.244  1.702897e+09   3.052
3   -0.073  1.702897e+09  0.967  1.702897e+09 -0.245  1.702897e+09   2.563
4   -0.075  1.702897e+09  0.963  1.702897e+09 -0.250  1.702897e+09   1.892
..     ...           ...    ...           ...    ...           ...     ...
844 -0.189  1.703156e+09  0.072  1.703156e+09  0.999  1.703156e+09  13.672
845 -0.255  1.703156e+09  0.031  1.703156e+09  1.004  1.703156e+09 -17.303
846 -0.244  1.703156e+09  0.052  1.703156e+09  1.010  1.703156e+09 -23.804
847 -0.274  1.703156e+09  0.077  1.703156e+09  1.010  1.703156e+09   0.366
848 -0.254  1.703156e+09  0.031  1.703156e+09  1.010  1.703156e+09   6.989

     GX_timestamp      GY  GY_timestamp      GZ  GZ_timestamp  MX  \
0    1.702897e+09  -8.148  1.702897e+09  -2.899  1.702897e+09  72
1    1.702897e+09  -6.927  1.702897e+09  -3.479  1.702897e+09  72
2    1.702897e+09  -1.678  1.702897e+09  -3.784  1.702897e+09  72
3    1.702897e+09   2.228  1.702897e+09  -2.960  1.702897e+09  72
4    1.702897e+09   2.563  1.702897e+09  -2.319  1.702897e+09  72
..            ...     ...           ...     ...           ...  ..
844  1.703156e+09  37.079  1.703156e+09 -21.240  1.703156e+09  30
845  1.703156e+09  29.327  1.703156e+09  -9.979  1.703156e+09  31
846  1.703156e+09 -11.383  1.703156e+09   3.906  1.703156e+09  29
847  1.703156e+09 -21.576  1.703156e+09  11.383  1.703156e+09  30
848  1.703156e+09   9.460  1.703156e+09  13.062  1.703156e+09  29

     MX_timestamp  MY  MY_timestamp  MZ  MZ_timestamp  Subject_no
0    1.702897e+09   6  1.702897e+09 -43  1.702897e+09           0
1    1.702897e+09   5  1.702897e+09 -43  1.702897e+09           0
2    1.702897e+09   8  1.702897e+09 -43  1.702897e+09           0
3    1.702897e+09   7  1.702897e+09 -42  1.702897e+09           0
4    1.702897e+09   7  1.702897e+09 -42  1.702897e+09           0
..            ...  ..           ...  ..           ...         ...
```

```
844  1.703156e+09  21  1.703156e+09  14  1.703156e+09          35
845  1.703156e+09  18  1.703156e+09  14  1.703156e+09          35
846  1.703156e+09  19  1.703156e+09  14  1.703156e+09          35
847  1.703156e+09  21  1.703156e+09  14  1.703156e+09          35
848  1.703156e+09  20  1.703156e+09  14  1.703156e+09          35

[41260 rows x 19 columns]
```

```
list(df_combined.columns)
```

```
['AX',
 'AX_timestamp',
 'AY',
 'AY_timestamp',
 'AZ',
 'AZ_timestamp',
 'GX',
 'GX_timestamp',
 'GY',
 'GY_timestamp',
 'GZ',
 'GZ_timestamp',
 'MX',
 'MX_timestamp',
 'MY',
 'MY_timestamp',
 'MZ',
 'MZ_timestamp',
 'Subject_no']
```

```
df = df_combined.copy()
```

```
df['Timestamp'] = df['AX_timestamp']
```

```
df = df [

    ['Subject_no',
```

```
    'Timestamp',
    'AX',
 'AY',
 'AZ',
 'GX',
 'GY',
 'GZ',
 'MX',
 'MY',
 'MZ',
 ]


]
df
```

```
   Subject_no     Timestamp     AX     AY     AZ      GX      GY      GZ  \
0           0  1.702897e+09 -0.039  0.962 -0.254   3.174  -8.148  -2.899
1           0  1.702897e+09 -0.033  0.953 -0.248   3.357  -6.927  -3.479
2           0  1.702897e+09 -0.046  0.956 -0.244   3.052  -1.678  -3.784
3           0  1.702897e+09 -0.073  0.967 -0.245   2.563   2.228  -2.960
4           0  1.702897e+09 -0.075  0.963 -0.250   1.892   2.563  -2.319
```

```
..          ...           ...    ...    ...    ...     ...     ...     ...
844          35  1.703156e+09 -0.189  0.072  0.999  13.672  37.079 -21.240
845          35  1.703156e+09 -0.255  0.031  1.004 -17.303  29.327  -9.979
846          35  1.703156e+09 -0.244  0.052  1.010 -23.804 -11.383   3.906
847          35  1.703156e+09 -0.274  0.077  1.010   0.366 -21.576  11.383
848          35  1.703156e+09 -0.254  0.031  1.010   6.989   9.460  13.062

     MX  MY  MZ
0    72   6 -43
1    72   5 -43
2    72   8 -43
3    72   7 -42
4    72   7 -42
..   ..  ..  ..
844  30  21  14
845  31  18  14
846  29  19  14
847  30  21  14
848  29  20  14

[41260 rows x 11 columns]
```

The Index resets for every subject. Pandas will however add a new global index when you save the file as .csv

```
df.to_csv('UIA_IMU_9ax_WG_Dataset_W_calibration_data.csv')
```

Below is an addendum to UIA IMU data wrangling. The creation of a dt column for use in attitude calculation (Quaternion and Euler angle expressions for IMU attitude) is added. The resulting dataframe has an error in it due to the fact that the python notebook ran twice, hence one of the columns in the dataset is duplicated.

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

```
# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_ID_Walking_Gait_Dataset_8_1_2024
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_IMU_9ax_WG_Dataset_W_Calibration_Data_U_21_Des_23.csv
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_IMU_9ax_WG_Dataset_U_19_Des_23
```

```
df = pd.read_csv('/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_IMU_9ax_WG_Dataset_U_19_Des_23')

pd.set_option("display.max_rows", 250)
pd.options.display.max_columns = 30

pd.options.display.max_colwidth = 100

df['Time_'] = df['Timestamp']

df
```

	Unnamed: 0	Sub_Index	Timestamp	dt	Subject_no	AX	AY	\
0	0	0	0.000000	NaN	0	-0.039	0.962	
1	1	1	0.000000	0.000000	0	-0.033	0.953	
2	2	2	0.000000	0.000000	0	-0.046	0.956	
3	3	3	0.000000	0.000000	0	-0.073	0.967	
4	4	4	0.034056	0.034056	0	-0.075	0.963	
...	...	...	...	...	...	...	...	
38832	38832	3313	132.178021	0.039903	12	0.108	0.880	
38833	38833	3314	132.217924	0.039903	12	0.107	0.849	
38834	38834	3315	132.257827	0.039903	12	0.138	0.812	

```
38835       38835       3316  132.297729  0.039902          12 -0.170  0.953
38836       38836       3317  132.337632  0.039903          12 -0.529  1.102

          AZ      GX       GY      GZ  MX  MY  MZ       Time_
0     -0.254   3.174   -8.148  -2.899  72   6 -43    0.000000
1     -0.248   3.357   -6.927  -3.479  72   5 -43    0.000000
2     -0.244   3.052   -1.678  -3.784  72   8 -43    0.000000
3     -0.245   2.563    2.228  -2.960  72   7 -42    0.000000
4     -0.250   1.892    2.563  -2.319  72   7 -42    0.034056
...      ...     ...      ...     ...  ..  ..  ..         ...
38832  0.072   2.777  218.170  -3.876  56  41 -27  132.178021
38833  0.062  -3.235  181.396   8.179  55  40 -27  132.217924
38834  0.022  32.562  104.889  20.264  55  41 -28  132.257827
38835 -0.428  49.622   88.623  23.132  56  39 -30  132.297729
38836  0.341  84.442   87.830   5.646  56  36 -31  132.337632

[38837 rows x 15 columns]
```


Want to set the time so that it counts from zero for every subject.

```
df['Time_'] = df.groupby('Subject_no')['Timestamp'].transform(lambda x: x - x.min())
```

```
df
```

```
       Unnamed: 0  Sub_Index   Timestamp        dt  Subject_no     AX     AY  \
0               0          0    0.000000       NaN           0 -0.039  0.962
1               1          1    0.000000  0.000000           0 -0.033  0.953
2               2          2    0.000000  0.000000           0 -0.046  0.956
3               3          3    0.000000  0.000000           0 -0.073  0.967
4               4          4    0.034056  0.034056           0 -0.075  0.963
...           ...        ...         ...       ...         ...    ...    ...
38832       38832       3313  132.178021  0.039903          12  0.108  0.880
38833       38833       3314  132.217924  0.039903          12  0.107  0.849
38834       38834       3315  132.257827  0.039903          12  0.138  0.812
38835       38835       3316  132.297729  0.039902          12 -0.170  0.953
38836       38836       3317  132.337632  0.039903          12 -0.529  1.102
```

```
          AZ      GX       GY      GZ  MX  MY  MZ       Time_
0     -0.254   3.174   -8.148  -2.899  72   6 -43    0.000000
1     -0.248   3.357   -6.927  -3.479  72   5 -43    0.000000
2     -0.244   3.052   -1.678  -3.784  72   8 -43    0.000000
3     -0.245   2.563    2.228  -2.960  72   7 -42    0.000000
4     -0.250   1.892    2.563  -2.319  72   7 -42    0.034056
...      ...     ...      ...     ...  ..  ..  ..         ...
38832  0.072   2.777  218.170  -3.876  56  41 -27  132.178021
38833  0.062  -3.235  181.396   8.179  55  40 -27  132.217924
38834  0.022  32.562  104.889  20.264  55  41 -28  132.257827
38835 -0.428  49.622   88.623  23.132  56  39 -30  132.297729
38836  0.341  84.442   87.830   5.646  56  36 -31  132.337632

[38837 rows x 15 columns]
```

```
df['dt'] = df.groupby('Subject_no')['Time_'].diff()

df = df.rename(columns={"Unnamed: 0" : "Sub_Index"})

display(df)
```

```
       Sub_Index  Sub_Index   Timestamp        dt  Subject_no     AX     AY  \
0              0          0    0.000000       NaN           0 -0.039  0.962
1              1          1    0.000000  0.000000           0 -0.033  0.953
2              2          2    0.000000  0.000000           0 -0.046  0.956
3              3          3    0.000000  0.000000           0 -0.073  0.967
4              4          4    0.034056  0.034056           0 -0.075  0.963
...          ...        ...         ...       ...         ...    ...    ...
38832      38832       3313  132.178021  0.039903          12  0.108  0.880
38833      38833       3314  132.217924  0.039903          12  0.107  0.849
38834      38834       3315  132.257827  0.039903          12  0.138  0.812
38835      38835       3316  132.297729  0.039902          12 -0.170  0.953
38836      38836       3317  132.337632  0.039903          12 -0.529  1.102

          AZ      GX       GY      GZ  MX  MY  MZ       Time_
```

```
0     -0.254   3.174   -8.148  -2.899  72   6 -43    0.000000
1     -0.248   3.357   -6.927  -3.479  72   5 -43    0.000000
2     -0.244   3.052   -1.678  -3.784  72   8 -43    0.000000
3     -0.245   2.563    2.228  -2.960  72   7 -42    0.000000
4     -0.250   1.892    2.563  -2.319  72   7 -42    0.034056
...      ...     ...      ...     ...  ..  ..  ..         ...
38832  0.072   2.777  218.170  -3.876  56  41 -27  132.178021
38833  0.062  -3.235  181.396   8.179  55  40 -27  132.217924
38834  0.022  32.562  104.889  20.264  55  41 -28  132.257827
38835 -0.428  49.622   88.623  23.132  56  39 -30  132.297729
38836  0.341  84.442   87.830   5.646  56  36 -31  132.337632

[38837 rows x 15 columns]
```

```
df.to_csv('Checking.csv')
```

df looks good. Replace timestamp column with time_

```
list(df.columns)
```

```
['Sub_Index',
 'Sub_Index',
 'Timestamp',
 'dt',
 'Subject_no',
 'AX',
 'AY',
 'AZ',
 'GX',
 'GY',
 'GZ',
 'MX',
 'MY',
 'MZ',
 'Time_']
```

```
df = df[[
```

```
    'Sub_Index',
 'Subject_no',
 'AX',
 'AY',
 'AZ',
 'GX',
 'GY',
 'GZ',
 'MX',
 'MY',
 'MZ',
 'Time_',
 'dt'
]]

df['Timestamp'] = df['Time_']

df = df[[
    'Sub_Index',
    'Timestamp',
    'dt',
 'Subject_no',
 'AX',
 'AY',
 'AZ',
 'GX',
 'GY',
 'GZ',
 'MX',
 'MY',
 'MZ',


]]
```

```
df
```

```
       Sub_Index  Sub_Index   Timestamp        dt  Subject_no     AX     AY  \
0              0          0    0.000000       NaN           0 -0.039  0.962
1              1          1    0.000000  0.000000           0 -0.033  0.953
2              2          2    0.000000  0.000000           0 -0.046  0.956
3              3          3    0.000000  0.000000           0 -0.073  0.967
4              4          4    0.034056  0.034056           0 -0.075  0.963
...          ...        ...         ...       ...         ...    ...    ...
38832      38832       3313  132.178021  0.039903          12  0.108  0.880
38833      38833       3314  132.217924  0.039903          12  0.107  0.849
38834      38834       3315  132.257827  0.039903          12  0.138  0.812
38835      38835       3316  132.297729  0.039902          12 -0.170  0.953
38836      38836       3317  132.337632  0.039903          12 -0.529  1.102

          AZ      GX       GY      GZ  MX  MY  MZ
0     -0.254   3.174   -8.148  -2.899  72   6 -43
1     -0.248   3.357   -6.927  -3.479  72   5 -43
2     -0.244   3.052   -1.678  -3.784  72   8 -43
3     -0.245   2.563    2.228  -2.960  72   7 -42
4     -0.250   1.892    2.563  -2.319  72   7 -42
...      ...     ...      ...     ...  ..  ..  ..
38832  0.072   2.777  218.170  -3.876  56  41 -27
38833  0.062  -3.235  181.396   8.179  55  40 -27
38834  0.022  32.562  104.889  20.264  55  41 -28
38835 -0.428  49.622   88.623  23.132  56  39 -30
38836  0.341  84.442   87.830   5.646  56  36 -31

[38837 rows x 14 columns]
```

```
df.to_csv('UIA_IMU_9ax_WG_Dataset_W_Calibration_Data_U_21_Des_23')
```

```
df
```

```
       Sub_Index  Sub_Index   Timestamp        dt  Subject_no     AX     AY  \
```

0	0	0	0.000000	NaN	0	-0.039	0.962	
1	1	1	0.000000	0.000000	0	-0.033	0.953	
2	2	2	0.000000	0.000000	0	-0.046	0.956	
3	3	3	0.000000	0.000000	0	-0.073	0.967	
4	4	4	0.034056	0.034056	0	-0.075	0.963	
...	...	...	...	...	...	...	...	
38832	38832	3313	132.178021	0.039903	12	0.108	0.880	
38833	38833	3314	132.217924	0.039903	12	0.107	0.849	
38834	38834	3315	132.257827	0.039903	12	0.138	0.812	
38835	38835	3316	132.297729	0.039902	12	-0.170	0.953	
38836	38836	3317	132.337632	0.039903	12	-0.529	1.102	

	AZ	GX	GY	GZ	MX	MY	MZ
0	-0.254	3.174	-8.148	-2.899	72	6	-43
1	-0.248	3.357	-6.927	-3.479	72	5	-43
2	-0.244	3.052	-1.678	-3.784	72	8	-43
3	-0.245	2.563	2.228	-2.960	72	7	-42
4	-0.250	1.892	2.563	-2.319	72	7	-42
...	...	...	...	...	..	..	..
38832	0.072	2.777	218.170	-3.876	56	41	-27
38833	0.062	-3.235	181.396	8.179	55	40	-27
38834	0.022	32.562	104.889	20.264	55	41	-28
38835	-0.428	49.622	88.623	23.132	56	39	-30
38836	0.341	84.442	87.830	5.646	56	36	-31

[38837 rows x 14 columns]

A.2 UIA IMU: Quaternion and Euler angle generation using Madwick filter

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import math

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a ...
   version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session

/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_ID_Walking_Gait_Dataset_8_1_2024
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_IMU_9ax_WG_Dataset_W_Calibration_Data_U_21_Des_23.csv
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_IMU_9ax_WG_Dataset_U_19_Des_23

df = ...
   pd.read_csv('/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_IMU_9ax_WG_Dataset_W_Calibration_Data_U_21_Des_23.csv')
```

```
df
```

	Unnamed: 0	Sub_Index	Timestamp	dt	Subject_no	AX	AY	\
0	0	0	0.000000	NaN	0	-0.039	0.962	
1	1	1	0.000000	0.000000	0	-0.033	0.953	
2	2	2	0.000000	0.000000	0	-0.046	0.956	
3	3	3	0.000000	0.000000	0	-0.073	0.967	
4	4	4	0.034056	0.034056	0	-0.075	0.963	
...	...	...	...	...	...	...	...	

```
41255       41255        844  33.430305  0.039171          35 -0.189  0.072
41256       41256        845  33.469476  0.039171          35 -0.255  0.031
41257       41257        846  33.508647  0.039171          35 -0.244  0.052
41258       41258        847  33.549364  0.040717          35 -0.274  0.077
41259       41259        848  33.590081  0.040717          35 -0.254  0.031

          AZ      GX      GY      GZ  MX  MY  MZ
0     -0.254   3.174  -8.148  -2.899  72   6 -43
1     -0.248   3.357  -6.927  -3.479  72   5 -43
2     -0.244   3.052  -1.678  -3.784  72   8 -43
3     -0.245   2.563   2.228  -2.960  72   7 -42
4     -0.250   1.892   2.563  -2.319  72   7 -42
...      ...     ...     ...     ...  ..  ..  ..
41255  0.999  13.672  37.079 -21.240  30  21  14
41256  1.004 -17.303  29.327  -9.979  31  18  14
41257  1.010 -23.804 -11.383   3.906  29  19  14
41258  1.010   0.366 -21.576  11.383  30  21  14
41259  1.010   6.989   9.460  13.062  29  20  14

[41260 rows x 14 columns]
```

```
df_1 = df.copy()
```

remove rows where dt is 0 or negative. Negative dt corresponds to transition between different subjects.

```
#filtered dataframe

df_1 = df_1[df_1['dt'] > 0]
df_1 = df_1.reset_index()

df_1['q0'] = ''
df_1['q1'] = ''
df_1['q2'] = ''
df_1['q3'] = ''
```

```
df_1['Pitch'] = ''
df_1['Roll'] = ''
df_1['Yaw'] = ''
```

The following orientation is observed throgh experimentation with the Emotibit sensor

Acc:

Y: Up +

X: FWD +

Z: R +

Gyr:

YAW = Y CCW +

PITCH = Z CCW +

ROLL = X CCW - (invert)

```
df_1['accX'] = df_1['AX'].copy()
df_1['accY'] = df_1['AZ'].copy()
df_1['accZ'] = -df_1['AY'].copy()+1


df_1['gyrX'] = -df_1['GX'].copy()*(math.pi/180)
df_1['gyrY'] = df_1['GZ'].copy()*(math.pi/180)
df_1['gyrZ'] = df_1['GY'].copy()*(math.pi/180)

df_1['magX'] = df_1['MX'].copy()
df_1['magY'] = df_1['MZ'].copy()
df_1['magZ'] = df_1['MY'].copy()

df_1
```

```
       index  Unnamed: 0  Sub_Index  Timestamp        dt  Subject_no     AX  \
0          4           4          4   0.034056  0.034056           0 -0.075
1          5           5          5   0.068112  0.034056           0 -0.082
2          6           6          6   0.102168  0.034056           0 -0.098
```

```
3          7           7          7   0.142416  0.040248           0 -0.099
4          8           8          8   0.182664  0.040248           0 -0.092
...      ...         ...        ...        ...       ...         ...    ...
41157  41255       41255        844  33.430305  0.039171          35 -0.189
41158  41256       41256        845  33.469476  0.039171          35 -0.255
41159  41257       41257        846  33.508647  0.039171          35 -0.244
41160  41258       41258        847  33.549364  0.040717          35 -0.274
41161  41259       41259        848  33.590081  0.040717          35 -0.254

          AY     AZ      GX  ...  Yaw   accX   accY   accZ      gyrX  \
0      0.963 -0.250   1.892  ...      -0.075 -0.250  0.037 -0.033022
1      0.964 -0.266   1.465  ...      -0.082 -0.266  0.036 -0.025569
2      0.967 -0.251   2.136  ...      -0.098 -0.251  0.033 -0.037280
3      0.960 -0.248   1.831  ...      -0.099 -0.248  0.040 -0.031957
4      0.965 -0.250   1.190  ...      -0.092 -0.250  0.035 -0.020769
...      ...    ...     ...  ...  ...    ...    ...    ...       ...
41157  0.072  0.999  13.672  ...      -0.189  0.999  0.928 -0.238621
41158  0.031  1.004 -17.303  ...      -0.255  1.004  0.969  0.301994
41159  0.052  1.010 -23.804  ...      -0.244  1.010  0.948  0.415458
41160  0.077  1.010   0.366  ...      -0.274  1.010  0.923 -0.006388
41161  0.031  1.010   6.989  ...      -0.254  1.010  0.969 -0.121981

           gyrY      gyrZ magX magY magZ
0     -0.040474  0.044733   72  -42    7
1     -0.044733  0.036757   73  -43    7
2     -0.034627  0.069237   72  -43    8
3     -0.026093  0.079901   72  -43    7
4     -0.020246  0.029287   72  -43    5
...         ...       ...  ...  ...  ...
41157 -0.370708  0.647151   30   14   21
41158 -0.174166  0.511853   31   14   18
41159  0.068173 -0.198671   29   14   19
41160  0.198671 -0.376572   30   14   21
41161  0.227975  0.165108   29   14   20
```

```
[41162 rows x 31 columns]
```

```
list(df_1.columns)
```

```
['index',
 'Unnamed: 0',
 'Sub_Index',
 'Timestamp',
 'dt',
 'Subject_no',
 'AX',
 'AY',
 'AZ',
 'GX',
 'GY',
 'GZ',
 'MX',
 'MY',
 'MZ',
 'q0',
 'q1',
 'q2',
 'q3',
 'Pitch',
 'Roll',
 'Yaw',
 'accX',
 'accY',
 'accZ',
 'gyrX',
 'gyrY',
 'gyrZ',
 'magX',
 'magY',
```

```
 'magZ']

df_1 = df_1[[
 'Sub_Index',
 'Timestamp',
 'dt',
 'Subject_no',
     'q0',
 'q1',
 'q2',
 'q3',
 'Pitch',
 'Roll',
 'Yaw',
 'accX',
 'accY',
 'accZ',
 'gyrX',
 'gyrY',
 'gyrZ',
 'magX',
 'magY',
 'magZ'
]]

df_1
```

```
       Sub_Index  Timestamp        dt  Subject_no q0 q1 q2 q3 Pitch Roll Yaw  \
0              4   0.034056  0.034056           0
1              5   0.068112  0.034056           0
2              6   0.102168  0.034056           0
3              7   0.142416  0.040248           0
4              8   0.182664  0.040248           0
...          ...        ...       ...         ... .. .. .. ..   ...  ...  ..
41157        844  33.430305  0.039171          35
```

```
41158        845  33.469476  0.039171          35
41159        846  33.508647  0.039171          35
41160        847  33.549364  0.040717          35
41161        848  33.590081  0.040717          35

        accX   accY   accZ      gyrX      gyrY      gyrZ  magX  magY  magZ
0     -0.075 -0.250  0.037 -0.033022 -0.040474  0.044733    72   -42     7
1     -0.082 -0.266  0.036 -0.025569 -0.044733  0.036757    73   -43     7
2     -0.098 -0.251  0.033 -0.037280 -0.034627  0.069237    72   -43     8
3     -0.099 -0.248  0.040 -0.031957 -0.026093  0.079901    72   -43     7
4     -0.092 -0.250  0.035 -0.020769 -0.020246  0.029287    72   -43     5
...      ...    ...    ...       ...       ...       ...   ...   ...   ...
41157 -0.189  0.999  0.928 -0.238621 -0.370708  0.647151    30    14    21
41158 -0.255  1.004  0.969  0.301994 -0.174166  0.511853    31    14    18
41159 -0.244  1.010  0.948  0.415458  0.068173 -0.198671    29    14    19
41160 -0.274  1.010  0.923 -0.006388  0.198671 -0.376572    30    14    21
41161 -0.254  1.010  0.969 -0.121981  0.227975  0.165108    29    14    20

[41162 rows x 20 columns]

features_ = [


 'accX',
 'accY',
 'accZ',
 'gyrX',
 'gyrY',
 'gyrZ'


]

from scipy.signal import butter, filtfilt
```

```
# Define the butterworth bandpass filter
def butter_bandpass(lowcut, highcut, fs, order=2):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b, a = butter(order, [low, high], btype='band')
    return b, a

# Apply the filter to the signal
def butter_bandpass_filter(data, lowcut, highcut, fs, order=2):
    b, a = butter_bandpass(lowcut, highcut, fs, order=order)
    y = filtfilt(b, a, data)
    return y

lowcut = 0.85
highcut = 4
fs = 28
```



```
for feature in features_ :

    df_1[feature] = butter_bandpass_filter(df_1[feature], lowcut, highcut, fs, order=10)
```

```
/tmp/ipykernel_42/2013497234.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_1[feature] = butter_bandpass_filter(df_1[feature], lowcut, highcut, fs, order=10)
```

```
df.to_csv('Check_df')
```

```
df_1.isna().sum()
```

```
Sub_Index      0
Timestamp      0
dt             0
Subject_no     0
q0             0
q1             0
q2             0
q3             0
Pitch          0
Roll           0
Yaw            0
accX           0
accY           0
accZ           0
gyrX           0
gyrY           0
gyrZ           0
magX           0
magY           0
magZ           0
dtype: int64
```

```
df_1
```

	Sub_Index	Timestamp	dt	Subject_no	q0	q1	q2	q3	Pitch	Roll	Yaw	\
0	4	0.034056	0.034056	0								
1	5	0.068112	0.034056	0								
2	6	0.102168	0.034056	0								
3	7	0.142416	0.040248	0								
4	8	0.182664	0.040248	0								
...	...	...	...	...	..	..	..	..	...	...	..	
41157	844	33.430305	0.039171	35								
41158	845	33.469476	0.039171	35								
41159	846	33.508647	0.039171	35								

```
41160        847  33.549364  0.040717          35
41161        848  33.590081  0.040717          35

          accX      accY      accZ      gyrX      gyrY      gyrZ  magX  magY  \
0     0.001269 -0.007415  0.007225 -0.015511 -0.128565 -0.013460    72   -42
1    -0.014983 -0.010159  0.003744  0.009013 -0.156497 -0.001248    73   -43
2    -0.024849 -0.013850 -0.001352  0.032911 -0.179508 -0.004759    72   -43
3    -0.025351 -0.018286 -0.007889  0.054869 -0.196698 -0.028749    72   -43
4    -0.018526 -0.021769 -0.014033  0.074107 -0.206459 -0.063698    72   -43
...        ...       ...       ...       ...       ...       ...   ...   ...
41157 -0.051769 -0.032006 -0.108341  0.506630 -0.877110  0.279173    30    14
41158 -0.049827 -0.041471 -0.096799  0.624267 -0.785783  0.069264    31    14
41159 -0.042838 -0.044227 -0.078125  0.600694 -0.578050 -0.103156    29    14
41160 -0.030025 -0.040083 -0.053133  0.409202 -0.288411 -0.176892    30    14
41161 -0.013343 -0.031016 -0.024143  0.101512  0.041359 -0.167326    29    14

       magZ
0         7
1         7
2         8
3         7
4         5
...     ...
41157    21
41158    18
41159    19
41160    21
41161    20

[41162 rows x 20 columns]
```

```
import matplotlib.pyplot as plt
```

Calibrate based on virtual subjects 33, 34, 35

Data is recorded while doing the following sequence:

1-10 seconds : Idle (For bias and drift)

10 sec : device is rotated 90 deg in counter-clock dir around the Y-axis

20 sec : device is rotated back to neutral

```
subjects = [33]

df_cal = df_1[df_1['Subject_no'].isin(subjects)]

df_cal = df_cal[(0 < df_cal['Timestamp']) & (df_cal['Timestamp'] < 500)]

plt.figure(figsize=(15, 5))

for subject in subjects:

    Cal_data = df_cal[df_cal['Subject_no'] == subject]
    #plt.plot(Cal_data['Timestamp'], Cal_data['accZ'])
    plt.plot(Cal_data['Timestamp'], Cal_data['gyrY'])

plt.title('Gyroscope Data (gyrY) for calibration')
plt.xlabel('Timestamp (seconds)')
plt.ylabel('Gyroscope Reading (degrees/second)')
plt.legend()
plt.grid(True)
plt.show()
```

As this data is generated using movements with other frequency components than the actual gait data, displaying this plot after filtering for gait components is not considered useful.

```
df_1
```

```
       Sub_Index  Timestamp        dt  Subject_no q0 q1 q2 q3 Pitch Roll Yaw  \
0              4   0.034056  0.034056           0
1              5   0.068112  0.034056           0
2              6   0.102168  0.034056           0
```

```
3              7   0.142416  0.040248           0
4              8   0.182664  0.040248           0
...          ...        ...       ...         ... .. .. .. ..   ...  ...  ..
41157        844  33.430305  0.039171          35
41158        845  33.469476  0.039171          35
41159        846  33.508647  0.039171          35
41160        847  33.549364  0.040717          35
41161        848  33.590081  0.040717          35

           accX      accY      accZ      gyrX      gyrY      gyrZ  magX  magY  \
0      0.001269 -0.007415  0.007225 -0.015511 -0.128565 -0.013460    72   -42
1     -0.014983 -0.010159  0.003744  0.009013 -0.156497 -0.001248    73   -43
2     -0.024849 -0.013850 -0.001352  0.032911 -0.179508 -0.004759    72   -43
3     -0.025351 -0.018286 -0.007889  0.054869 -0.196698 -0.028749    72   -43
4     -0.018526 -0.021769 -0.014033  0.074107 -0.206459 -0.063698    72   -43
...         ...       ...       ...       ...       ...       ...   ...   ...
41157 -0.051769 -0.032006 -0.108341  0.506630 -0.877110  0.279173    30    14
41158 -0.049827 -0.041471 -0.096799  0.624267 -0.785783  0.069264    31    14
41159 -0.042838 -0.044227 -0.078125  0.600694 -0.578050 -0.103156    29    14
41160 -0.030025 -0.040083 -0.053133  0.409202 -0.288411 -0.176892    30    14
41161 -0.013343 -0.031016 -0.024143  0.101512  0.041359 -0.167326    29    14

       magZ
0         7
1         7
2         8
3         7
4         5
...     ...
41157    21
41158    18
41159    19
41160    21
41161    20
```

```
[41162 rows x 20 columns]
'''

Madwick filter for generating quaternion expression for IMU attitude

'''


# Need to find constant for Gyro measurement error
pi = np.pi

BMI_Sensitivity_Rfs150 = 131.2 #LSB per deg/s, from BMI160 datasheet

BMI_Sens = BMI_Sensitivity_Rfs150 *(pi/180) #convert ~ 2.29 rad/s

Desired_Accuracy_Deg = 2.5
Desired_Accuracy = Desired_Accuracy_Deg * (pi/180) #convert ~ 0.0436 rad/s

Gyro_Meas_Error = BMI_Sens * Desired_Accuracy # 2.29 * 0.0436 ~ 0.099 r/s

GME = Gyro_Meas_Error #shorts are nice



'''
Global

'''

acc_X, acc_y, acc_Z = 0.0, 0.0, 0.0

gyr_X, gyr_Y, gyr_Z = 0.0, 0.0, 0.0
```

```
mag_X, mag_Y, mag_Z = 0.0, 0.0, 0.0

SE_q0, SE_q1, SE_q2, SE_q3 = 0.0, 0.0, 0.0, 0.0

beta = 1.5 * GME

#beta = 0.001 * GME


'''
Initialize the filter

'''


def Madwick_Init():

    global SE_q0, SE_q1, SE_q2, SE_q3
    SE_q0 = 1.0
    SE_q1 = 0.0
    SE_q2 = 0.0
    SE_q3 = 0.0



#Madwick_Init() # initialized


def Madwick_quat(dt, acc_X, acc_Y, acc_Z, gyr_X, gyr_Y, gyr_Z):

    global SE_q0, SE_q1, SE_q2, SE_q3 # this eliviates the need for return

    # normalize acc and mag measurements
```

```
acc_N = math.sqrt(acc_X**2 + acc_Y**2 + acc_Z**2)

if acc_N != 0:

    acc_X, acc_Y, acc_Z = acc_X / acc_N, acc_Y / acc_N, acc_Z / acc_N

#mag_N = math.sqrt(mag_X**2 + mag_Y**2 + mag_Z**2)
#mag_X, mag_Y, mag_Z = mag_X / mag_N, mag_Y / mag_N, mag_Z / mag_N

# Variables for computational easing

hf_q0 = 0.5 * SE_q0

hf_q1 = 0.5 * SE_q1

hf_q2 = 0.5 * SE_q2

hf_q3 = 0.5 * SE_q3

tw_q0 = 2.0 * SE_q0

tw_q1 = 2.0 * SE_q1

tw_q2 = 2.0 * SE_q2

tw_q3 = 2.0 * SE_q3

fr_q1 = 2.0 * tw_q1

fr_q2 = 2.0 * tw_q2

 # Gradient descent algorithm corrective step
s0 = tw_q1 * SE_q3 - tw_q0 * SE_q2 - acc_X
```

```
s1 = tw_q0 * SE_q1 + tw_q2 * SE_q3 - acc_Y
s2 = 1.0 - tw_q1 * SE_q1 - tw_q2 * SE_q2 - acc_Z # simplified 1.0 - 2.0 * SE_q1 * SE_q1 - 2.0 * SE_q2 * SE_q2 - acc_Z

'''
State Vector:
| s0 |
| s1 |
| s2 |

'''

# Compute the gradient (Jacobians)
J_11or24 = tw_q2  # J_11 negated in matrix multiplication
J_12or23 = tw_q3 # replaced 2.0 * SE_q3 with a placeholder
J_13or22 = tw_q0  # raplaced 2.0 * SE_q0
J_14or21 = tw_q1  # J_13 negated in matrix multiplication
J_32 = fr_q1  # replaced 2.0 * J_14or21 with 2 * 2 * q1
J_33 = fr_q2  # replaced 2 * J_11or24 with 2 * 2 * q2

'''
    Jacobian Matrix:
| J_11or24 J_12or23 J_13or22 J_14or21 |
| 0        0        0        J_32     |
| 0        0        0        J_33     |

'''


'''
J_11or24 = two_q2  # J_11 negated in matrix multiplication
J_12or23 = 2.0 * SE_q3
J_13or22 = two_q0  # J_12 negated in matrix multiplication
J_14or21 = two_q1  # J_13 negated in matrix multiplication
J_32 = 2.0 * J_14or21  # negated in matrix multiplication
```

```
J_33 = 2.0 * J_11or24



'''









# Compute the gradient (matrix multiplication)
SEqHatDot_0 = J_14or21 * s1 - J_11or24 * s0
SEqHatDot_1 = J_12or23 * s0 + J_13or22 * s1 - J_32 * s2
SEqHatDot_2 = J_12or23 * s1 - J_33 * s2 - J_13or22 * s0
SEqHatDot_3 = J_14or21 * s0 + J_11or24 * s1

# Normalize the gradient
norm = math.sqrt(SEqHatDot_0**2 + SEqHatDot_1**2 + SEqHatDot_2**2 + SEqHatDot_3**2)

if norm != 0:

    SEqHatDot_0 /= norm
    SEqHatDot_1 /= norm
    SEqHatDot_2 /= norm
    SEqHatDot_3 /= norm
```

```
'''
 hf_q0 = 0.5 * SE_q0

hf_q1 = 0.5 * SE_q1

hf_q2 = 0.5 * SE_q2

hf_q3 = 0.5 * SE_q3
'''

# Compute the quaternion derivative measured by gyroscopes

SEqDot_omega_0 = -hf_q1 * gyr_X - hf_q2 * gyr_Y - hf_q3 * gyr_Z
#SEqDot_omega_1 = -halfSEq_2 * w_x - halfSEq_3 * w_y - halfSEq_4 * w_z;

SEqDot_omega_1 = hf_q0 * gyr_X + hf_q2 * gyr_Z - hf_q3 * gyr_Y
#SEqDot_omega_2 = halfSEq_1 * w_x + halfSEq_3 * w_z - halfSEq_4 * w_y;


SEqDot_omega_2 = hf_q0 * gyr_Y - hf_q1 * gyr_Z + hf_q3 * gyr_X
#SEqDot_omega_3 = halfSEq_1 * w_y - halfSEq_2 * w_z + halfSEq_4 * w_x;


SEqDot_omega_3 = hf_q0 * gyr_Z + hf_q1 * gyr_Y - hf_q2 * gyr_X
#SEqDot_omega_4 = halfSEq_1 * w_z + halfSEq_2 * w_y - halfSEq_3 * w_x;



# Compute and remove the gyroscope biases
SEqDot_omega_0 -= beta * SEqHatDot_0
SEqDot_omega_1 -= beta * SEqHatDot_1
SEqDot_omega_2 -= beta * SEqHatDot_2
```

```
    SEqDot_omega_3 -= beta * SEqHatDot_3



    # Integrate rate of change of quaternion to yield quaternion
    SE_q0 += SEqDot_omega_0 * dt
    SE_q1 += SEqDot_omega_1 * dt
    SE_q2 += SEqDot_omega_2 * dt
    SE_q3 += SEqDot_omega_3 * dt

    # Normalize quaternion
    norm = math.sqrt(SE_q0**2 + SE_q1**2 + SE_q2**2 + SE_q3**2)

    if norm != 0: # precautionary measure for the case where the sum of the squares is zero
        SE_q0 /= norm
        SE_q1 /= norm
        SE_q2 /= norm
        SE_q3 /= norm
```



```
import math

# df_1 is a DataFrame

'''Variable to keep track of the last subject number = last_subject_no, we re-initialize the filter for every subject,
so that there is no sudden jumps the values used in out calculations '''

last_subject_no = None

# Iterate through each row of the DataFrame
for index, row in df_1.iterrows():
    current_subject_no = row['Subject_no']
```

```
    # Check if the subject number has changed
    if current_subject_no != last_subject_no:

        # If so, reinitialize the filter for the new subject
        Madwick_Init()
        last_subject_no = current_subject_no

    # Extract the necessary values from the row
    dt = row['dt']
    acc_X = row['accX']
    acc_Y = row['accY']
    acc_Z = row['accZ']
    gyr_X = row['gyrX']
    gyr_Y = row['gyrY']
    gyr_Z = row['gyrZ']

    # Apply the Madwick filter
    Madwick_quat(dt, acc_X, acc_Y, acc_Z, gyr_X, gyr_Y, gyr_Z)

    # Store the updated quaternion values in the DataFrame
    df_1.at[index, 'q0'] = SE_q0
    df_1.at[index, 'q1'] = SE_q1
    df_1.at[index, 'q2'] = SE_q2
    df_1.at[index, 'q3'] = SE_q3

'''



'''
import matplotlib.pyplot as plt

# Extract the first 100 rows
```

```
df_100 = df_1.tail(500)

# Plotting Pitch, accZ (AZ), and accX (AX) for the first 100 indices
plt.figure(figsize=(12, 6))

plt.subplot(4, 1, 1)
plt.plot(df_100['Sub_Index'], df_100['q1'], label='q1')
plt.title('q1')
plt.xlabel('Sub_Index')
plt.ylabel('q1 Value')

plt.subplot(4, 1, 2)
plt.plot(df_100['Sub_Index'], df_100['q2'], label='q2', color='green')
plt.title('q2')
plt.xlabel('Sub_Index')
plt.ylabel('q2 Value')

plt.subplot(4, 1, 3)
plt.plot(df_100['Sub_Index'], df_100['q3'], label='q3', color='red')
plt.title('q3')
plt.xlabel('Sub_Index')
plt.ylabel('q3 Value')

plt.subplot(4, 1, 4)
plt.plot(df_100['Sub_Index'], df_100['q0'], label='q0', color='red')
plt.title('q0')
plt.xlabel('Sub_Index')
plt.ylabel('q0 Value')


plt.tight_layout()
plt.show()
```

Not included, as the image does not reveal much in terms of information

```
def euler_from_quaternion(q0, q1, q2, q3):

    """
    Convert a quaternion into euler angles (roll, pitch, yaw)
    roll is rotation around x  (+ counterclockwise)
    pitch is rotation around y  (+ counterclockwise)
    yaw is rotation around z  (+ counterclockwise)

    """

    # Yaw (Z-axis rotation)
    t0 = +2.0 * (q0 * q1 + q2 * q3)
    t1 = +1.0 - 2.0 * (q1 * q1 + q2 * q2)
    yaw_z = math.atan2(t0, t1)

    # Pitch (Y-axis rotation)
    t2 = +2.0 * (q0 * q2 - q3 * q1)
    t2 = +1.0 if t2 > +1.0 else t2  # clamp to avoid NaN in asin
    t2 = -1.0 if t2 < -1.0 else t2
    pitch_y = math.asin(t2)

    # Roll (X-axis rotation)
    t3 = +2.0 * (q0 * q3 + q1 * q2)
    t4 = +1.0 - 2.0 * (q2 * q3 + q3 * q3)
    roll_x = math.atan2(t3, t4)

    roll_x = math.degrees(roll_x)
    pitch_y = math.degrees(pitch_y)
    yaw_z = math.degrees(yaw_z)

    return roll_x, pitch_y, yaw_z  # in radians

def unwrap_angles(yaw_angles):
    unwrapped = [yaw_angles[0]]  # Start with the first angle
```

```
    for i in range(1, len(yaw_angles)):
        delta = yaw_angles[i] - unwrapped[-1]  # Compare to the last unwrapped value
        if delta > 180:
            unwrapped.append(yaw_angles[i] - 360)
        elif delta < -180:
            unwrapped.append(yaw_angles[i] + 360)
        else:
            unwrapped.append(yaw_angles[i])
    return unwrapped


# Apply the ZYX conversion to each row in the dataframe


'''
 No need for subject-spesific execution of conversion, as the underlying quat values
 allready have this implemented. There are therefore no sudden jumps in the underlying values.



last_subject_no = None

# Iterate through each row of the DataFrame
for index, row in df_1.iterrows():
    current_subject_no = row['Subject_no']

    # Check if the subject number has changed
    if current_subject_no != last_subject_no:
        # Reinitialize the filter for the new subject
        Madwick_Init()
        last_subject_no = current_subject_no

'''
```

```
df_1[['Roll', 'Pitch', 'Yaw']] = df_1.apply(lambda row: euler_from_quaternion(row['q0'], row['q1'], row['q2'], row['q3']), axis=1, result_
```

```
# Display the updated dataframe
df_1.head()
```

```
/tmp/ipykernel_42/1768277756.py:71: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_1[['Roll', 'Pitch', 'Yaw']] = df_1.apply(lambda row: euler_from_quaternion(row['q0'], row['q1'], row['q2'], row['q3']), axis=1, resul
```

	Sub_Index	Timestamp	dt	Subject_no	q0	q1	q2	\
0	4	0.034056	0.034056	0	0.999981	-0.005295	-0.00305	
1	5	0.068112	0.034056	0	0.999967	-0.007995	-0.001483	
2	6	0.102168	0.034056	0	0.999951	-0.009922	-0.000083	
3	7	0.142416	0.040248	0	0.999923	-0.012365	0.000831	
4	8	0.182664	0.040248	0	0.999878	-0.015495	0.000539	

	q3	Pitch	Roll	Yaw	accX	accY	accZ	\
0	-0.000229	-0.349617	-0.024413	-0.606688	0.001269	-0.007415	0.007225	
1	-0.000205	-0.170067	-0.022105	-0.916112	-0.014983	-0.010159	0.003744	
2	-0.000221	-0.009767	-0.025253	-1.136978	-0.024849	-0.013850	-0.001352	
3	-0.000712	0.094222	-0.082716	-1.417035	-0.025351	-0.018286	-0.007889	
4	-0.001899	0.058380	-0.218530	-1.775726	-0.018526	-0.021769	-0.014033	

	gyrX	gyrY	gyrZ	magX	magY	magZ
0	-0.015511	-0.128565	-0.013460	72	-42	7
1	0.009013	-0.156497	-0.001248	73	-43	7
2	0.032911	-0.179508	-0.004759	72	-43	8
3	0.054869	-0.196698	-0.028749	72	-43	7

```
4  0.074107 -0.206459 -0.063698    72   -43     5
```

```
df_1['Yaw'] = unwrap_angles(df_1['Yaw'].values).copy()
```

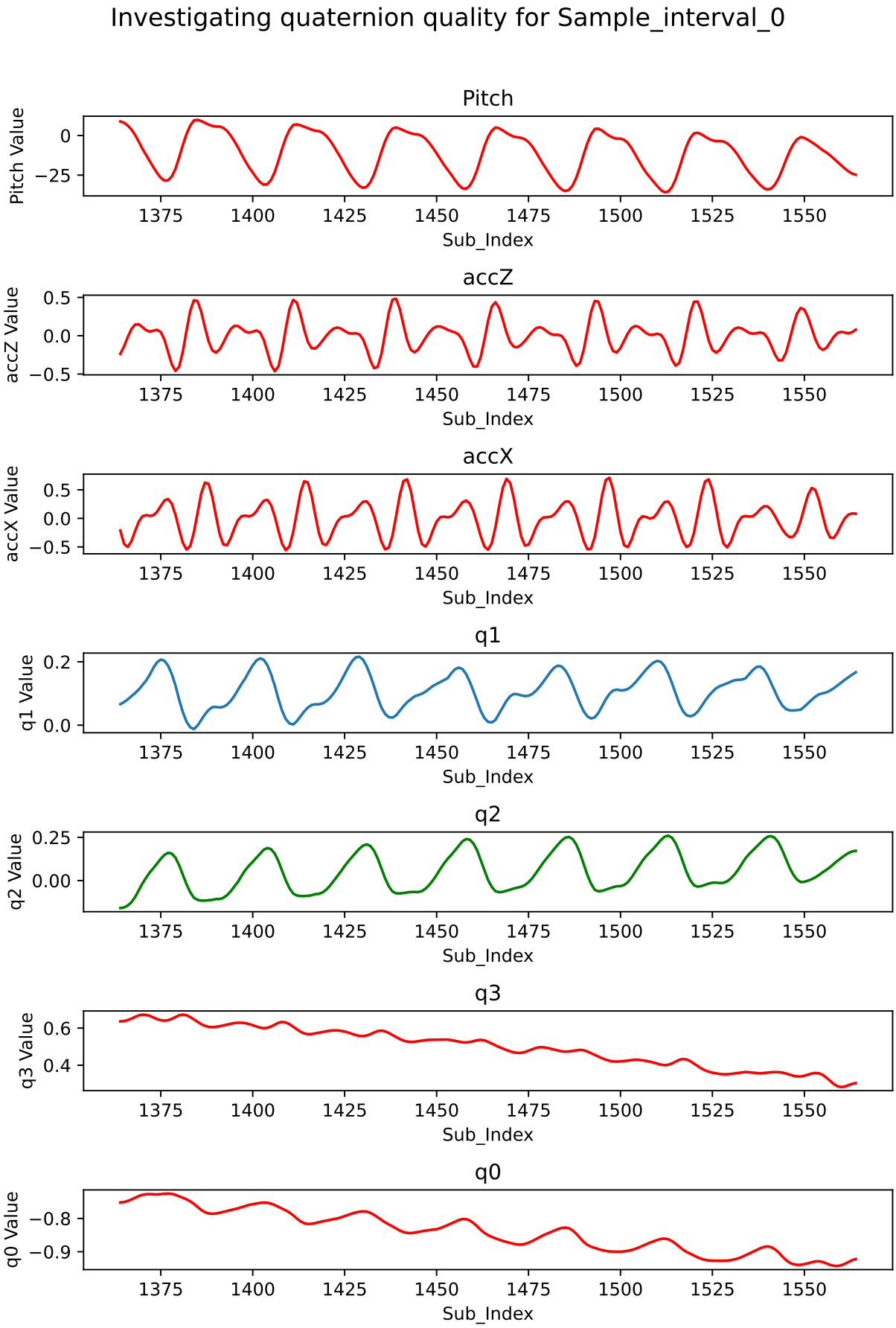
```
/tmp/ipykernel_42/678538172.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_1['Yaw'] = unwrap_angles(df_1['Yaw'].values).copy()
```

```
import matplotlib.pyplot as plt


# Extracting part of df

name = 'Sample_interval_0'

df_subset = df_1.loc[5400:5600] #Random subset 1


plt.figure(figsize=(7.27, 10.69)) # calculated to fit an a4 with aprox 0.5 inch margins


plt.suptitle(f'Investigating quaternion quality for {name}', fontsize=14)




'''


plt.subplot(7, 1, 1)
plt.plot(df_subset['Sub_Index'], df_subset['Roll'], label='Roll', color='red')
plt.title('Roll')
```

```
plt.xlabel('Sub_Index')
plt.ylabel('Roll Value')


'''

plt.subplot(7, 1, 1)
plt.plot(df_subset['Sub_Index'], df_subset['Pitch'], label='Pitch', color='red')
plt.title('Pitch')
plt.xlabel('Sub_Index')
plt.ylabel('Pitch Value')


'''
plt.subplot(7, 1, 1)
plt.plot(df_subset['Sub_Index'], df_subset['Yaw'], label='Yaw', color='red')
plt.title('Yaw')
plt.xlabel('Sub_Index')
plt.ylabel('Yaw Value')
'''


plt.subplot(7, 1, 2)
plt.plot(df_subset['Sub_Index'], df_subset['accZ'], label='accZ', color='red')
plt.title('accZ')
plt.xlabel('Sub_Index')
plt.ylabel('accZ Value')

plt.subplot(7, 1, 3)
plt.plot(df_subset['Sub_Index'], df_subset['accX'], label='accX', color='red')
plt.title('accX')
plt.xlabel('Sub_Index')
plt.ylabel('accX Value')
```

```
plt.subplot(7, 1, 4)
plt.plot(df_subset['Sub_Index'], df_subset['q1'], label='q1')
plt.title('q1')
plt.xlabel('Sub_Index')
plt.ylabel('q1 Value')

plt.subplot(7, 1, 5)
plt.plot(df_subset['Sub_Index'], df_subset['q2'], label='q2', color='green')
plt.title('q2')
plt.xlabel('Sub_Index')
plt.ylabel('q2 Value')

plt.subplot(7, 1, 6)
plt.plot(df_subset['Sub_Index'], df_subset['q3'], label='q3', color='red')
plt.title('q3')
plt.xlabel('Sub_Index')
plt.ylabel('q3 Value')

plt.subplot(7, 1, 7)
plt.plot(df_subset['Sub_Index'], df_subset['q0'], label='q0', color='red')
plt.title('q0')
plt.xlabel('Sub_Index')
plt.ylabel('q0 Value')



plt.tight_layout()
plt.subplots_adjust(top=0.90)


plt.savefig(f'Quaternion_check_{name}.eps', format = 'eps')
```

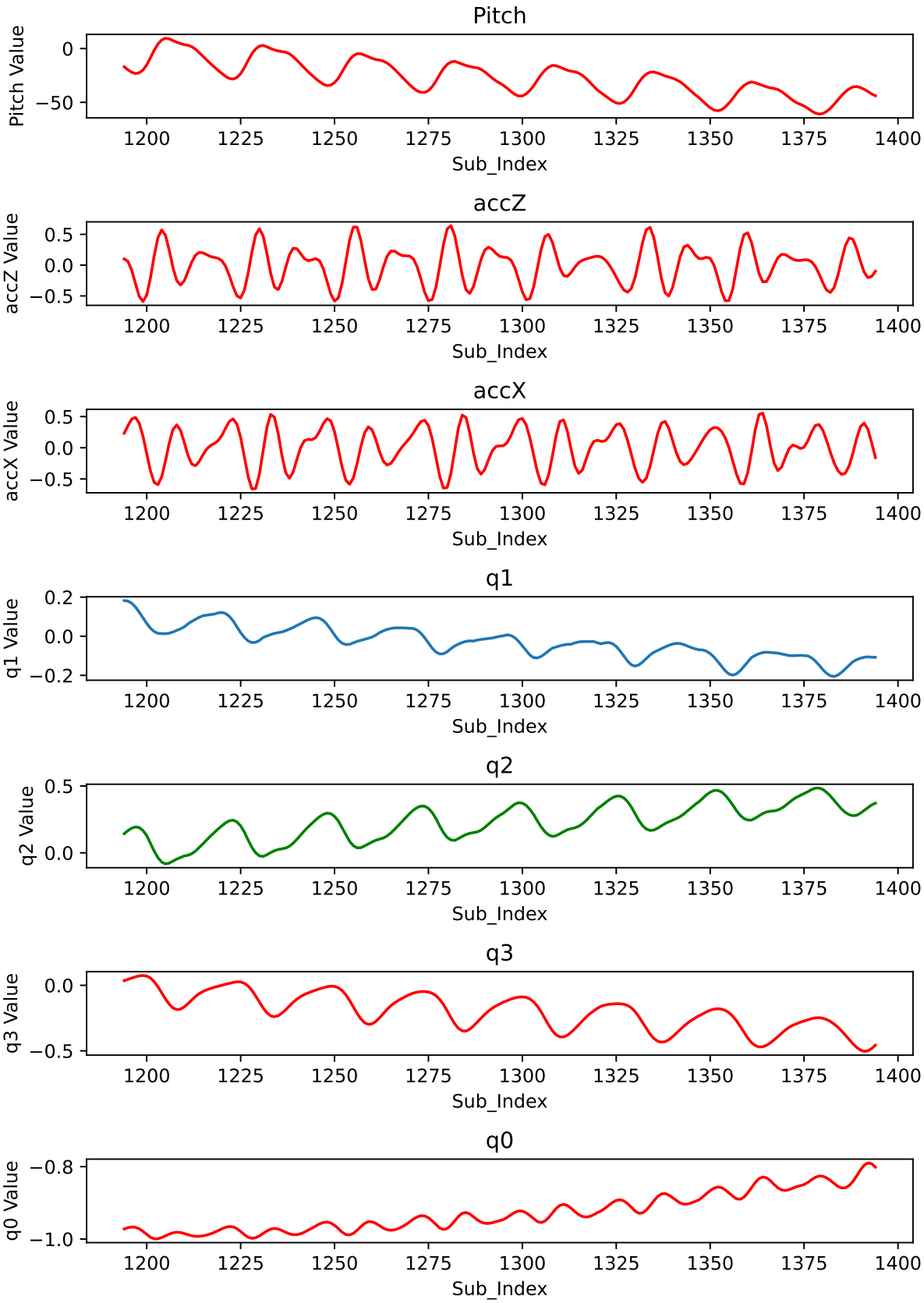
```
plt.show()
```


Investigating quaternion quality for Sample_interval_0
Pitch
Pitch Value
0
−25
1375
1400
1425
1450
1475
1500
1525
1550
Sub_Index
accZ
accZ Value
0.5
0.0
−0.5
accX
accX Value
q1
q1 Value
0.2
0.0
q2
q2 Value
0.25
0.00
q3
q3 Value
0.6
0.4
q0
q0 Value
−0.8
−0.9

```
import matplotlib.pyplot as plt


# Extracting part of df

name = 'Sample_interval_1'

df_subset = df_1.loc[8400:8600] #Random subset 2


plt.figure(figsize=(7.27, 10.69)) # calculated to fit an a4 with aprox 0.5 inch margins


plt.suptitle(f'Investigating quaternion quality for {name}', fontsize=14)



'''


plt.subplot(7, 1, 1)
plt.plot(df_subset['Sub_Index'], df_subset['Roll'], label='Roll', color='red')
plt.title('Roll')
plt.xlabel('Sub_Index')
plt.ylabel('Roll Value')


'''

plt.subplot(7, 1, 1)
plt.plot(df_subset['Sub_Index'], df_subset['Pitch'], label='Pitch', color='red')
plt.title('Pitch')
```

```
plt.xlabel('Sub_Index')
plt.ylabel('Pitch Value')


'''
plt.subplot(7, 1, 1)
plt.plot(df_subset['Sub_Index'], df_subset['Yaw'], label='Yaw', color='red')
plt.title('Yaw')
plt.xlabel('Sub_Index')
plt.ylabel('Yaw Value')
'''


plt.subplot(7, 1, 2)
plt.plot(df_subset['Sub_Index'], df_subset['accZ'], label='accZ', color='red')
plt.title('accZ')
plt.xlabel('Sub_Index')
plt.ylabel('accZ Value')

plt.subplot(7, 1, 3)
plt.plot(df_subset['Sub_Index'], df_subset['accX'], label='accX', color='red')
plt.title('accX')
plt.xlabel('Sub_Index')
plt.ylabel('accX Value')


plt.subplot(7, 1, 4)
plt.plot(df_subset['Sub_Index'], df_subset['q1'], label='q1')
plt.title('q1')
plt.xlabel('Sub_Index')
plt.ylabel('q1 Value')

plt.subplot(7, 1, 5)
plt.plot(df_subset['Sub_Index'], df_subset['q2'], label='q2', color='green')
```

```
plt.title('q2')
plt.xlabel('Sub_Index')
plt.ylabel('q2 Value')

plt.subplot(7, 1, 6)
plt.plot(df_subset['Sub_Index'], df_subset['q3'], label='q3', color='red')
plt.title('q3')
plt.xlabel('Sub_Index')
plt.ylabel('q3 Value')

plt.subplot(7, 1, 7)
plt.plot(df_subset['Sub_Index'], df_subset['q0'], label='q0', color='red')
plt.title('q0')
plt.xlabel('Sub_Index')
plt.ylabel('q0 Value')



plt.tight_layout()
plt.subplots_adjust(top=0.90)


plt.savefig(f'Quaternion_check_{name}.eps', format = 'eps')


plt.show()
```


Investigating quaternion quality for Sample_interval_1
Pitch
Pitch Value
0
−50
1200
1225
1250
1275
1300
1325
1350
1375
1400
Sub_Index
accZ
accZ Value
0.5
0.0
−0.5
Sub_Index
accX
accX Value
Sub_Index
q1
q1 Value
0.2
0.0
−0.2
Sub_Index
q2
q2 Value
0.5
0.0
Sub_Index
q3
q3 Value
0.0
−0.5
Sub_Index
q0
q0 Value
−0.8
−1.0
Sub_Index

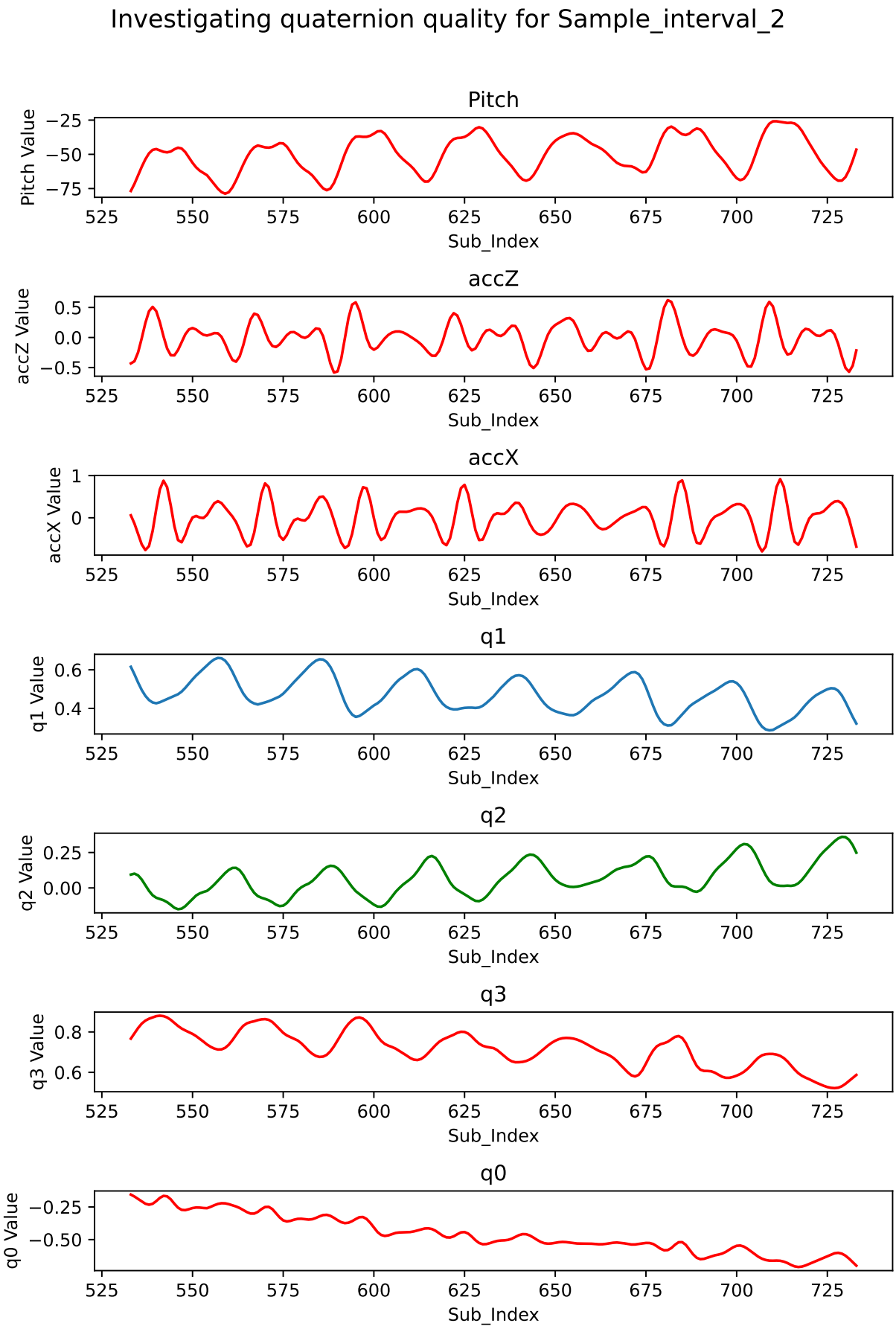
```
import matplotlib.pyplot as plt


# Extracting part of df

name = 'Sample_interval_2'

df_subset = df_1.loc[33000:33200] #Random subset 3


plt.figure(figsize=(7.27, 10.69)) # calculated to fit an a4 with aprox 0.5 inch margins


plt.suptitle(f'Investigating quaternion quality for {name}', fontsize=14)



'''


plt.subplot(7, 1, 1)
plt.plot(df_subset['Sub_Index'], df_subset['Roll'], label='Roll', color='red')
plt.title('Roll')
plt.xlabel('Sub_Index')
plt.ylabel('Roll Value')


'''

plt.subplot(7, 1, 1)
plt.plot(df_subset['Sub_Index'], df_subset['Pitch'], label='Pitch', color='red')
plt.title('Pitch')
```

```
plt.xlabel('Sub_Index')
plt.ylabel('Pitch Value')


'''
plt.subplot(7, 1, 1)
plt.plot(df_subset['Sub_Index'], df_subset['Yaw'], label='Yaw', color='red')
plt.title('Yaw')
plt.xlabel('Sub_Index')
plt.ylabel('Yaw Value')
'''


plt.subplot(7, 1, 2)
plt.plot(df_subset['Sub_Index'], df_subset['accZ'], label='accZ', color='red')
plt.title('accZ')
plt.xlabel('Sub_Index')
plt.ylabel('accZ Value')

plt.subplot(7, 1, 3)
plt.plot(df_subset['Sub_Index'], df_subset['accX'], label='accX', color='red')
plt.title('accX')
plt.xlabel('Sub_Index')
plt.ylabel('accX Value')


plt.subplot(7, 1, 4)
plt.plot(df_subset['Sub_Index'], df_subset['q1'], label='q1')
plt.title('q1')
plt.xlabel('Sub_Index')
plt.ylabel('q1 Value')

plt.subplot(7, 1, 5)
plt.plot(df_subset['Sub_Index'], df_subset['q2'], label='q2', color='green')
```

```
plt.title('q2')
plt.xlabel('Sub_Index')
plt.ylabel('q2 Value')

plt.subplot(7, 1, 6)
plt.plot(df_subset['Sub_Index'], df_subset['q3'], label='q3', color='red')
plt.title('q3')
plt.xlabel('Sub_Index')
plt.ylabel('q3 Value')

plt.subplot(7, 1, 7)
plt.plot(df_subset['Sub_Index'], df_subset['q0'], label='q0', color='red')
plt.title('q0')
plt.xlabel('Sub_Index')
plt.ylabel('q0 Value')
```



```
plt.tight_layout()
plt.subplots_adjust(top=0.90)


plt.savefig(f'Quaternion_check_{name}.eps', format = 'eps')


plt.show()
```


Investigating quaternion quality for Sample_interval_2
Pitch
Pitch Value
−25
−50
−75
525
550
575
600
625
650
675
700
725
Sub_Index
accZ
accZ Value
0.5
0.0
−0.5
Sub_Index
accX
accX Value
1
0
Sub_Index
q1
q1 Value
0.6
0.4
Sub_Index
q2
q2 Value
0.25
0.00
Sub_Index
q3
q3 Value
0.8
0.6
Sub_Index
q0
q0 Value
−0.25
−0.50
Sub_Index

prep data for ML (ID), remove calibration data and unnecessary columns

```
list(df_1.columns)
```

```
['Sub_Index',
 'Timestamp',
 'dt',
 'Subject_no',
 'q0',
 'q1',
 'q2',
 'q3',
 'Pitch',
 'Roll',
 'Yaw',
 'accX',
 'accY',
 'accZ',
 'gyrX',
 'gyrY',
 'gyrZ',
 'magX',
 'magY',
 'magZ']
```

```
df = df_1[['Timestamp',
 'dt',
 'Subject_no',
 'q0',
 'q1',
 'q2',
 'q3',
 'Pitch',
 'Roll',
 'Yaw',
 'accX',
```

```
 'accY',
 'accZ',
 'gyrX',
 'gyrY',
 'gyrZ',
 'magX',
 'magY',
 'magZ']]
```

```
df.to_csv('Check_')
```

Last subject measurement ends at 38758

```
df.loc[:38759]
```

```
        Timestamp        dt  Subject_no        q0        q1        q2  \
0        0.034056  0.034056           0  0.999981 -0.005295  -0.00305
1        0.068112  0.034056           0  0.999967 -0.007995 -0.001483
2        0.102168  0.034056           0  0.999951 -0.009922 -0.000083
3        0.142416  0.040248           0  0.999923 -0.012365  0.000831
4        0.182664  0.040248           0  0.999878 -0.015495  0.000539
...           ...       ...         ...       ...       ...       ...
38755  132.217924  0.039903          12 -0.980173  -0.19498 -0.004102
38756  132.257827  0.039903          12 -0.982184 -0.185914  0.004001
38757  132.297729  0.039902          12 -0.984022  -0.17534  0.004847
38758  132.337632  0.039903          12 -0.985002 -0.166676  0.002149
38759    0.029236  0.029236          33  0.999884 -0.003419  0.000101

             q3     Pitch      Roll         Yaw      accX      accY      accZ  \
0     -0.000229 -0.349617 -0.024413   -0.606688  0.001269 -0.007415  0.007225
1     -0.000205 -0.170067 -0.022105   -0.916112 -0.014983 -0.010159  0.003744
2     -0.000221 -0.009767 -0.025253   -1.136978 -0.024849 -0.013850 -0.001352
3     -0.000712  0.094222 -0.082716   -1.417035 -0.025351 -0.018286 -0.007889
4     -0.001899  0.058380 -0.218530   -1.775726 -0.018526 -0.021769 -0.014033
...         ...       ...       ...         ...       ...       ...       ...
38755  0.035017  1.243237 -3.843961 -337.540503  0.114871 -0.046202  0.108562
```

```
38756  0.027092  0.126920 -3.136616 -338.566529  0.002566 -0.098474  0.088519
38757  0.030541  0.067100 -3.544386 -339.795425 -0.110981 -0.107848  0.041200
38758   0.04456  0.608490 -5.078503 -340.818421 -0.178788 -0.079801 -0.012090
38759 -0.014872  0.005695 -1.704289 -360.391971 -0.176306 -0.037139 -0.050086

           gyrX      gyrY      gyrZ  magX  magY  magZ
0     -0.015511 -0.128565 -0.013460    72   -42     7
1      0.009013 -0.156497 -0.001248    73   -43     7
2      0.032911 -0.179508 -0.004759    72   -43     8
3      0.054869 -0.196698 -0.028749    72   -43     7
4      0.074107 -0.206459 -0.063698    72   -43     5
...         ...       ...       ...   ...   ...   ...
38755  0.023426 -0.404651  0.938618    55   -27    40
38756 -0.176852 -0.326740  0.471631    55   -28    41
38757 -0.331125 -0.271021 -0.084528    56   -30    39
38758 -0.334927 -0.259580 -0.616271    56   -31    36
38759 -0.172163 -0.286426 -1.017489    66   -24    27

[38760 rows x 19 columns]
```

```
df = df.loc[:38758]

df.to_csv('UIA_ID_Walking_Gait_Dataset_8_1_2024')
```

Appendix B

Appendix: Vu et. al Dataset

B.1 Vu et. al: GD data wrangling

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use('ggplot')
pd.set_option("display.max_columns", 200)
```

Load data

```
dataframe = pd.read_excel('/kaggle/input/vu-et-al-1020-steps/1020stepsFSRpercent100.xlsx')

df = dataframe
```

Evaluate dataset

```
df.shape
```

```
(129163, 26)
```

```
df.head(10)
```

```
   Unnamed: 0  Unnamed: 1  Unnamed: 2  Unnamed: 3  Unnamed: 4  Unnamed: 5  \
0       311.0       931.0       311.0       311.0       993.0      -809.0
1       358.0      1628.0       603.0       603.0       628.0      -534.0
2       603.0      1704.0      1270.0      1270.0       -74.0       -56.0
3       376.0      1316.0       828.0       828.0      -309.0      -746.0
4      -101.0       721.0       -25.0       -25.0      -374.0     -1068.0
5        57.0       723.0       -61.0       -61.0      -177.0      -548.0
6       160.0       844.0        80.0        80.0       -54.0      -134.0
7       197.0       935.0       222.0       222.0       -23.0        -2.0
8       138.0       943.0       250.0       250.0       -66.0       -26.0
9       139.0       948.0       242.0       242.0      -109.0        25.0

   Unnamed: 6  Unnamed: 7  Unnamed: 8  Unnamed: 9  Unnamed: 10  Unnamed: 11  \
0      2493.0      2679.0     -1199.0       289.0       -492.0       -120.0
1      1246.0      2779.0      -322.0        61.0       -343.0       -487.0
2      -420.0      2283.0       -32.0      -110.0         39.0       -800.0
3      -108.0      2056.0      -587.0       -87.0        261.0       -761.0
```

```
4       310.0      2047.0      -720.0      -110.0        288.0       -756.0
5       548.0      2115.0      -612.0      -147.0        279.0       -722.0
6       394.0      1934.0      -461.0      -130.0        253.0       -803.0
7       283.0      1782.0      -530.0       -43.0        181.0       -828.0
8       362.0      1835.0      -625.0       -23.0        144.0       -788.0
9       306.0      1858.0      -752.0       -22.0        144.0       -771.0

   Unnamed: 12  Unnamed: 13  Unnamed: 14  Unnamed: 15  Unnamed: 16  \
0       2074.0        648.0       1983.0       -300.0        905.0
1        493.0       1440.0       1443.0       -208.0       1304.0
2       -174.0       1027.0       1308.0       -169.0       2364.0
3       -107.0        480.0       2752.0       -455.0       2363.0
4        542.0        151.0       2558.0       -318.0       1979.0
5        482.0        181.0       2291.0       -366.0       1659.0
6        429.0        360.0       1864.0       -316.0       1283.0
7        538.0        572.0       1737.0       -229.0       1095.0
8        504.0        337.0       1716.0       -139.0       1114.0
9        488.0        441.0       1650.0       -134.0       1212.0

   Unnamed: 17  Subject  Unnamed: 19  Unnamed: 20  Unnamed: 21  Unnamed: 22  \
0       -583.0      0.0       13.850       13.850      13850.0     0.031411
1       -425.0      0.0       24.249       10.399      10399.0     0.094108
2       -127.0      0.0       34.629       10.380      10380.0     0.156434
3        179.0      0.0       47.995       13.366      13366.0     0.156434
4        142.0      0.0       63.785       15.790      15790.0     0.218143
5         25.0      0.0       74.077       10.292      10292.0     0.278991
6        -58.0      0.0       84.461       10.384      10384.0     0.278991
7         -6.0      0.0       96.590       12.129      12129.0     0.338738
8         -5.0      0.0      113.874       17.284      17284.0     0.397148
9        -39.0      0.0      124.573       10.699      10699.0     0.453990

   Unnamed: 23  Percent  Desi
0     0.999507      1.0   1.0
1     0.995562      2.0   1.0
```

```
2     0.987688      3.0   1.0
3     0.987688      3.0   1.0
4     0.975917      4.0   1.0
5     0.960294      5.0   1.0
6     0.960294      5.0   1.0
7     0.940881      6.0   1.0
8     0.917755      7.0   1.0
9     0.891007      8.0   1.0
```

Note: Columns 19, 20,and 21 seems to be time related

```
df.columns
```

```
Index(['Unnamed: 0', 'Unnamed: 1', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4',
       'Unnamed: 5', 'Unnamed: 6', 'Unnamed: 7', 'Unnamed: 8', 'Unnamed: 9',
       'Unnamed: 10', 'Unnamed: 11', 'Unnamed: 12', 'Unnamed: 13',
       'Unnamed: 14', 'Unnamed: 15', 'Unnamed: 16', 'Unnamed: 17', 'Subject',
       'Unnamed: 19', 'Unnamed: 20', 'Unnamed: 21', 'Unnamed: 22',
       'Unnamed: 23', 'Percent', 'Desi'],
      dtype='object')
```

Rename time-related columns + rename error (subject = gait phase)

```
df = df.rename(columns={'Unnamed: 19':'Time_Milli',
                        'Unnamed: 20':'Time_Delta_Milli',
                        'Unnamed: 21':'Time_Delta_Micro',
                        'Subject':'Gait_Phase'})


#df = df.rename(columns={'coaster_name':'Coaster_Name',
 #                       'year_introduced':'Year_Introduced',
  #                      'opening_date_clean':'Opening_Date',
   #                     'speed_mph':'Speed_mph',
    #                    'height_ft':'Height_ft',
     #                   'Inversions_clean':'Inversions',
      #                  'Gforce_clean':'Gforce'})
```

```
df.columns
```

```
Index(['Unnamed: 0', 'Unnamed: 1', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4',
       'Unnamed: 5', 'Unnamed: 6', 'Unnamed: 7', 'Unnamed: 8', 'Unnamed: 9',
       'Unnamed: 10', 'Unnamed: 11', 'Unnamed: 12', 'Unnamed: 13',
       'Unnamed: 14', 'Unnamed: 15', 'Unnamed: 16', 'Unnamed: 17',
       'Gait_Phase', 'Time_Milli', 'Time_Delta_Milli', 'Time_Delta_Micro',
       'Unnamed: 22', 'Unnamed: 23', 'Percent', 'Desi'],
      dtype='object')
```

Rename unknown IMU data columns

```
df = df.rename(columns={'Unnamed: 0':'Sensor_0',
                        'Unnamed: 1':'Sensor_1',
                        'Unnamed: 2':'Sensor_2',
                        'Unnamed: 3':'Sensor_3',
                        'Unnamed: 4':'Sensor_4',
                        'Unnamed: 5':'Sensor_5',
                        'Unnamed: 6':'Sensor_6',
                        'Unnamed: 7':'Sensor_7',
                        'Unnamed: 8':'Sensor_8',
                        'Unnamed: 9':'Sensor_9',
                        'Unnamed: 10':'Sensor_10',
                        'Unnamed: 11':'Sensor_11',
                        'Unnamed: 12':'Sensor_12',
                        'Unnamed: 13':'Sensor_13',
                        'Unnamed: 14':'Sensor_14',
                        'Unnamed: 15':'Sensor_15',
                        'Unnamed: 16':'Sensor_16',
                        'Unnamed: 17':'Sensor_17',
                        'Unnamed: 22':'Sensor_18',
                        'Unnamed: 23':'Sensor_19'})
```

```
df.columns
```

```
Index(['Sensor_0', 'Sensor_1', 'Sensor_2', 'Sensor_3', 'Sensor_4', 'Sensor_5',
       'Sensor_6', 'Sensor_7', 'Sensor_8', 'Sensor_9', 'Sensor_10',
```

```
       'Sensor_11', 'Sensor_12', 'Sensor_13', 'Sensor_14', 'Sensor_15',
       'Sensor_16', 'Sensor_17', 'Gait_Phase', 'Time_Milli',
       'Time_Delta_Milli', 'Time_Delta_Micro', 'Sensor_18', 'Sensor_19',
       'Percent', 'Desi'],
      dtype='object')
```

```
df.head()
```

	Sensor_0	Sensor_1	Sensor_2	Sensor_3	Sensor_4	Sensor_5	Sensor_6	\
0	311.0	931.0	311.0	311.0	993.0	-809.0	2493.0	
1	358.0	1628.0	603.0	603.0	628.0	-534.0	1246.0	
2	603.0	1704.0	1270.0	1270.0	-74.0	-56.0	-420.0	
3	376.0	1316.0	828.0	828.0	-309.0	-746.0	-108.0	
4	-101.0	721.0	-25.0	-25.0	-374.0	-1068.0	310.0	

	Sensor_7	Sensor_8	Sensor_9	Sensor_10	Sensor_11	Sensor_12	Sensor_13	\
0	2679.0	-1199.0	289.0	-492.0	-120.0	2074.0	648.0	
1	2779.0	-322.0	61.0	-343.0	-487.0	493.0	1440.0	
2	2283.0	-32.0	-110.0	39.0	-800.0	-174.0	1027.0	
3	2056.0	-587.0	-87.0	261.0	-761.0	-107.0	480.0	
4	2047.0	-720.0	-110.0	288.0	-756.0	542.0	151.0	

	Sensor_14	Sensor_15	Sensor_16	Sensor_17	Gait_Phase	Time_Milli	\
0	1983.0	-300.0	905.0	-583.0	0.0	13.850	
1	1443.0	-208.0	1304.0	-425.0	0.0	24.249	
2	1308.0	-169.0	2364.0	-127.0	0.0	34.629	
3	2752.0	-455.0	2363.0	179.0	0.0	47.995	
4	2558.0	-318.0	1979.0	142.0	0.0	63.785	

	Time_Delta_Milli	Time_Delta_Micro	Sensor_18	Sensor_19	Percent	Desi
0	13.850	13850.0	0.031411	0.999507	1.0	1.0
1	10.399	10399.0	0.094108	0.995562	2.0	1.0
2	10.380	10380.0	0.156434	0.987688	3.0	1.0
3	13.366	13366.0	0.156434	0.987688	3.0	1.0
4	15.790	15790.0	0.218143	0.975917	4.0	1.0

Reorder columns

```
neworder = ['Sensor_0',
            'Sensor_1',
            'Sensor_2',
            'Sensor_3',
            'Sensor_4',
            'Sensor_5',
            'Sensor_6',
            'Sensor_7',
            'Sensor_8',
            'Sensor_9',
            'Sensor_10',
            'Sensor_11',
            'Sensor_12',
            'Sensor_13',
            'Sensor_14',
            'Sensor_15',
            'Sensor_16',
            'Sensor_17',
            'Sensor_18',
            'Sensor_19',
            'Gait_Phase',
            'Time_Milli',
            'Time_Delta_Milli',
            'Time_Delta_Micro',
            'Percent',
            'Desi'
           ]
df=df.reindex(columns=neworder)
```

Datatypes for columns

```
df.columns
```

```
Index(['Sensor_0', 'Sensor_1', 'Sensor_2', 'Sensor_3', 'Sensor_4', 'Sensor_5',
```

```
       'Sensor_6', 'Sensor_7', 'Sensor_8', 'Sensor_9', 'Sensor_10',
       'Sensor_11', 'Sensor_12', 'Sensor_13', 'Sensor_14', 'Sensor_15',
       'Sensor_16', 'Sensor_17', 'Sensor_18', 'Sensor_19', 'Gait_Phase',
       'Time_Milli', 'Time_Delta_Milli', 'Time_Delta_Micro', 'Percent',
       'Desi'],
      dtype='object')
```

```
df.head(10)
```

	Sensor_0	Sensor_1	Sensor_2	Sensor_3	Sensor_4	Sensor_5	Sensor_6	\
0	311.0	931.0	311.0	311.0	993.0	-809.0	2493.0	
1	358.0	1628.0	603.0	603.0	628.0	-534.0	1246.0	
2	603.0	1704.0	1270.0	1270.0	-74.0	-56.0	-420.0	
3	376.0	1316.0	828.0	828.0	-309.0	-746.0	-108.0	
4	-101.0	721.0	-25.0	-25.0	-374.0	-1068.0	310.0	
5	57.0	723.0	-61.0	-61.0	-177.0	-548.0	548.0	
6	160.0	844.0	80.0	80.0	-54.0	-134.0	394.0	
7	197.0	935.0	222.0	222.0	-23.0	-2.0	283.0	
8	138.0	943.0	250.0	250.0	-66.0	-26.0	362.0	
9	139.0	948.0	242.0	242.0	-109.0	25.0	306.0	

	Sensor_7	Sensor_8	Sensor_9	Sensor_10	Sensor_11	Sensor_12	Sensor_13	\
0	2679.0	-1199.0	289.0	-492.0	-120.0	2074.0	648.0	
1	2779.0	-322.0	61.0	-343.0	-487.0	493.0	1440.0	
2	2283.0	-32.0	-110.0	39.0	-800.0	-174.0	1027.0	
3	2056.0	-587.0	-87.0	261.0	-761.0	-107.0	480.0	
4	2047.0	-720.0	-110.0	288.0	-756.0	542.0	151.0	
5	2115.0	-612.0	-147.0	279.0	-722.0	482.0	181.0	
6	1934.0	-461.0	-130.0	253.0	-803.0	429.0	360.0	
7	1782.0	-530.0	-43.0	181.0	-828.0	538.0	572.0	
8	1835.0	-625.0	-23.0	144.0	-788.0	504.0	337.0	
9	1858.0	-752.0	-22.0	144.0	-771.0	488.0	441.0	

	Sensor_14	Sensor_15	Sensor_16	Sensor_17	Sensor_18	Sensor_19	\
0	1983.0	-300.0	905.0	-583.0	0.031411	0.999507	

```
1     1443.0     -208.0     1304.0     -425.0   0.094108   0.995562
2     1308.0     -169.0     2364.0     -127.0   0.156434   0.987688
3     2752.0     -455.0     2363.0      179.0   0.156434   0.987688
4     2558.0     -318.0     1979.0      142.0   0.218143   0.975917
5     2291.0     -366.0     1659.0       25.0   0.278991   0.960294
6     1864.0     -316.0     1283.0      -58.0   0.278991   0.960294
7     1737.0     -229.0     1095.0       -6.0   0.338738   0.940881
8     1716.0     -139.0     1114.0       -5.0   0.397148   0.917755
9     1650.0     -134.0     1212.0      -39.0   0.453990   0.891007

   Gait_Phase  Time_Milli  Time_Delta_Milli  Time_Delta_Micro  Percent  Desi
0         0.0      13.850            13.850           13850.0      1.0   1.0
1         0.0      24.249            10.399           10399.0      2.0   1.0
2         0.0      34.629            10.380           10380.0      3.0   1.0
3         0.0      47.995            13.366           13366.0      3.0   1.0
4         0.0      63.785            15.790           15790.0      4.0   1.0
5         0.0      74.077            10.292           10292.0      5.0   1.0
6         0.0      84.461            10.384           10384.0      5.0   1.0
7         0.0      96.590            12.129           12129.0      6.0   1.0
8         0.0     113.874            17.284           17284.0      7.0   1.0
9         0.0     124.573            10.699           10699.0      8.0   1.0

df.describe()

            Sensor_0       Sensor_1       Sensor_2       Sensor_3  \
count  129160.000000  129162.000000  129162.000000  129162.000000
mean      145.363263    1144.771179     -97.487628      -4.372788
std       287.785416     545.386811     469.141985    2080.926494
min     -3416.000000   -3416.000000   -3397.000000   -6666.000000
25%        69.000000     925.000000    -178.000000    -213.000000
50%       161.000000    1020.000000     -36.000000     114.000000
75%       243.000000    1458.000000      48.000000     694.000000
max      5311.000000    5321.000000    5321.000000    6733.000000

            Sensor_4       Sensor_5       Sensor_6       Sensor_7  \
```

```
count  129162.000000  129162.000000  129162.000000  129162.000000
mean      -61.821372      66.898182    -350.833728    2111.413233
std       603.346724     864.716265     844.595219     664.393378
min     -6666.000000   -4884.000000   -5010.000000   -5010.000000
25%      -220.000000    -276.000000    -550.000000    1934.000000
50%        18.000000     -27.000000    -285.000000    2048.000000
75%       179.000000     342.000000       5.000000    2337.000000
max      6733.000000    3730.000000    7544.000000    7544.000000

            Sensor_8       Sensor_9      Sensor_10      Sensor_11  \
count  129162.000000  129162.000000  129162.000000  129162.000000
mean     -422.374166      13.442390     -35.874429     -18.647381
std       354.694848     630.013284     526.331573    1976.567512
min     -5359.000000   -5359.000000   -3343.000000   -3966.000000
25%      -614.000000    -343.000000    -292.000000   -1067.000000
50%      -464.000000    -115.000000     -44.000000    -655.000000
75%      -262.000000     156.000000     248.000000     577.000000
max      5321.000000    2557.000000    3326.000000    5722.000000

           Sensor_12      Sensor_13      Sensor_14      Sensor_15  \
count  129162.000000  129162.000000  129162.000000  129162.000000
mean     -299.366106     337.828618    2027.694012      57.199006
std      1141.577893     720.347149    1112.771213     877.005226
min    -22026.000000  -22026.000000  -10978.000000   -7490.000000
25%      -471.000000     204.000000    1747.000000    -258.000000
50%      -171.000000     368.000000    1824.000000     -10.000000
75%        11.000000     522.000000    2114.000000     243.000000
max     18462.000000   18462.000000   23833.000000   23833.000000

           Sensor_16      Sensor_17      Sensor_18      Sensor_19  \
count  129162.000000  129162.000000  129160.000000  129160.000000
mean      -25.485112    -119.998939       0.002452      -0.004079
std      2507.447166     577.637452       0.707048       0.707155
min     -7502.000000   -7502.000000      -0.999507      -0.999507
```

```
25%      -729.750000    -190.000000      -0.707107      -0.707107
50%        51.000000     -58.000000      -0.031411      -0.031411
75%       545.000000      54.000000       0.707107       0.707107
max     18422.000000   18422.000000       0.999507       0.999507

          Gait_Phase     Time_Milli  Time_Delta_Milli  Time_Delta_Micro  \
count  129160.000000  129160.000000     129160.000000     129160.000000
mean        1.673119  140586.173669         10.385115      10385.114788
std         1.086283   96163.213683          1.753104       1753.103706
min         0.000000       8.835000          5.138000       5138.000000
25%         1.000000   53007.195500          9.099000       9099.000000
50%         2.000000  136178.822000         10.183000      10183.000000
75%         3.000000  219514.986750         10.983000      10983.000000
max         3.000000  361330.840000         21.207000      21207.000000

             Percent          Desi
count  129160.000000  129160.000000
mean       50.750054       5.566329
std        28.794790       2.864900
min         1.000000       1.000000
25%        26.000000       3.000000
50%        51.000000       6.000000
75%        76.000000       8.000000
max       100.000000      10.000000

print(df)

        Sensor_0  Sensor_1  Sensor_2  Sensor_3  Sensor_4  Sensor_5  Sensor_6  \
0          311.0     931.0     311.0     311.0     993.0    -809.0    2493.0
1          358.0    1628.0     603.0     603.0     628.0    -534.0    1246.0
2          603.0    1704.0    1270.0    1270.0     -74.0     -56.0    -420.0
3          376.0    1316.0     828.0     828.0    -309.0    -746.0    -108.0
4         -101.0     721.0     -25.0     -25.0    -374.0   -1068.0     310.0
...          ...       ...       ...       ...       ...       ...       ...
129158   -1524.0    1797.0    -302.0       8.0    1136.0    -634.0   -1524.0
```

```
129159   -1469.0    1844.0    -315.0       4.0    1276.0    -936.0   -1469.0
129160       NaN       NaN       NaN       NaN       NaN       NaN       NaN
129161       NaN    5311.0    5321.0    3942.0    6733.0    3398.0    3730.0
129162       NaN   -3416.0   -1685.0   -3397.0   -6666.0   -4027.0   -4884.0

        Sensor_7  Sensor_8  Sensor_9  Sensor_10  Sensor_11  Sensor_12  \
0         2679.0   -1199.0     289.0     -492.0     -120.0     2074.0
1         2779.0    -322.0      61.0     -343.0     -487.0      493.0
2         2283.0     -32.0    -110.0       39.0     -800.0     -174.0
3         2056.0    -587.0     -87.0      261.0     -761.0     -107.0
4         2047.0    -720.0    -110.0      288.0     -756.0      542.0
...          ...       ...       ...        ...        ...        ...
129158    1797.0    -302.0       8.0     1136.0     -634.0    -1793.0
129159    1844.0    -315.0       4.0     1276.0     -936.0    -1913.0
129160       NaN       NaN       NaN        NaN        NaN        NaN
129161    7544.0    5321.0    2064.0     2557.0     3326.0     5722.0
129162   -5010.0     -63.0   -5359.0    -1601.0    -3343.0    -3966.0

        Sensor_13  Sensor_14  Sensor_15  Sensor_16  Sensor_17  Sensor_18  \
0           648.0     1983.0     -300.0      905.0     -583.0   0.031411
1          1440.0     1443.0     -208.0     1304.0     -425.0   0.094108
2          1027.0     1308.0     -169.0     2364.0     -127.0   0.156434
3           480.0     2752.0     -455.0     2363.0      179.0   0.156434
4           151.0     2558.0     -318.0     1979.0      142.0   0.218143
...           ...        ...        ...        ...        ...        ...
129158     -168.0      645.0     -618.0     1482.0      936.0  -0.031411
129159        1.0      458.0     -802.0     1881.0      858.0  -0.031411
129160        NaN        NaN        NaN        NaN        NaN        NaN
129161    18462.0     9873.0    23833.0     8282.0    18422.0        NaN
129162   -22026.0   -10978.0    -7064.0    -7490.0    -7502.0        NaN

        Sensor_19  Gait_Phase  Time_Milli  Time_Delta_Milli  Time_Delta_Micro  \
0        0.999507         0.0      13.850            13.850           13850.0
1        0.995562         0.0      24.249            10.399           10399.0
```

```
2        0.987688         0.0      34.629           10.380          10380.0
3        0.987688         0.0      47.995           13.366          13366.0
4        0.975917         0.0      63.785           15.790          15790.0
...           ...         ...         ...              ...              ...
129158   0.999507         3.0   15266.000           15.266          15266.0
129159   0.999507         3.0   11468.000           11.468          11468.0
129160        NaN         NaN         NaN              NaN              NaN
129161        NaN         NaN         NaN              NaN              NaN
129162        NaN         NaN         NaN              NaN              NaN

        Percent  Desi
0           1.0   1.0
1           2.0   1.0
2           3.0   1.0
3           3.0   1.0
4           4.0   1.0
...         ...   ...
129158    100.0  10.0
129159    100.0  10.0
129160      NaN   NaN
129161      NaN   NaN
129162      NaN   NaN

[129163 rows x 26 columns]
```

Numerical data analysis

Problem: Some nan values, and some differences in column lengths Removing rows with nan values, since they are grouped at end of df

```
#Find out where nan values are
df[df.isna().any(axis=1)]
```

```
        Sensor_0  Sensor_1  Sensor_2  Sensor_3  Sensor_4  Sensor_5  Sensor_6  \
129160       NaN       NaN       NaN       NaN       NaN       NaN       NaN
129161       NaN    5311.0    5321.0    3942.0    6733.0    3398.0    3730.0
129162       NaN   -3416.0   -1685.0   -3397.0   -6666.0   -4027.0   -4884.0
```

```
        Sensor_7  Sensor_8  Sensor_9  Sensor_10  Sensor_11  Sensor_12  \
129160       NaN       NaN       NaN        NaN        NaN        NaN
129161    7544.0    5321.0    2064.0     2557.0     3326.0     5722.0
129162   -5010.0     -63.0   -5359.0    -1601.0    -3343.0    -3966.0

        Sensor_13  Sensor_14  Sensor_15  Sensor_16  Sensor_17  Sensor_18  \
129160        NaN        NaN        NaN        NaN        NaN        NaN
129161    18462.0     9873.0    23833.0     8282.0    18422.0        NaN
129162   -22026.0   -10978.0    -7064.0    -7490.0    -7502.0        NaN

        Sensor_19  Gait_Phase  Time_Milli  Time_Delta_Milli  Time_Delta_Micro  \
129160        NaN         NaN         NaN               NaN               NaN
129161        NaN         NaN         NaN               NaN               NaN
129162        NaN         NaN         NaN               NaN               NaN

        Percent  Desi
129160      NaN   NaN
129161      NaN   NaN
129162      NaN   NaN
```

```
df = df.iloc[:-4, :]
```

Check new dataset for problematic values

```
df[df.isna().any(axis=1)]
```

Empty DataFrame

```
Columns: [Sensor_0, Sensor_1, Sensor_2, Sensor_3, Sensor_4, Sensor_5, Sensor_6, Sensor_7, Sensor_8, Sensor_9, ...
   Sensor_10, Sensor_11, Sensor_12, Sensor_13, Sensor_14, Sensor_15, Sensor_16, Sensor_17, Sensor_18, Sensor_19, ...
   Gait_Phase, Time_Milli, Time_Delta_Milli, Time_Delta_Micro, Percent, Desi]
Index: []
```

```
print(df)
```

```
        Sensor_0  Sensor_1  Sensor_2  Sensor_3  Sensor_4  Sensor_5  Sensor_6  \
0          311.0     931.0     311.0     311.0     993.0    -809.0    2493.0
1          358.0    1628.0     603.0     603.0     628.0    -534.0    1246.0
2          603.0    1704.0    1270.0    1270.0     -74.0     -56.0    -420.0
3          376.0    1316.0     828.0     828.0    -309.0    -746.0    -108.0
4         -101.0     721.0     -25.0     -25.0    -374.0   -1068.0     310.0
...          ...       ...       ...       ...       ...       ...       ...
129154   -1396.0    2103.0    -281.0     -22.0    1282.0     647.0   -1396.0
129155   -1528.0    2098.0     -16.0     -41.0    1281.0     299.0   -1528.0
129156   -1441.0    2042.0    -257.0     -35.0    1241.0      29.0   -1441.0
129157   -1529.0    1922.0    -347.0     -10.0    1156.0    -367.0   -1529.0
129158   -1524.0    1797.0    -302.0       8.0    1136.0    -634.0   -1524.0

        Sensor_7  Sensor_8  Sensor_9  Sensor_10  Sensor_11  Sensor_12  \
0         2679.0   -1199.0     289.0     -492.0     -120.0     2074.0
1         2779.0    -322.0      61.0     -343.0     -487.0      493.0
2         2283.0     -32.0    -110.0       39.0     -800.0     -174.0
3         2056.0    -587.0     -87.0      261.0     -761.0     -107.0
4         2047.0    -720.0    -110.0      288.0     -756.0      542.0
...          ...       ...       ...        ...        ...        ...
129154    2103.0    -281.0     -22.0     1282.0      647.0    -1727.0
129155    2098.0     -16.0     -41.0     1281.0      299.0    -1773.0
129156    2042.0    -257.0     -35.0     1241.0       29.0    -1816.0
129157    1922.0    -347.0     -10.0     1156.0     -367.0    -1880.0
129158    1797.0    -302.0       8.0     1136.0     -634.0    -1793.0

        Sensor_13  Sensor_14  Sensor_15  Sensor_16  Sensor_17  Sensor_18  \
0           648.0     1983.0     -300.0      905.0     -583.0   0.031411
1          1440.0     1443.0     -208.0     1304.0     -425.0   0.094108
2          1027.0     1308.0     -169.0     2364.0     -127.0   0.156434
3           480.0     2752.0     -455.0     2363.0      179.0   0.156434
4           151.0     2558.0     -318.0     1979.0      142.0   0.218143
...           ...        ...        ...        ...        ...        ...
129154     -148.0     1006.0       13.0      -11.0     1448.0  -0.156434
```

```
129155     -258.0      691.0     -108.0      394.0     1331.0  -0.156434
129156     -286.0      585.0     -241.0      740.0     1207.0  -0.094108
129157     -231.0      569.0     -440.0     1207.0     1029.0  -0.094108
129158     -168.0      645.0     -618.0     1482.0      936.0  -0.031411

        Sensor_19  Gait_Phase  Time_Milli  Time_Delta_Milli  Time_Delta_Micro  \
0        0.999507         0.0      13.850            13.850           13850.0
1        0.995562         0.0      24.249            10.399           10399.0
2        0.987688         0.0      34.629            10.380           10380.0
3        0.987688         0.0      47.995            13.366           13366.0
4        0.975917         0.0      63.785            15.790           15790.0
...           ...         ...         ...               ...               ...
129154   0.987688         3.0    9146.000             9.146            9146.0
129155   0.987688         3.0    8943.000             8.943            8943.0
129156   0.995562         3.0   12005.000            12.005           12005.0
129157   0.995562         3.0   10652.000            10.652           10652.0
129158   0.999507         3.0   15266.000            15.266           15266.0

        Percent  Desi
0           1.0   1.0
1           2.0   1.0
2           3.0   1.0
3           3.0   1.0
4           4.0   1.0
...         ...   ...
129154     98.0  10.0
129155     98.0  10.0
129156     99.0  10.0
129157     99.0  10.0
129158    100.0  10.0

[129159 rows x 26 columns]
```

Nan values removed

Setting dtypes to save computing power

```
df.dtypes
```

```
Sensor_0            float64
Sensor_1            float64
Sensor_2            float64
Sensor_3            float64
Sensor_4            float64
Sensor_5            float64
Sensor_6            float64
Sensor_7            float64
Sensor_8            float64
Sensor_9            float64
Sensor_10           float64
Sensor_11           float64
Sensor_12           float64
Sensor_13           float64
Sensor_14           float64
Sensor_15           float64
Sensor_16           float64
Sensor_17           float64
Sensor_18           float64
Sensor_19           float64
Gait_Phase          float64
Time_Milli          float64
Time_Delta_Milli    float64
Time_Delta_Micro    float64
Percent             float64
Desi                float64
dtype: object
```

```
statistical_summary = df.describe(include='all')
```

```
print(statistical_summary)
```

```
            Sensor_0       Sensor_1       Sensor_2       Sensor_3  \
count  129159.000000  129159.000000  129159.000000  129159.000000
mean      145.375762    1144.768820     -97.515605      -4.377140
std       287.751470     545.118758     468.883905    2080.900275
min     -3416.000000   -1685.000000   -3397.000000   -6666.000000
25%        69.000000     925.000000    -178.000000    -213.000000
50%       161.000000    1020.000000     -36.000000     114.000000
75%       243.000000    1458.000000      48.000000     694.000000
max      5311.000000    5321.000000    3942.000000    6733.000000

            Sensor_4       Sensor_5       Sensor_6       Sensor_7  \
count  129159.000000  129159.000000  129159.000000  129159.000000
mean      -61.833206      66.911853    -350.821569    2111.428379
std       602.765886     864.597088     844.428760     663.933055
min     -4027.000000   -4884.000000   -5010.000000     -63.000000
25%      -220.000000    -276.000000    -550.000000    1934.000000
50%        18.000000     -27.000000    -285.000000    2048.000000
75%       179.000000     342.000000       5.000000    2337.000000
max      3398.000000    3730.000000    7544.000000    5321.000000

            Sensor_8       Sensor_9      Sensor_10      Sensor_11  \
count  129159.000000  129159.000000  129159.000000  129159.000000
mean     -422.422247      13.468183     -35.892543     -18.640435
std       354.337227     629.817378     526.257556    1976.545264
min     -5359.000000   -1601.000000   -3343.000000   -3966.000000
25%      -614.000000    -343.000000    -292.000000   -1067.000000
50%      -464.000000    -115.000000     -44.000000    -655.000000
75%      -262.000000     156.000000     248.000000     577.000000
max      2064.000000    2557.000000    3326.000000    5722.000000

           Sensor_12      Sensor_13      Sensor_14      Sensor_15  \
count  129159.000000  129159.000000  129159.000000  129159.000000
mean     -299.371844     337.864051    2027.746119      57.076712
std      1141.413767     715.887970    1111.972704     874.288824
```

```
min    -22026.000000  -10978.000000   -7064.000000   -7490.000000
25%       -471.000000     204.000000    1747.000000    -258.000000
50%       -171.000000     368.000000    1824.000000     -10.000000
75%         11.000000     522.000000    2114.000000     243.000000
max      18462.000000    9873.000000   23833.000000    8282.000000

           Sensor_16      Sensor_17      Sensor_18      Sensor_19  \
count  129159.000000  129159.000000  129159.000000  129159.000000
mean      -25.506399    -120.092916       0.002453      -0.004087
std      2507.278094     574.962212       0.707051       0.707152
min     -7502.000000   -7458.000000      -0.999507      -0.999507
25%      -729.500000    -190.000000      -0.707107      -0.707107
50%        51.000000     -58.000000      -0.031411      -0.031411
75%       545.000000      54.000000       0.707107       0.707107
max     18422.000000    2890.000000       0.999507       0.999507

          Gait_Phase     Time_Milli  Time_Delta_Milli  Time_Delta_Micro  \
count  129159.000000  129159.000000     129159.000000     129159.000000
mean        1.673108  140587.173353         10.385106      10385.106404
std         1.086281   96162.914805          1.753108       1753.107903
min         0.000000       8.835000          5.138000       5138.000000
25%         1.000000   53008.695500          9.099000       9099.000000
50%         2.000000  136179.615000         10.183000      10183.000000
75%         3.000000  219515.475500         10.983000      10983.000000
max         3.000000  361330.840000         21.207000      21207.000000

             Percent           Desi
count  129159.000000  129159.000000
mean       50.749673       5.566294
std        28.794576       2.864884
min         1.000000       1.000000
25%        26.000000       3.000000
50%        51.000000       6.000000
75%        76.000000       8.000000
```

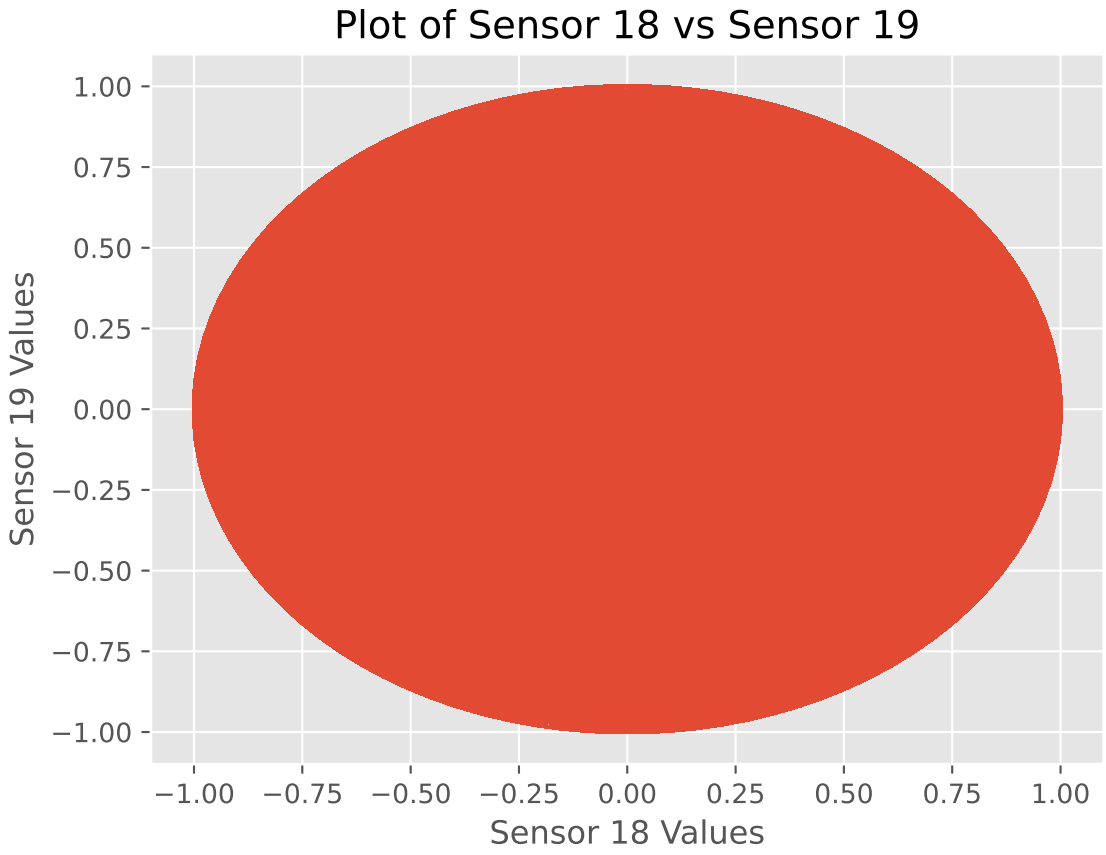
```
max       100.000000      10.000000
```

Sensor readings (columns) [0 , 17] all contain pure integer numbers with a max value around 23k. Typecast to int16 to save computing power.

Suspect Columns 'Sensor 18' and 'Sensor 19' exhibit characteristics of a sinusoidal nature, given that they demonstrate a confined numerical scope [-1, 1]. This pattern may also be the result of a normalization process.

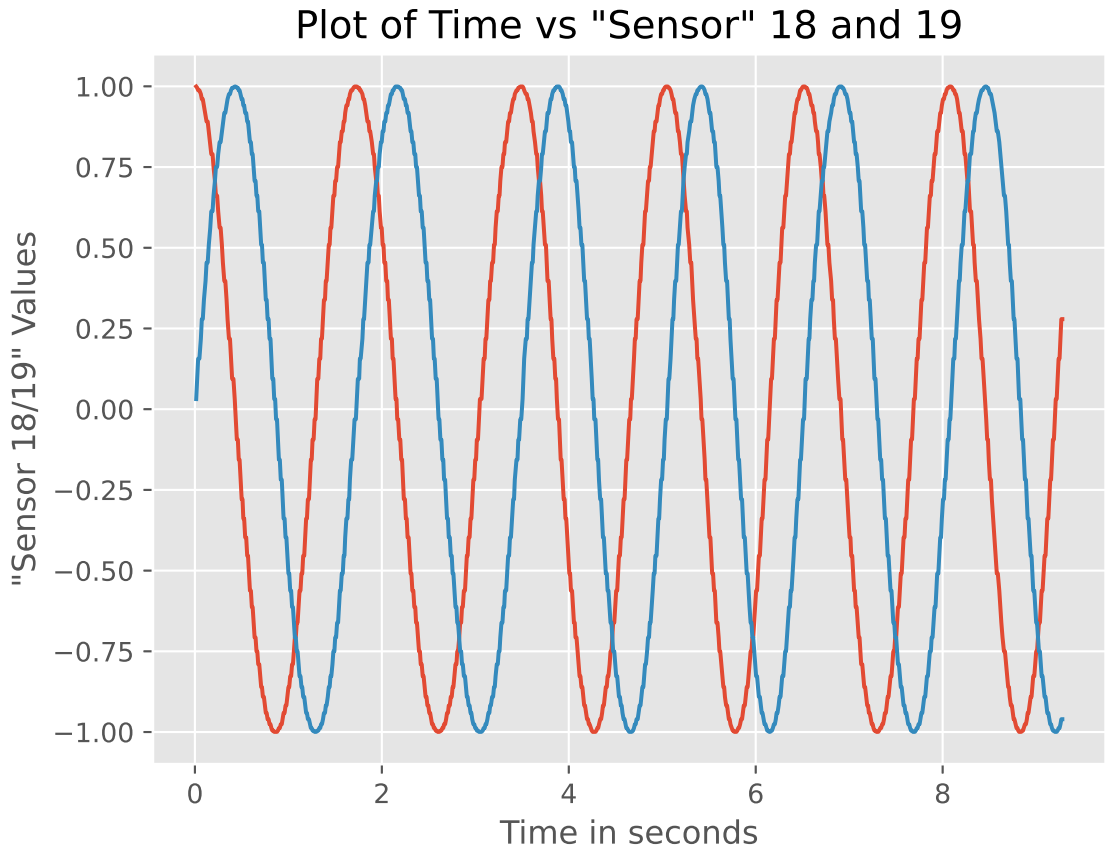
Given that the data was shared without any additional documentation, these columns are simply dropped. This is done to ensure that any successful machine learning implementation is not inadvertently a result of the columns being directly related to the label values, such as being representations of the label values in sinusoidal form.

```
plt.plot(df['Sensor_18'], df['Sensor_19'])

plt.xlabel('Sensor 18 Values')
plt.ylabel('Sensor 19 Values')
plt.title('Plot of Sensor 18 vs Sensor 19')
plt.savefig('Dropping_Sensor_18_19_circular.eps', format= 'eps')
```




```
start = 0
end = 900

plotdf = df[start:end:1]

plt.plot(plotdf['Time_Milli']/1e3, plotdf['Sensor_19'])
```

```
plt.plot(plotdf['Time_Milli']/1e3, plotdf['Sensor_18'])

plt.xlabel('Time in seconds')
plt.ylabel('"Sensor 18/19" Values')
plt.title('Plot of Time vs "Sensor" 18 and 19')

plt.savefig('Dropping_Sensor_18_19_sinusoidal.eps', format= 'eps')
```




```
df.columns
```

```
Index(['Sensor_0', 'Sensor_1', 'Sensor_2', 'Sensor_3', 'Sensor_4', 'Sensor_5',
       'Sensor_6', 'Sensor_7', 'Sensor_8', 'Sensor_9', 'Sensor_10',
       'Sensor_11', 'Sensor_12', 'Sensor_13', 'Sensor_14', 'Sensor_15',
       'Sensor_16', 'Sensor_17', 'Gait_Phase', 'Time_Milli',
       'Time_Delta_Milli', 'Time_Delta_Micro', 'Sensor_18', 'Sensor_19',
       'Percent', 'Desi'],
```

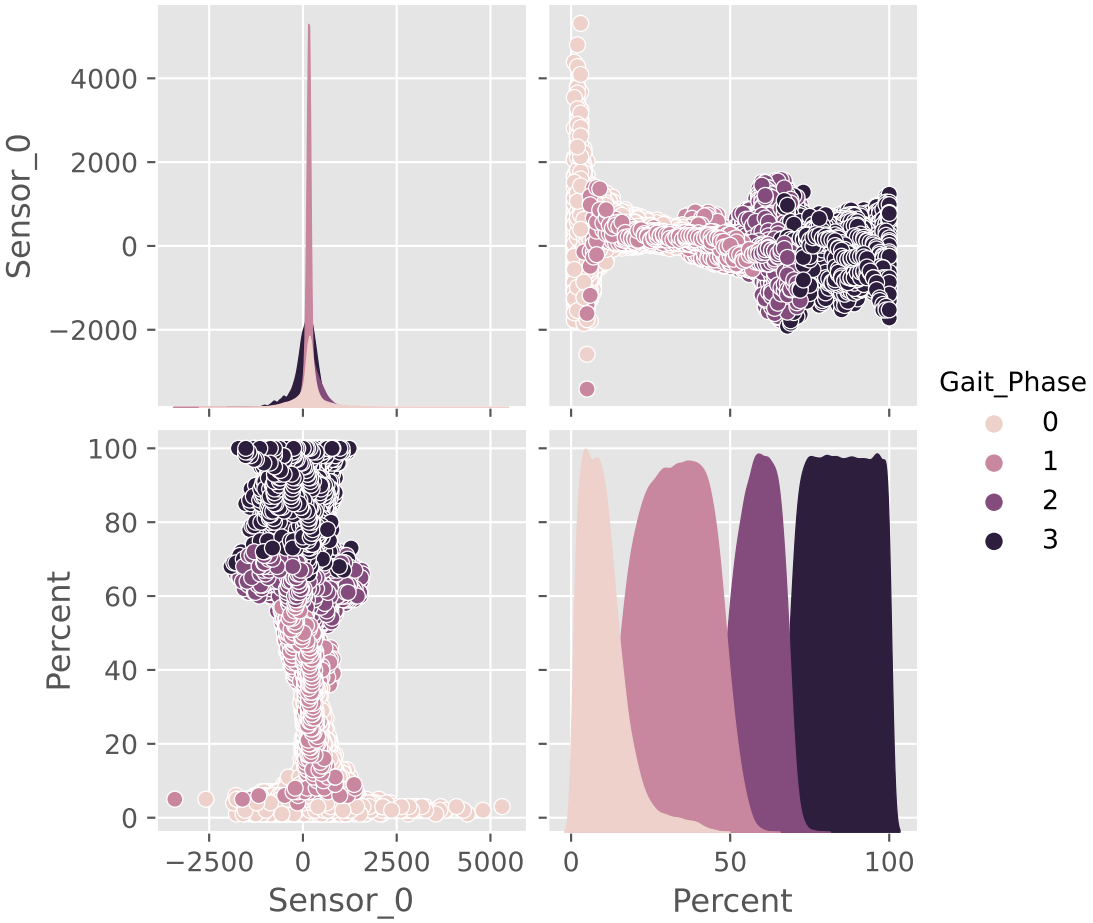
```
      dtype='object')


# df = df.astype({'col1': 'object', 'col2': 'int'})

df = df[['Sensor_0', 'Sensor_1', 'Sensor_2', 'Sensor_3', 'Sensor_4', 'Sensor_5',
       'Sensor_6', 'Sensor_7', 'Sensor_8', 'Sensor_9', 'Sensor_10',
       'Sensor_11', 'Sensor_12', 'Sensor_13', 'Sensor_14', 'Sensor_15',
       'Sensor_16', 'Sensor_17', 'Gait_Phase', 'Time_Milli',
       'Time_Delta_Milli', 'Time_Delta_Micro',
       'Percent', 'Desi']]

df = df.astype({
            'Sensor_0': 'int16',
            'Sensor_1': 'int16',
            'Sensor_2': 'int16',
            'Sensor_3': 'int16',
            'Sensor_4': 'int16',
            'Sensor_5': 'int16',
            'Sensor_6': 'int16',
            'Sensor_7': 'int16',
            'Sensor_8': 'int16',
            'Sensor_9': 'int16',
            'Sensor_10': 'int16',
            'Sensor_11': 'int16',
            'Sensor_12': 'int16',
            'Sensor_13': 'int16',
            'Sensor_14': 'int16',
            'Sensor_15': 'int16',
            'Sensor_16': 'int16',
            'Sensor_17': 'int16',
            'Percent' : 'int8',
            'Desi' : 'int8',
            'Gait_Phase' : 'int8'
})
```

```
df.dtypes
```

```
Sensor_0              int16
Sensor_1              int16
Sensor_2              int16
Sensor_3              int16
Sensor_4              int16
Sensor_5              int16
Sensor_6              int16
Sensor_7              int16
Sensor_8              int16
Sensor_9              int16
Sensor_10             int16
Sensor_11             int16
Sensor_12             int16
Sensor_13             int16
Sensor_14             int16
Sensor_15             int16
Sensor_16             int16
Sensor_17             int16
Gait_Phase             int8
Time_Milli          float64
Time_Delta_Milli    float64
Time_Delta_Micro    float64
Percent                int8
Desi                   int8
dtype: object
```

```
sns.pairplot(df,
             vars=['Sensor_0','Percent'],
            hue='Gait_Phase')
plt.show()
plt.savefig('Pairplot_s0_Percent_hue_GP.eps', format='eps')
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```


Sensor_0
4000
2000
0
−2000
Percent
100
80
60
40
20
0
−2500
0
2500
5000
Sensor_0
0
50
100
Percent
Gait_Phase
0
1
2
3

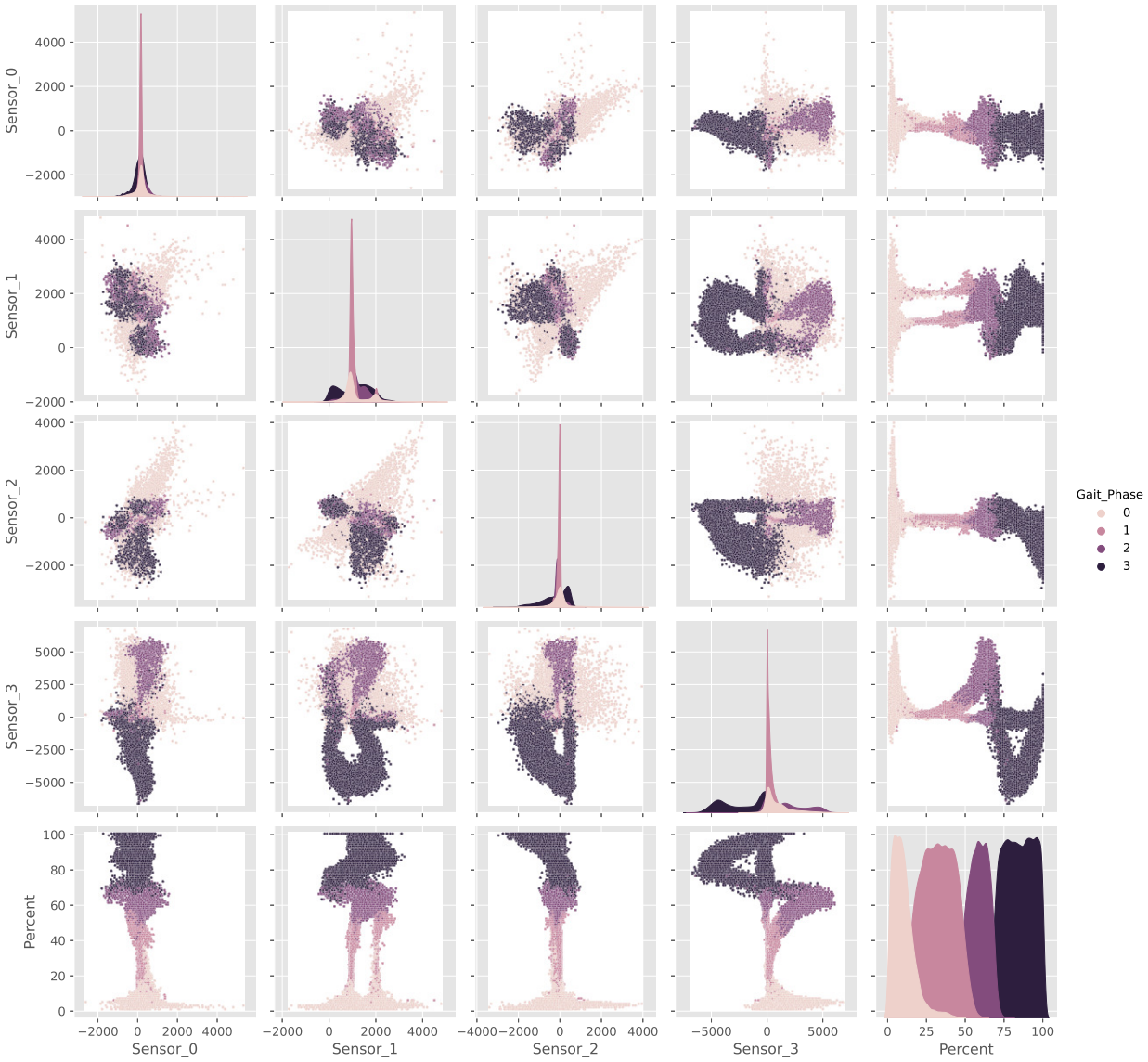
```
<Figure size 640x480 with 0 Axes>
```

Overlap in gait phase might be because of the use of FRS to detect gait events such as IC and TO

It is also interesting to note the higher variance in values measured in the IC-phase compared to other phases.

```
sns.pairplot(df,
             vars=['Sensor_0','Sensor_1','Sensor_2','Sensor_3','Percent'],
            hue='Gait_Phase')
plt.show()
plt.savefig('Pairplot_s0__s1_s2_s3_Percent_hue_GP.eps', format='eps')
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```


Sensor_0
Sensor_1
Sensor_2
Sensor_3
Percent
Gait_Phase
0
1
2
3

```
<Figure size 640x480 with 0 Axes>
```

Problem with time data in column 'Time_Milli'

starts in row 117440

117443?

```
# Time_Milli_Prob = df['Time_Milli']

# temp_dataframe = dataframe.loc[:, ['rain (mm)']].rolling(18).sum().round(2)

print(df.loc[117440::, ['Time_Milli']])
```

```
        Time_Milli
117440   361330.84
117441    12798.00
117442    10714.00
117443    10958.00
117444     8437.00
...            ...
129154     9146.00
129155     8943.00
129156    12005.00
129157    10652.00
129158    15266.00

[11719 rows x 1 columns]
```

Problem starts at 117441

```
print(df.loc[117441::, ['Time_Milli']])
```

```
        Time_Milli
117441     12798.0
117442     10714.0
117443     10958.0
117444      8437.0
```

```
117445      8951.0
...            ...
129154      9146.0
129155      8943.0
129156     12005.0
129157     10652.0
129158     15266.0

[11718 rows x 1 columns]

df.tail()

        Sensor_0  Sensor_1  Sensor_2  Sensor_3  Sensor_4  Sensor_5  Sensor_6  \
129154     -1396      2103      -281       -22      1282       647     -1396
129155     -1528      2098       -16       -41      1281       299     -1528
129156     -1441      2042      -257       -35      1241        29     -1441
129157     -1529      1922      -347       -10      1156      -367     -1529
129158     -1524      1797      -302         8      1136      -634     -1524

        Sensor_7  Sensor_8  Sensor_9  Sensor_10  Sensor_11  Sensor_12  \
129154      2103      -281       -22       1282        647      -1727
129155      2098       -16       -41       1281        299      -1773
129156      2042      -257       -35       1241         29      -1816
129157      1922      -347       -10       1156       -367      -1880
129158      1797      -302         8       1136       -634      -1793

        Sensor_13  Sensor_14  Sensor_15  Sensor_16  Sensor_17  Gait_Phase  \
129154       -148       1006         13        -11       1448           3
129155       -258        691       -108        394       1331           3
129156       -286        585       -241        740       1207           3
129157       -231        569       -440       1207       1029           3
129158       -168        645       -618       1482        936           3

        Time_Milli  Time_Delta_Milli  Time_Delta_Micro  Percent  Desi
129154      9146.0             9.146            9146.0       98    10
```

```
129155      8943.0             8.943            8943.0       98    10
129156     12005.0            12.005           12005.0       99    10
129157     10652.0            10.652           10652.0       99    10
129158     15266.0            15.266           15266.0      100    10
```

Replace Time_Micro with temp_column in range (117441,129158)

```
df_f = df.copy()

'''
import pandas as pd
import numpy as np

# Create a dataframe
df1 = pd.DataFrame({"A":[2, 3, 8, 14],
                    "B":[1, 2, 4, 3],
                    "C":[5, 3, 9,2]})

# Computing sum over Index axis
print(df1.cumsum(axis = 0))

'''


x = 117441
sum = 0


for x in range(117441,129159):

    # Fetch column entry from updated list in range(117441,129158)
    a = df_f.loc[x,['Time_Milli']]

    #print(a)
    a = a/1e3
    #print(a)
```

```
        sum = sum + a
        #print (sum)

        # Overwrite ['Time_Milli'] with summing time entry
        df_f.loc[x,['Time_Milli']] = sum

        #print(df_f.loc[x,['Time_Milli']])

        # Check result
       # print(df_f.loc[x, ['Time_Milli']])

#emp_dataframe = dataframe.loc[:, ['rain (mm)']].rolling(18).sum().round(2)



df_f.head(20)
```

```
    Sensor_0  Sensor_1  Sensor_2  Sensor_3  Sensor_4  Sensor_5  Sensor_6  \
0        311       931       311       311       993      -809      2493
1        358      1628       603       603       628      -534      1246
2        603      1704      1270      1270       -74       -56      -420
3        376      1316       828       828      -309      -746      -108
4       -101       721       -25       -25      -374     -1068       310
5         57       723       -61       -61      -177      -548       548
6        160       844        80        80       -54      -134       394
7        197       935       222       222       -23        -2       283
8        138       943       250       250       -66       -26       362
9        139       948       242       242      -109        25       306
10       156       949       238       238      -188        77       232
11       190       924       214       214      -265        18       165
12       214       925       191       191      -259       -73       148
13       209       932       165       165      -242      -213       151
14       187       931       142       142      -177      -429       213
15       163       928       125       125      -142      -443       234
16       131       955       151       151      -106      -399       109
```

```
17       143       984       159       159       -98      -351         1
18       130       978       135       135      -118      -316        -4
19       119       933        92        92      -132      -205        24

    Sensor_7  Sensor_8  Sensor_9  Sensor_10  Sensor_11  Sensor_12  Sensor_13  \
0       2679     -1199       289       -492       -120       2074        648
1       2779      -322        61       -343       -487        493       1440
2       2283       -32      -110         39       -800       -174       1027
3       2056      -587       -87        261       -761       -107        480
4       2047      -720      -110        288       -756        542        151
5       2115      -612      -147        279       -722        482        181
6       1934      -461      -130        253       -803        429        360
7       1782      -530       -43        181       -828        538        572
8       1835      -625       -23        144       -788        504        337
9       1858      -752       -22        144       -771        488        441
10      1912      -758       -33        107       -710        485        586
11      1947      -710       -45         52       -621        226        762
12      1959      -622       -55         51       -611         98        868
13      2026      -574       -44         98       -628        136        691
14      2085      -527       -17        152       -623        186        347
15      2098      -489         7        207       -645        120         78
16      2055      -454        18        227       -641         91        288
17      1977      -459        62        239       -612         52        382
18      1979      -577       116        216       -547         36        426
19      1989      -583       128        149       -449          3        586

    Sensor_14  Sensor_15  Sensor_16  Sensor_17  Gait_Phase  Time_Milli  \
0        1983       -300        905       -583           0      13.850
1        1443       -208       1304       -425           0      24.249
2        1308       -169       2364       -127           0      34.629
3        2752       -455       2363        179           0      47.995
4        2558       -318       1979        142           0      63.785
5        2291       -366       1659         25           0      74.077
6        1864       -316       1283        -58           0      84.461
```

7	1737	-229	1095	-6	0	96.590
8	1716	-139	1114	-5	0	113.874
9	1650	-134	1212	-39	0	124.573
10	1658	-137	1381	-28	0	136.702
11	1824	-232	1517	0	0	148.697
12	1881	-286	1405	93	0	164.299
13	1818	-319	1394	211	0	175.114
14	1941	-300	1389	254	0	188.692
15	2190	-124	1213	192	0	199.357
16	1945	109	1127	148	0	214.262
17	1997	265	1086	136	0	224.799
18	1967	398	1070	74	0	237.938
19	2000	240	927	12	0	249.052

	Time_Delta_Milli	Time_Delta_Micro	Percent	Desi
0	13.850	13850.0	1	1
1	10.399	10399.0	2	1
2	10.380	10380.0	3	1
3	13.366	13366.0	3	1
4	15.790	15790.0	4	1
5	10.292	10292.0	5	1
6	10.384	10384.0	5	1
7	12.129	12129.0	6	1
8	17.284	17284.0	7	1
9	10.699	10699.0	8	1
10	12.129	12129.0	8	1
11	11.995	11995.0	9	1
12	15.602	15602.0	10	1
13	10.815	10815.0	11	2
14	13.578	13578.0	11	2
15	10.665	10665.0	12	2
16	14.905	14905.0	13	2
17	10.537	10537.0	14	2
18	13.139	13139.0	14	2

```
19            11.114           11114.0       15     2
```

```
df_f.tail(5)
```

```
        Sensor_0  Sensor_1  Sensor_2  Sensor_3  Sensor_4  Sensor_5  Sensor_6  \
129154     -1396      2103      -281       -22      1282       647     -1396
129155     -1528      2098       -16       -41      1281       299     -1528
129156     -1441      2042      -257       -35      1241        29     -1441
129157     -1529      1922      -347       -10      1156      -367     -1529
129158     -1524      1797      -302         8      1136      -634     -1524

        Sensor_7  Sensor_8  Sensor_9  Sensor_10  Sensor_11  Sensor_12  \
129154      2103      -281       -22       1282        647      -1727
129155      2098       -16       -41       1281        299      -1773
129156      2042      -257       -35       1241         29      -1816
129157      1922      -347       -10       1156       -367      -1880
129158      1797      -302         8       1136       -634      -1793

        Sensor_13  Sensor_14  Sensor_15  Sensor_16  Sensor_17  Gait_Phase  \
129154       -148       1006         13        -11       1448           3
129155       -258        691       -108        394       1331           3
129156       -286        585       -241        740       1207           3
129157       -231        569       -440       1207       1029           3
129158       -168        645       -618       1482        936           3

        Time_Milli  Time_Delta_Milli  Time_Delta_Micro  Percent  Desi
129154  129845.436             9.146            9146.0       98    10
129155  129854.379             8.943            8943.0       98    10
129156  129866.384            12.005           12005.0       99    10
129157  129877.036            10.652           10652.0       99    10
129158  129892.302            15.266           15266.0      100    10
```

```
df_f.to_csv('Vu_et_al_clean.csv')
```

B.2 Vu et. al: GD label analysis

```
df = pd.read_csv('/kaggle/input/vu-et-al-clean/Vu_et_al_clean.csv')
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
plt.style.use('ggplot')
```

Analysis: Gait phase reckognition, justifying the need for application

Is the data cyclic? How many instances are there of each label?

Consideration: The data is not continous, as mulitple sessions/users has been stringed together in the dataframe.

From data analysis, we know data time resets at x = 117441

Create sub-set of data from row x until end + create additional, smaller subset for plots

```
x = 117441
y = 2985
df_s = df.iloc[x:,:]

df_75 = df.iloc[x+y:(x+y+75),:]

df_100 = df.iloc[x+y:(x+y+100),:]

df_300 = df.iloc[x+y:(x+y+300),:]

df_450 = df.iloc[x+y:(x+y+450),:]


df_750 = df.iloc[x+y:(x+y+750),:]
```

Analysis 1: Exploring 'Percent' label

```
Percent = df_s['Percent']
```

```
df_s.query('Percent == 100')
```

```
        Unnamed: 0  Sensor_0  Sensor_1  Sensor_2  Sensor_3  Sensor_4  \
117609      117609      -646      1966       -19       -25       503
117610      117610      -380      2268        84       -93       767
117771      117771      -457      2228      -162       -66       588
117772      117772      -568      2192      -117       -72       450
117932      117932     -1017      2016      -292        93       912
...            ...       ...       ...       ...       ...       ...
128783      128783     -1017      1889       -97        47      1180
128970      128970      -761      1933       -97      -137       462
128971      128971      -871      1928       -16      -124       512
128972      128972     -1151      1875      -304        -2       767
129158      129158     -1524      1797      -302         8      1136

        Sensor_5  Sensor_6  Sensor_7  Sensor_8  ...  Sensor_16  Sensor_17  \
117609       193      -646      1966       -19  ...        354        411
117610       -76      -380      2268        84  ...        812        215
117771      -261      -457      2228      -162  ...        827        199
117772      -348      -568      2192      -117  ...       1097       -149
117932       -99     -1017      2016      -292  ...       1040       1106
...          ...       ...       ...       ...  ...        ...        ...
128783      -123     -1017      1889       -97  ...       3079       1975
128970       392      -761      1933       -97  ...       1027        962
128971       195      -871      1928       -16  ...       1503        994
128972       -64     -1151      1875      -304  ...       1795        975
129158      -634     -1524      1797      -302  ...       1482        936

        Sensor_18  Sensor_19  Gait_Phase  Time_Milli  Time_Delta_Milli  \
117609  -0.031411   0.999507           3    1864.903            10.194
117610  -0.031411   0.999507           3    1877.354            12.451
117771  -0.031411   0.999507           3    3661.816            10.864
117772  -0.031411   0.999507           3    3672.088            10.272
117932  -0.031411   0.999507           3    5463.414            10.625
...           ...        ...         ...         ...               ...
```

```
128783  -0.031411   0.999507           3  125852.424            10.200
128970  -0.031411   0.999507           3  127864.376            10.610
128971  -0.031411   0.999507           3  127874.413            10.037
128972  -0.031411   0.999507           3  127883.630             9.217
129158  -0.031411   0.999507           3  129892.302            15.266

        Time_Delta_Micro  Percent  Desi
117609           10194.0      100    10
117610           12451.0      100    10
117771           10864.0      100    10
117772           10272.0      100    10
117932           10625.0      100    10
...                  ...      ...   ...
128783           10200.0      100    10
128970           10610.0      100    10
128971           10037.0      100    10
128972            9217.0      100    10
129158           15266.0      100    10

[134 rows x 27 columns]
```

```
df_s.query('Percent == 75')
```

```
        Unnamed: 0  Sensor_0  Sensor_1  Sensor_2  Sensor_3  Sensor_4  \
117571      117571       520      1234      -230       271        -2
117730      117730       -82      1760      -272      -658       217
117731      117731      -115      1957      -352      -655        47
117890      117890       460      1385      -241         1       -81
117891      117891       517      1384       -18        53       -69
...            ...       ...       ...       ...       ...       ...
128731      128731      -939      1521      -429       104      -126
128920      128920      -334      1201      -265      -135       510
128921      128921      -253       919       -16       -44       511
129113      129113       145      1213       -21         0       254
129114      129114         5      1312       -82      -312       -12
```

```
        Sensor_5  Sensor_6  Sensor_7  Sensor_8  ...  Sensor_16  Sensor_17  \
117571      -360       520      1234      -230  ...       -513       -195
117730      2426       -82      1760      -272  ...      -2719        299
117731      2600      -115      1957      -352  ...      -2895        138
117890        54       460      1385      -241  ...       -472       -460
117891       309       517      1384       -18  ...       -631       -501
...          ...       ...       ...       ...  ...        ...        ...
128731      -980      -939      1521      -429  ...       2098      -1710
128920      -312      -334      1201      -265  ...      -1535        303
128921        -7      -253       919       -16  ...      -1530        269
129113      -371       145      1213       -21  ...      -1311       -745
129114      -174         5      1312       -82  ...      -1215       -707

        Sensor_18  Sensor_19  Gait_Phase  Time_Milli  Time_Delta_Milli  \
117571  -0.999507  -0.031411           3    1397.404             9.867
117730  -0.999507  -0.031411           3    3211.658            13.449
117731  -0.999507  -0.031411           3    3222.763            11.105
117890  -0.999507  -0.031411           3    5012.289            12.716
117891  -0.999507  -0.031411           3    5022.735            10.446
...           ...        ...         ...         ...               ...
128731  -0.999507  -0.031411           3  125330.411            10.329
128920  -0.999507  -0.031411           3  127360.703             9.866
128921  -0.999507  -0.031411           3  127370.998            10.295
129113  -0.999507  -0.031411           3  129383.696             9.353
129114  -0.999507  -0.031411           3  129392.676             8.980

        Time_Delta_Micro  Percent  Desi
117571            9867.0       75     8
117730           13449.0       75     8
117731           11105.0       75     8
117890           12716.0       75     8
117891           10446.0       75     8
...                  ...      ...   ...
```

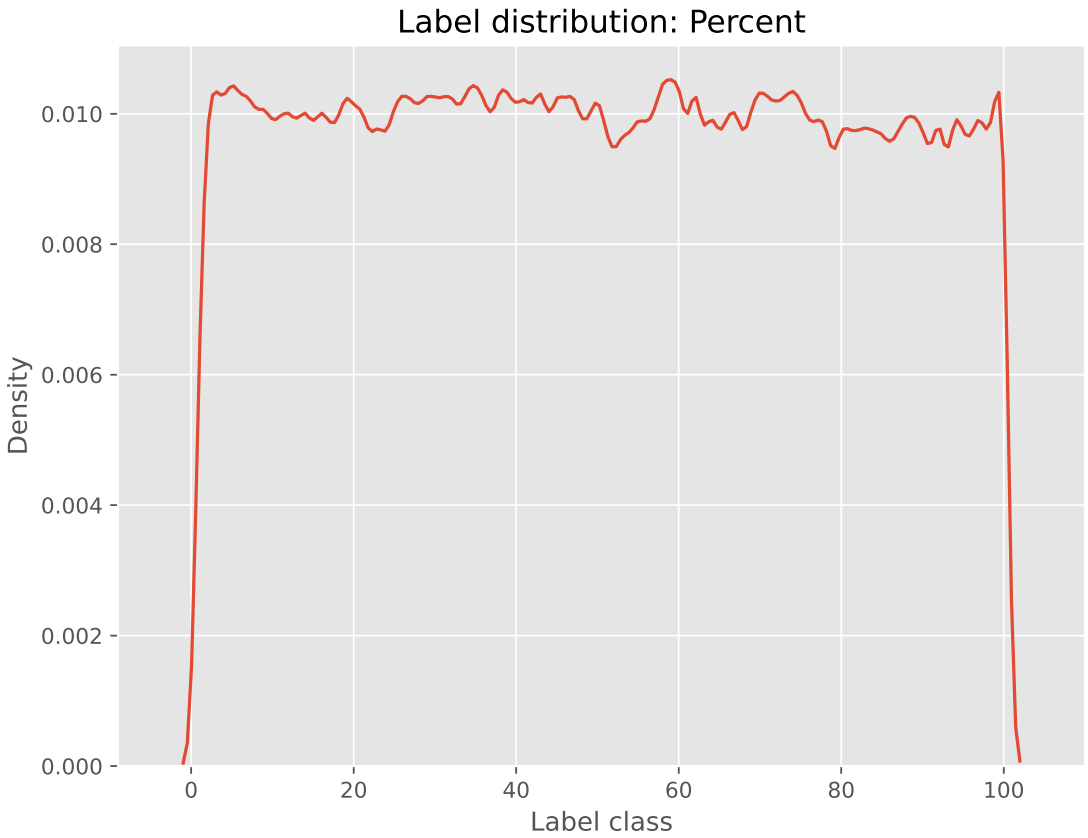
```
128731           10329.0       75     8
128920            9866.0       75     8
128921           10295.0       75     8
129113            9353.0       75     8
129114            8980.0       75     8

[120 rows x 27 columns]
```

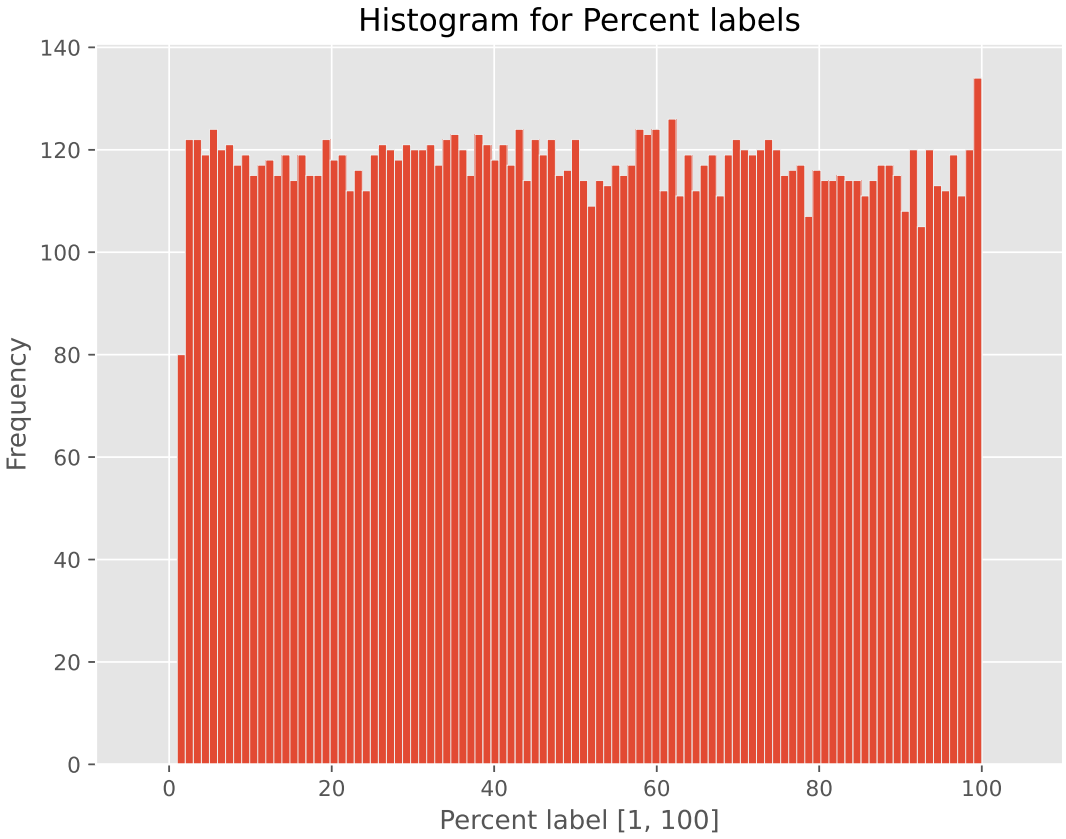
The number of instances for label 100 is not equal to the number of instances of label 75.

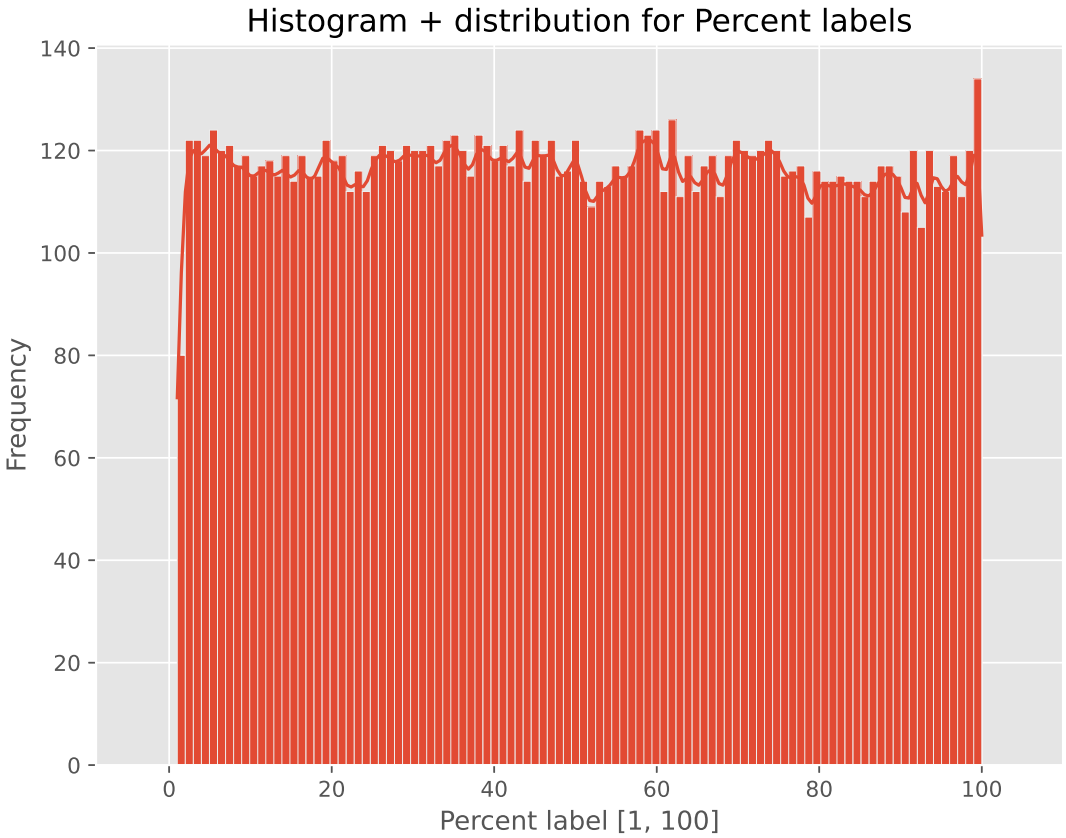
```
labels = 'Percent'

interval = (1,100)

'''
percent_kde = df_s['Percent'].plot(kind='kde',
                        title='Labeled value: Percent')
percent_kde.set_xlabel('Class_Label')

plt.savefig('')

'''

import matplotlib.pyplot as plt
import seaborn as sns

addon = 0.1

interval_p = (interval[0]-interval[1]*addon, interval[1]+interval[1]*addon)

#Dist:


data = df_s[labels]

plt.figure(figsize=(8, 6))
```

```
sns.kdeplot(data, bw_adjust=0.15)
''' going very fine-grain on the ditribution to
highlight any dominating class
'''

plt.title(f'Label distribution: {labels}')

plt.xlabel('Label class')

plt.xlim(interval_p)

plt.savefig(f'{labels}_1_distribution.eps', format='eps')
plt.savefig(f'{labels}_distribution.jpeg', format='jpeg')

plt.show()

plt.close()

#Hist:

plt.figure(figsize=(8,6))



sns.histplot(data, bins=interval[1])

plt.title(f'Histogram for {labels} labels')
plt.xlabel(f'{labels} label [{interval[0]}, {interval[1]}]')
plt.ylabel('Frequency')

plt.xlim(interval_p)

plt.savefig(f'{labels}_histogram.eps', format='eps')
```

```
plt.show()


plt.close()



#Hist and dist in same:

plt.figure(figsize=(8,6))



sns.histplot(data, bins=interval[1], kde=True, kde_kws={'bw_adjust': 0.15})

plt.title(f'Histogram + distribution for {labels} labels')
plt.xlabel(f'{labels} label [{interval[0]}, {interval[1]}]')
plt.ylabel('Frequency')

plt.xlim(interval_p)

plt.savefig(f'{labels}_histogram_distribution.eps', format='eps')
plt.savefig(f'{labels}_histogram_distribution.jpeg', format='jpeg')

plt.show()
```


Label distribution: Percent
Density
0.010
0.008
0.006
0.004
0.002
0.000
0
20
40
60
80
100
Label class


Histogram for Percent labels
Frequency
Percent label [1, 100]
0
20
40
60
80
100
120
140


Histogram + distribution for Percent labels
Frequency
Percent label [1, 100]
0
20
40
60
80
100
120
140

Distribution: Possible variation in count for all the label values.

Histogram confirms this.

Class 100 count seems to be much higher.

```
Counts = df_s['Percent'].value_counts()
```

Print the count for all label values, descending order

```
print(Counts.to_string())
```

```
Percent
100    134
62     126
5      124
60     124
58     124
43     124
35     123
59     123
38     123
45     122
34     122
19     122
50     122
47     122
2      122
70     122
74     122
3      122
26     121
39     121
32     121
29     121
41     121
7      121
```

36	120
6	120
71	120
92	120
73	120
75	120
94	120
99	120
31	120
30	120
27	120
64	119
9	119
67	119
69	119
25	119
46	119
14	119
21	119
72	119
97	119
4	119
16	119
12	118
20	118
28	118
40	118
11	117
78	117
42	117
88	117
55	117
57	117
33	117

8	117
66	117
89	117
77	116
80	116
23	116
49	116
13	115
17	115
18	115
76	115
37	115
90	115
48	115
56	115
83	115
10	115
84	114
81	114
87	114
82	114
85	114
51	114
53	114
44	114
15	114
54	113
95	113
65	112
61	112
24	112
96	112
22	112
98	111

```
63     111
86     111
68     111
52     109
91     108
79     107
93     105
1       80
```

```
Counts.describe()
```

```
count    100.000000
mean     117.180000
std        5.796864
min       80.000000
25%      114.750000
50%      118.000000
75%      120.000000
max      134.000000
Name: count, dtype: float64
```

```
Counts.var()
```

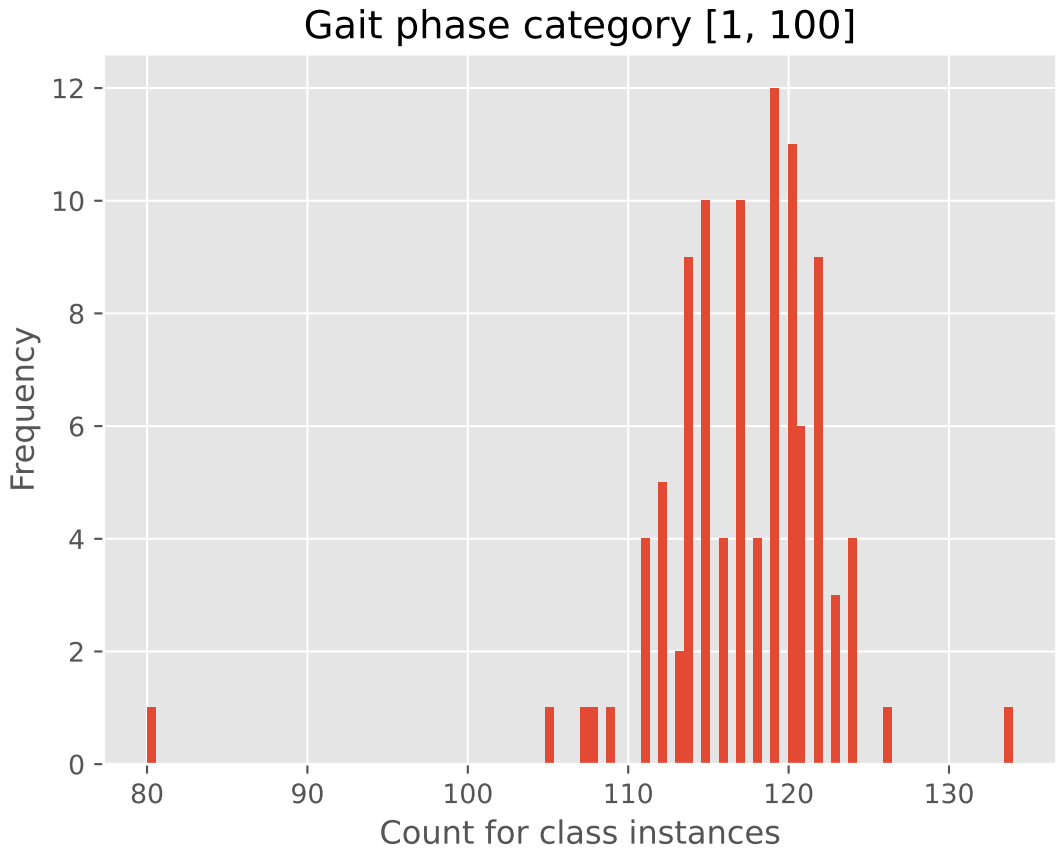
```
33.603636363636355
```

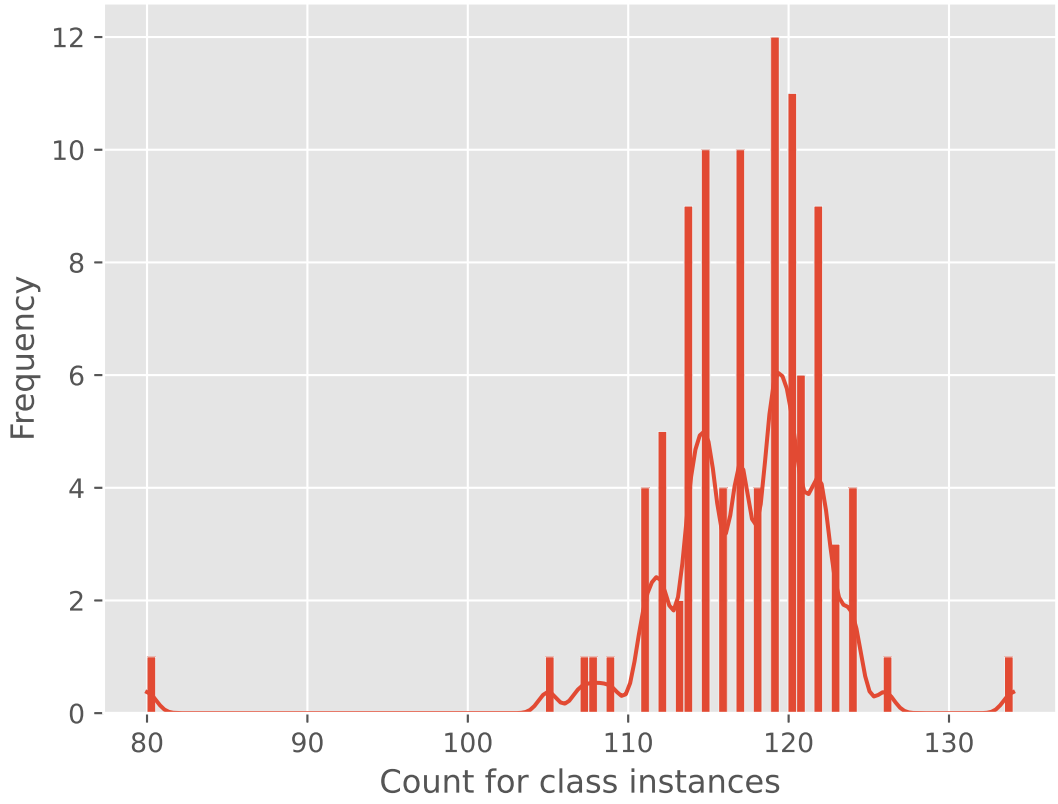
There is variance for all classes. Plot the count for all label values

```
Count_hist = Counts.plot(kind='hist',
                         bins=100,
                        title='Gait phase category [1, 100]')
Count_hist.set_xlabel('Count for class instances')
```

```
plt.savefig(f'{labels}_histogram.eps', format='eps')
plt.savefig(f'{labels}_histogram.jpeg', format='jpeg')
```

```
plt.show()
plt.close()


his_dist_count = sns.histplot(Counts, bins=interval[1], kde=True, kde_kws={'bw_adjust': 0.25})

his_dist_count.set_xlabel('Count for class instances')
his_dist_count.set_ylabel('Frequency')


plt.savefig(f'{labels}_counts_histogram_distribution.eps', format='eps')
plt.savefig(f'{labels}_counts_histogram_distribution.jpeg', format='jpeg')

plt.show()
plt.close()
```


Gait phase category [1, 100]
Frequency
Count for class instances
0
2
4
6
8
10
12
80
90
100
110
120
130



12 of the labels have equal occurence in the dataset.

The above graph displays the frequency of sum for all label counts, range [1, 100].

From print we can read that count for label_1 = 80 and count for label_100 = 134. This huge difference might be linked to mislabeling of data in the tranistion between each step. The labeling technique might be biased towards selecting 100(end of step) over 1 (start of step) in the transition from one step to another. It could also be a result of one of the label classes happening in a shorter temoporal interval. This can be because of the nature of walking gait: Certain parts of the gait may happen faster than others.

Finding 1: There is variance in the distribution of label values for Percent label.

The variance is fairly big throughout the entire dataset, and the categorization of each frame of data can not be excecuted simply by following a cyclic time-dependent pattern. This is in other words a good task for a ML model.

Analysis 2: 'Desi' label

```
Desi = df_s['Desi']
```

```
df_s.query('Desi == 10')
```

```
        Unnamed: 0  Sensor_0  Sensor_1  Sensor_2  Sensor_3  Sensor_4  \
117594      117594      -879      2662      -248      -209       173
117595      117595     -1062      2546      -331      -255       263
117596      117596     -1015      2468      -269      -143       270
117597      117597      -843      2393       -19      -107       265
117598      117598      -815      2310       -18       -88       406
...            ...       ...       ...       ...       ...       ...
129154      129154     -1396      2103      -281       -22      1282
129155      129155     -1528      2098       -16       -41      1281
129156      129156     -1441      2042      -257       -35      1241
129157      129157     -1529      1922      -347       -10      1156
129158      129158     -1524      1797      -302         8      1136

        Sensor_5  Sensor_6  Sensor_7  Sensor_8  ...  Sensor_16  Sensor_17  \
117594      2776      -879      2662      -248  ...      -1565        345
117595      2605     -1062      2546      -331  ...      -1535        491
117596      2471     -1015      2468      -269  ...      -1284        595
117597      2322      -843      2393       -19  ...      -1287        689
117598      2093      -815      2310       -18  ...      -1575        893
...          ...       ...       ...       ...  ...        ...        ...
129154       647     -1396      2103      -281  ...        -11       1448
129155       299     -1528      2098       -16  ...        394       1331
129156        29     -1441      2042      -257  ...        740       1207
129157      -367     -1529      1922      -347  ...       1207       1029
129158      -634     -1524      1797      -302  ...       1482        936
```

```
        Sensor_18  Sensor_19  Gait_Phase  Time_Milli  Time_Delta_Milli  \
117594  -0.612907   0.790155           3    1686.055            12.322
117595  -0.562083   0.827081           3    1701.498            15.443
117596  -0.509041   0.860742           3    1712.491            10.993
117597  -0.509041   0.860742           3    1723.796            11.305
117598  -0.453990   0.891007           3    1734.872            11.076
...           ...        ...         ...         ...               ...
129154  -0.156434   0.987688           3  129845.436             9.146
129155  -0.156434   0.987688           3  129854.379             8.943
129156  -0.094108   0.995562           3  129866.384            12.005
129157  -0.094108   0.995562           3  129877.036            10.652
129158  -0.031411   0.999507           3  129892.302            15.266

        Time_Delta_Micro  Percent  Desi
117594           12322.0       90    10
117595           15443.0       91    10
117596           10993.0       92    10
117597           11305.0       92    10
117598           11076.0       93    10
...                  ...      ...   ...
129154            9146.0       98    10
129155            8943.0       98    10
129156           12005.0       99    10
129157           10652.0       99    10
129158           15266.0      100    10

[1208 rows x 27 columns]

df_s.query('Desi == 7')

        Unnamed: 0  Sensor_0  Sensor_1  Sensor_2  Sensor_3  Sensor_4  \
117548      117548      -230      2210      -144      -144      -265
117549      117549      -200      2038      -146       -90      -258
117550      117550      -217      2048      -222       -38      -232
117551      117551      -218      2116      -238       -23      -239
```

```
117552      117552      -284      2083      -341       -11      -295
...            ...       ...       ...       ...       ...       ...
129099      129099      -274      2257      -366        82      -350
129100      129100      -690      2225      -462        70       -50
129101      129101     -1047      2301      -617       118       306
129102      129102     -1028      2436      -528        72       592
129103      129103      -914      2240      -622         2       722

        Sensor_5  Sensor_6  Sensor_7  Sensor_8  ...  Sensor_16  Sensor_17  \
117548     -1190      -230      2210      -144  ...        166         20
117549     -1233      -200      2038      -146  ...        190          7
117550     -1318      -217      2048      -222  ...        144         14
117551     -1353      -218      2116      -238  ...        245        -44
117552     -1452      -284      2083      -341  ...        282        -75
...          ...       ...       ...       ...  ...        ...        ...
129099     -2129      -274      2257      -366  ...       1004       -158
129100     -2262      -690      2225      -462  ...       2100       -373
129101     -2382     -1047      2301      -617  ...       3297       -553
129102     -2285     -1028      2436      -528  ...       4354       -687
129103     -2038      -914      2240      -622  ...       5575       -906

        Sensor_18  Sensor_19  Gait_Phase  Time_Milli  Time_Delta_Milli  \
117548  -0.562083  -0.827081           2    1125.417            16.484
117549  -0.612907  -0.790155           2    1135.511            10.094
117550  -0.661312  -0.750111           2    1145.913            10.402
117551  -0.661312  -0.750111           2    1154.474             8.561
117552  -0.661312  -0.750111           2    1163.027             8.553
...           ...        ...         ...         ...               ...
129099  -0.891007  -0.453990           2  129242.028             9.735
129100  -0.891007  -0.453990           2  129250.526             8.498
129101  -0.917755  -0.397148           2  129262.035            11.509
129102  -0.917755  -0.397148           2  129272.946            10.911
129103  -0.940881  -0.338738           2  129282.903             9.957
```

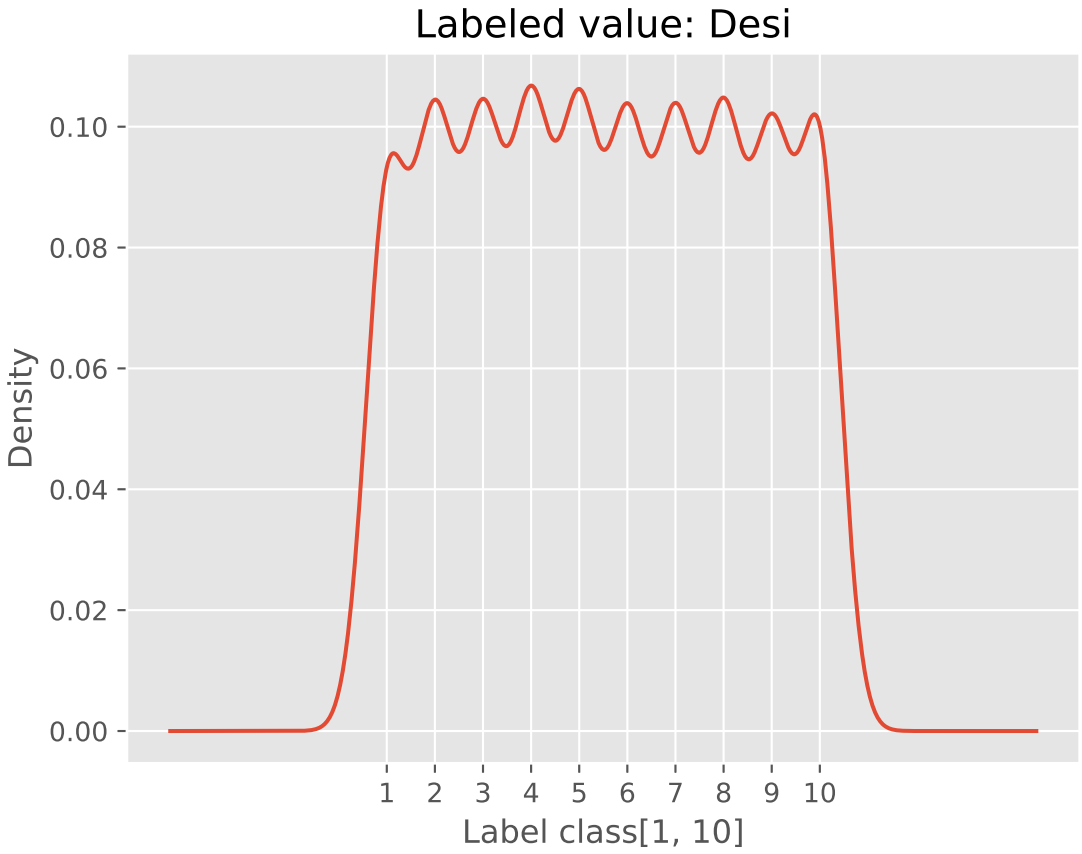
```
        Time_Delta_Micro  Percent  Desi
117548           16484.0       60     7
117549           10094.0       61     7
117550           10402.0       62     7
117551            8561.0       62     7
117552            8553.0       62     7
...                  ...      ...   ...
129099            9735.0       68     7
129100            8498.0       68     7
129101           11509.0       69     7
129102           10911.0       69     7
129103            9957.0       70     7

[1166 rows x 27 columns]
```

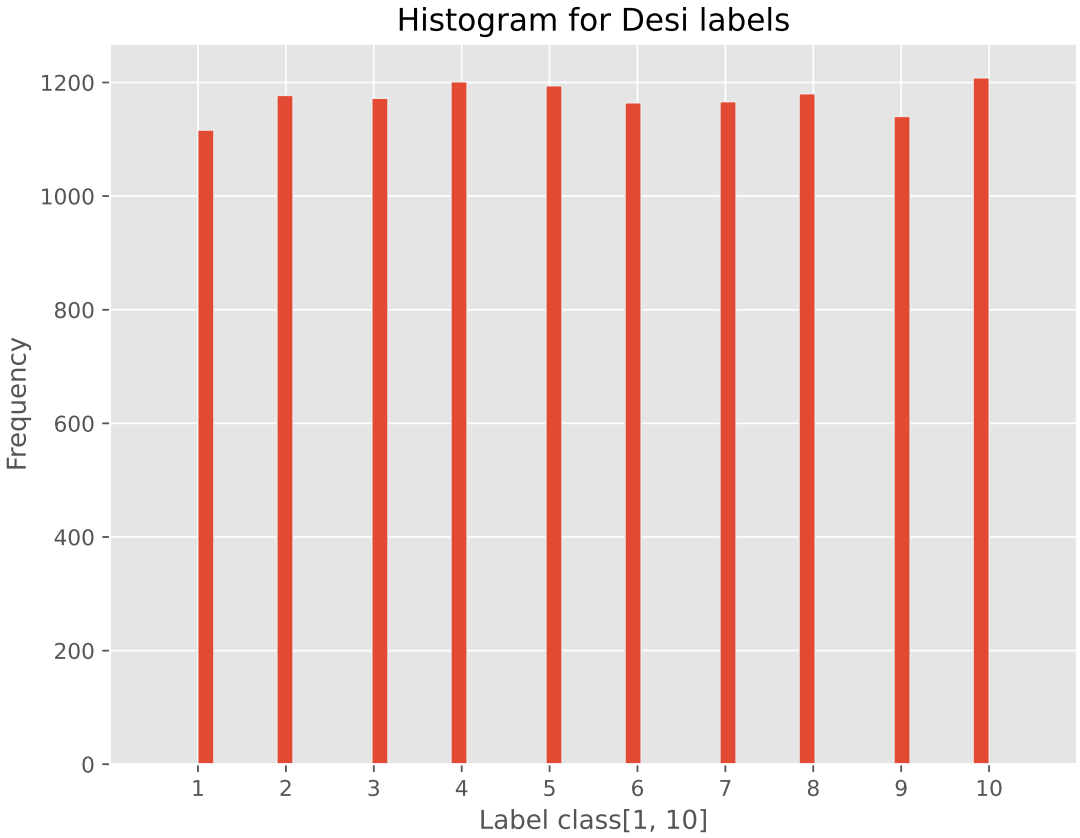
The number of instances for label 10 and 7 does not match. Plot the distribution of class instances:


```
labels = 'Desi'

interval = (1,10)

data = df_s[labels]

import matplotlib.pyplot as plt
import seaborn as sns

addon = 0.1

interval_p = (interval[0]-interval[1]*addon, interval[1]+interval[1]*addon)

#Dist:


Desi_kde = df_s['Desi'].plot(kind='kde',
                        title='Labeled value: Desi')
```

```
Desi_kde.set_xlabel(f'Label class[{interval[0]}, {interval[1]}]')

plt.xticks(np.unique(data))

plt.savefig(f'{labels}_distribution.eps', format='eps')
plt.savefig(f'{labels}_distribution.jpeg', format='jpeg')

plt.show()

plt.close()

#Hist:

plt.figure(figsize=(8,6))



sns.histplot(data, bins=50)

plt.title(f'Histogram for {labels} labels')
plt.xlabel(f'Label class[{interval[0]}, {interval[1]}]')
plt.ylabel('Frequency')

plt.xlim(interval_p)
plt.xticks(np.unique(data)) #absolute hacker

plt.savefig(f'{labels}_histogram.eps', format='eps')
plt.savefig(f'{labels}_histogram.jpeg', format='jpeg')
plt.show()


plt.close()

'''
```

```
    Distribution not as useful for few classes.

    Skipping combined hist and dist.

    '''
```


Labeled value: Desi
Density
0.10
0.08
0.06
0.04
0.02
0.00
1
2
3
4
5
6
7
8
9
10
Label class[1, 10]


Histogram for Desi labels
Frequency
0
200
400
600
800
1000
1200
1
2
3
4
5
6
7
8
9
10
Label class[1, 10]

```
'\n\nDistribution not as useful for few classes. \n\nSkipping combined hist and dist.\n\n'
```

```
Desi = df_s['Desi']


Counts_Desi = Desi.value_counts()
```

```
print(Counts_Desi.to_string())
```

```
Desi
10    1208
4     1201
5     1194
8     1180
2     1177
3     1172
7     1166
6     1164
9     1140
1     1116
```


Once again the biggest difference in count is between the first [1] and last [10] label

```
Counts_Desi.describe()
```

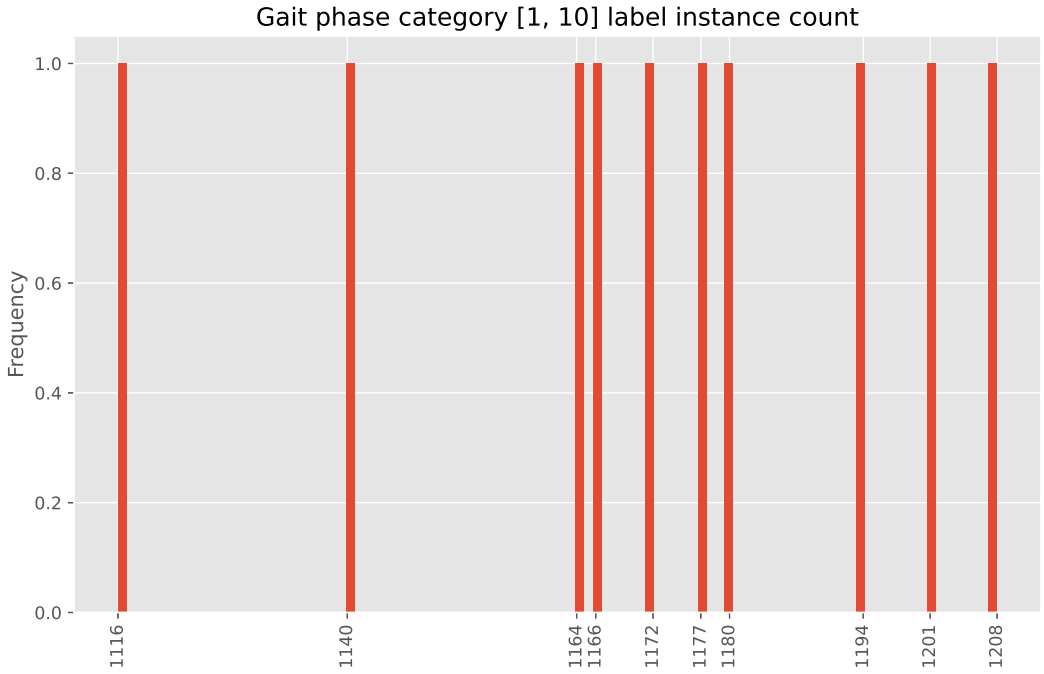
```
count      10.000000
mean     1171.800000
std        27.828043
min      1116.000000
25%      1164.500000
50%      1174.500000
75%      1190.500000
max      1208.000000
Name: count, dtype: float64
```

```
Counts_Desi.var()
```

```
774.4
```

Variance in the count for the different labels. Histogram for counts (not labels, count for labels:

```
labels = 'Desi'

plt.figure(figsize=(10, 6))

Count_hist_d = Counts_Desi.plot(kind='hist',
                          bins=100,
                         title='Gait phase category [1, 10] label instance count')
Count_hist_d.set_xlabel('')

plt.xticks(np.unique(Counts_Desi), rotation=90); #Long numbers, use vertical to make room

plt.savefig(f'{labels}_count_histogram.eps', format='eps')
plt.savefig(f'{labels}_count_histogram.jpeg', format='jpeg')
```


Gait phase category [1, 10] label instance count
Frequency
1.0
0.8
0.6
0.4
0.2
0.0
1116
1140
1164
1166
1172
1177
1180
1194
1201
1208

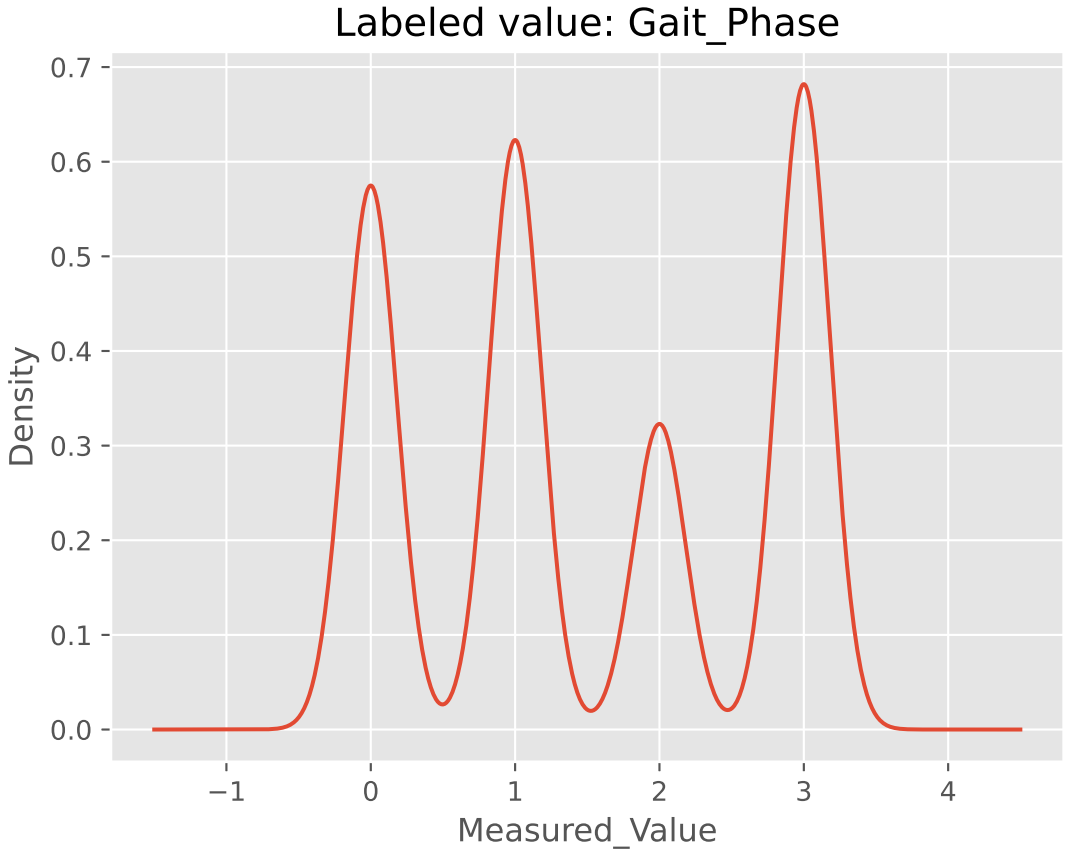
Finding 2: As for Percent, the Desi label has varying count for the differnt bins [1, 10]

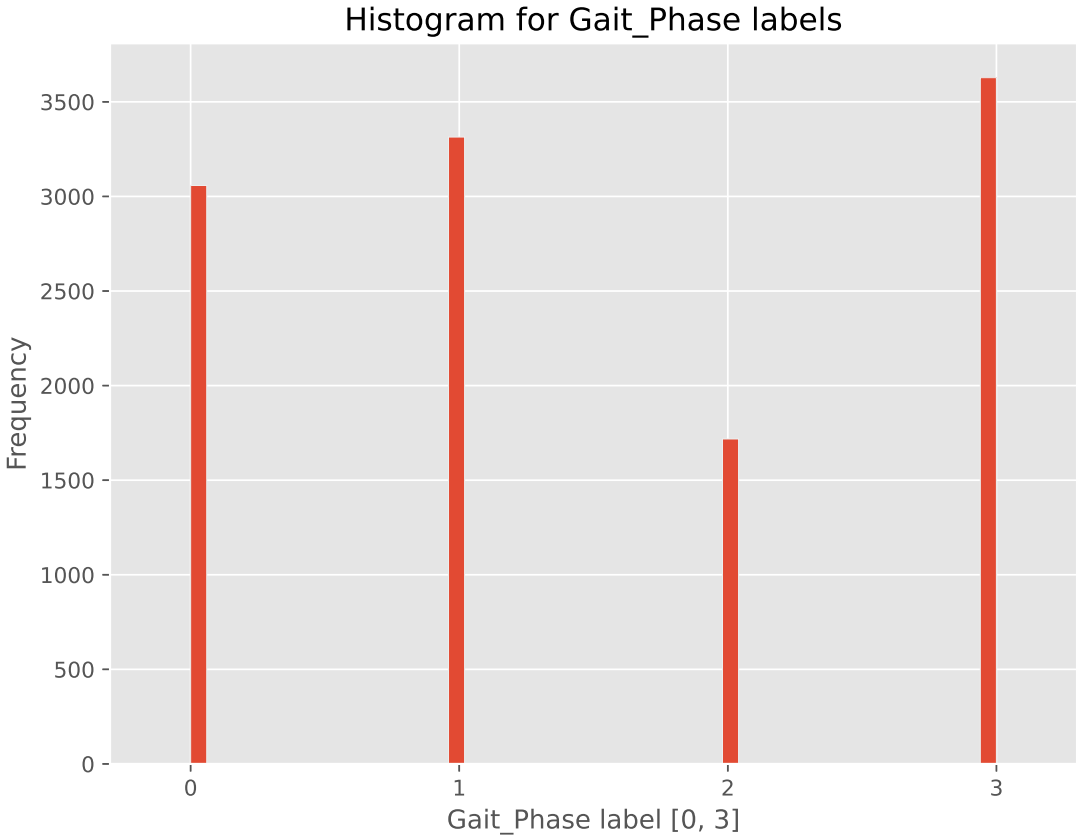
Check 'Gait Phase' to see the distribution of label values

Analysis 3: Looking at the number of instances for each gait phase using histogram

```
labels = 'Gait_Phase'

interval = (0,3) #four classes

addon = 0.1

interval_p = (interval[0]-interval[1]*addon, interval[1]+interval[1]*addon)

data = df_s[labels]

import matplotlib.pyplot as plt
import seaborn as sns

addon = 0.1

interval_p = (interval[0]-interval[1]*addon, interval[1]+interval[1]*addon)

#Dist:


Desi_kde = df_s[labels].plot(kind='kde',
                         title=f'Labeled value: {labels}')
Desi_kde.set_xlabel('Measured_Value')

plt.savefig(f'{labels}_distribution.eps', format='eps')
plt.savefig(f'{labels}_distribution.jpeg', format='jpeg')

plt.show()

plt.close()
```

```
#Hist:

plt.figure(figsize=(8,6))



sns.histplot(data, bins=50)

plt.title(f'Histogram for {labels} labels')
plt.xlabel(f'{labels} label [{interval[0]}, {interval[1]}]')
plt.ylabel('Frequency')

plt.xlim(interval_p)
plt.xticks([0,1,2,3])

plt.savefig(f'{labels}_histogram.eps', format='eps')
plt.savefig(f'{labels}_histogram.jpeg', format='jpeg')

plt.show()


plt.close()
```


Labeled value: Gait_Phase
Density
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−1
0
1
2
3
4
Measured_Value


Histogram for Gait_Phase labels
Frequency
3500
3000
2500
2000
1500
1000
500
0
0
1
2
3
Gait_Phase label [0, 3]

```
Counts_Gphase = df_s['Gait_Phase'].value_counts()

print(Counts_Gphase.to_string())
```

```
Gait_Phase
3    3628
1    3314
0    3058
2    1718
```

```
Counts_Gphase.describe()
```

```
count       4.000000
mean     2929.500000
std       840.632103
min      1718.000000
25%      2723.000000
50%      3186.000000
75%      3392.500000
max      3628.000000
Name: count, dtype: float64
```

```
Counts_Gphase.var()
```

```
706662.3333333334
```

Finding 3: There is huge variance in the count for each gait phase label.

Investigating cyclic nature of data: Number of datapoints per cycle [1,100]

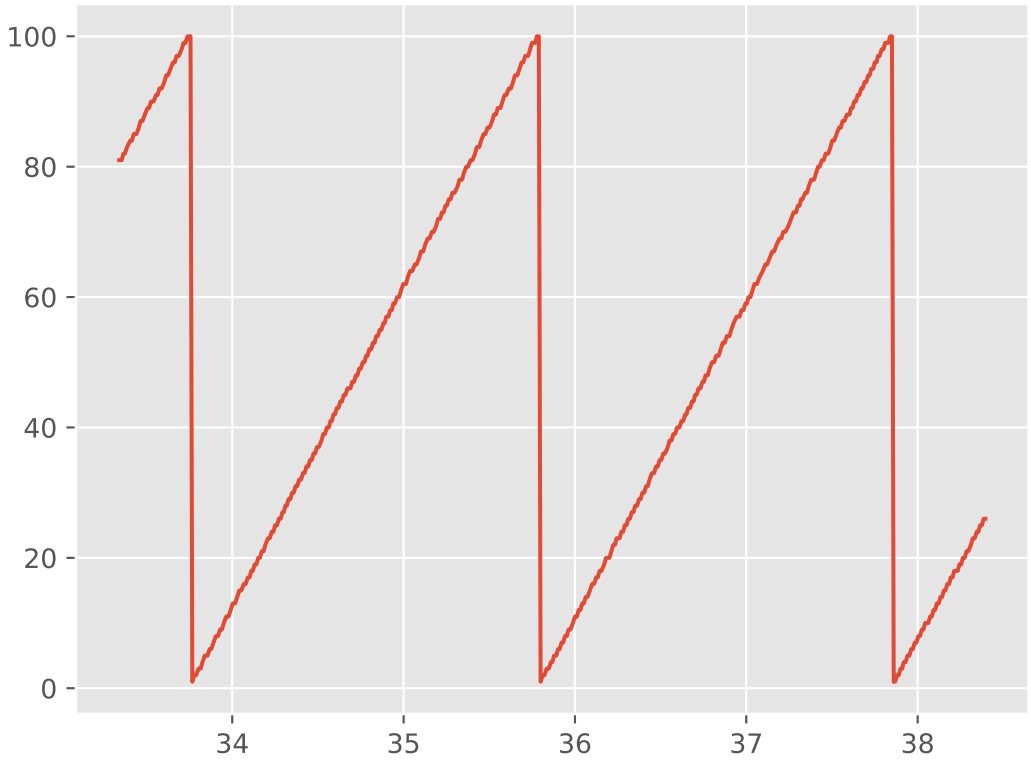
```
plt.plot(df_450['Time_Milli']/1e3, df_450['Percent'])
```

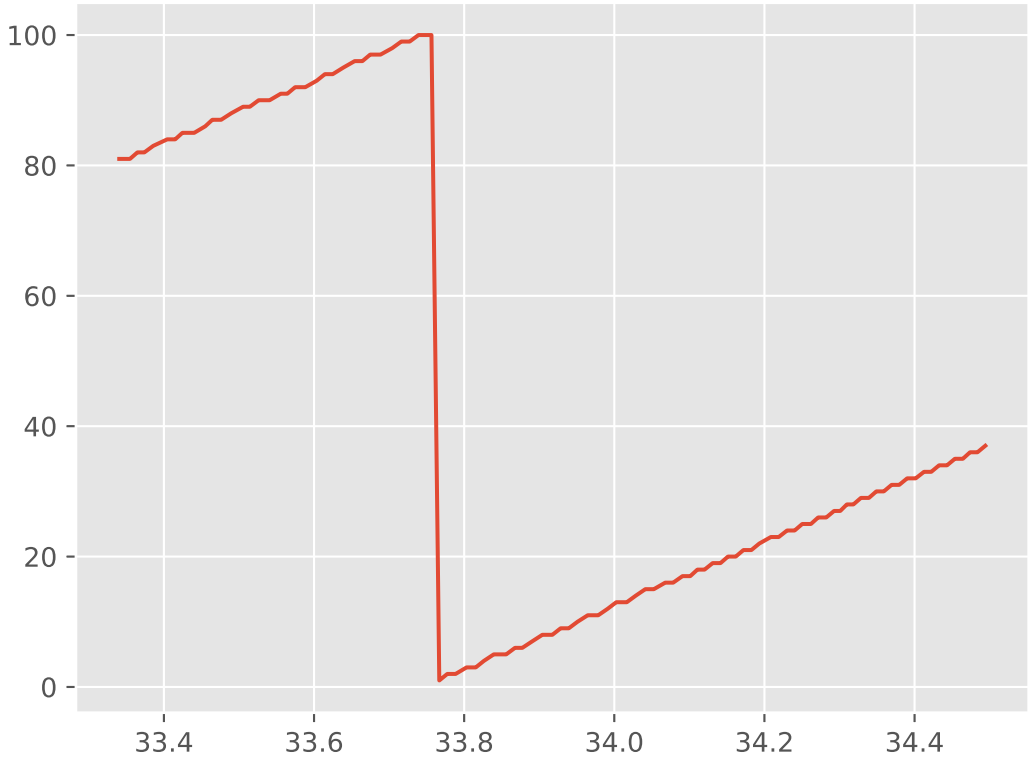
```
[<matplotlib.lines.Line2D at 0x7e0c74e24250>]
```

Data has cyclic nature, and each step seems to take around 2s.

```
plt.plot(df_100['Time_Milli']/1e3, df_100['Percent'])
```

```
[<matplotlib.lines.Line2D at 0x7e0c751e3820>]
```


100
80
60
40
20
0
34
35
36
37
38


100
80
60
40
20
0
33.4
33.6
33.8
34.0
34.2
34.4

The ripple characteristic in the ascending part of the saw-tooth is consistent with finding 1: There is variance in the distribution of label values for Percent label

As expected, findings 1 through 3 indicates that determining the gait phase for a given frame of data is complex. the techniques needed for determining the correct gait phase will most likely go beyond linear relationships with the features or temporal order of the data. The task is considered suitable for an ML algorithm.

Appendix C

Appendix: Topman et. al Dataset

C.1 Data wrangling

Some initial data exploration is needed to ensure the correct handling of any missing/NAN values

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/lim-movement-dataset/leg_test_features.csv
/kaggle/input/lim-movement-dataset/leg_test_raw.csv
/kaggle/input/lim-movement-dataset/leg_train_features.csv
/kaggle/input/lim-movement-dataset/leg_train_raw.csv
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

```
df = pd.read_csv('/kaggle/input/lim-movement-dataset/leg_train_raw.csv')
```

```
df
```

```
        Participant_no  Measure number  Timestamp      q0      q1      q2  \
0                    1               4      0.000  0.2401 -0.3053 -0.7398
1                    1               4      0.012  0.2398 -0.3051 -0.7400
2                    1               4      0.024  0.2396 -0.3049 -0.7401
3                    1               4      0.036  0.2394 -0.3047 -0.7402
4                    1               4      0.048  0.2392 -0.3044 -0.7403
...                ...             ...        ...     ...     ...     ...
102330              36               1     10.332  0.5152 -0.6502  0.3419
102331              36               1     10.344  0.5155 -0.6501  0.3419
102332              36               1     10.356  0.5158 -0.6501  0.3418
```

```
102333              36               1     10.368  0.5160 -0.6502  0.3418
102334              36               1     10.380  0.5163 -0.6502  0.3417

            q3  MotionDeg        Roll      Pitch        Yaw    accX    accY  \
0       0.5492   0.047959  -76.149650  176.82552  162.33590  0.0063 -0.9570
1       0.5493   0.081745  -76.140390  176.85538  162.33032  0.0102 -0.9589
2       0.5493   0.115745  -76.129555  176.85905  162.33324  0.0122 -0.9707
3       0.5493   0.158387  -76.118710  176.86276  162.33615  0.0205 -0.9702
4       0.5494   0.308524  -76.115050  176.85962  162.35720  0.0200 -0.9599
...        ...        ...         ...        ...        ...     ...     ...
102330 -0.4413   4.964061 -103.596620  152.92755  178.48517  0.1655 -0.9848
102331 -0.4411   4.943203 -103.560100  153.00020  178.46739  0.1679 -0.9868
102332 -0.4408   4.923749 -103.530480  153.06064  178.45561  0.1708 -0.9907
102333 -0.4404   4.900333 -103.505930  153.11937  178.50203  0.1762 -0.9990
102334 -0.4400   4.886818 -103.472490  153.19025  178.50578  0.1743 -0.9995

          accZ   gyrX   gyrY   gyrZ    magX    magY    magZ
0      -0.2895 -0.012 -0.074 -0.019  14.999  10.813 -25.637
1      -0.2900 -0.013 -0.070 -0.020  14.999  10.813 -25.637
2      -0.2856 -0.011 -0.067 -0.011  14.999  10.813 -25.637
3      -0.2978 -0.008 -0.062 -0.015  14.999  10.813 -25.637
4      -0.3002 -0.002 -0.062 -0.022  14.999  10.813 -25.637
...        ...    ...    ...    ...     ...     ...     ...
102330 -0.0717  0.017 -0.028  0.033   6.736  -5.987  26.713
102331 -0.0708  0.012 -0.047  0.034   6.736  -5.987  26.713
102332 -0.0708  0.018 -0.061  0.030   6.736  -5.987  26.713
102333 -0.0766  0.005 -0.072  0.034   6.736  -5.987  26.713
102334 -0.0747  0.016 -0.089  0.031   6.736  -5.987  26.713

[102335 rows x 20 columns]
```

```
len(df.loc[df['Participant_no'] == 1])
```

```
2622
```

Check for nan values

```
problematic = df[df.isna().any(axis=1)]
```

```
problematic
```

```
      Participant_no  Measure number  Timestamp      q0      q1      q2  \
8378               3               1      8.892  0.5408 -0.7330 -0.3716
25227              7               1      7.284  0.2023 -0.2469 -0.7720
99395             36               1      8.808  0.3364 -0.4718  0.5391

           q3  MotionDeg       Roll      Pitch        Yaw    accX    accY  \
8378   0.1791        NaN  -89.75357  155.76445  135.64088  0.1137 -0.9663
25227  0.5496        NaN  -72.91930  173.16595  165.78380  0.0361 -0.9697
99395 -0.6111        NaN -100.39845  155.20538  193.77452  0.0053 -1.0517

         accZ   gyrX   gyrY   gyrZ    magX    magY    magZ
8378  -0.2377 -0.170 -0.219 -0.127  22.036  28.799  16.664
25227 -0.3076  0.006  0.048 -0.030   9.286 -13.337 -12.723
99395 -0.0205 -0.103 -0.239 -0.286  28.336  28.963  10.513
```

```
df[df.isna().any(axis=1)]
```

```
       Participant_no  Measure number  Timestamp      q0      q1      q2  \
8378                3               1      8.892  0.5408 -0.7330 -0.3716
25227               7               1      7.284  0.2023 -0.2469 -0.7720
99395              36               1      8.808  0.3364 -0.4718  0.5391

           q3  MotionDeg       Roll      Pitch        Yaw    accX    accY  \
8378   0.1791        NaN  -89.75357  155.76445  135.64088  0.1137 -0.9663
25227  0.5496        NaN  -72.91930  173.16595  165.78380  0.0361 -0.9697
99395 -0.6111        NaN -100.39845  155.20538  193.77452  0.0053 -1.0517

         accZ   gyrX   gyrY   gyrZ    magX    magY    magZ
8378  -0.2377 -0.170 -0.219 -0.127  22.036  28.799  16.664
25227 -0.3076  0.006  0.048 -0.030   9.286 -13.337 -12.723
99395 -0.0205 -0.103 -0.239 -0.286  28.336  28.963  10.513
```

Three nan-value incidents, let's refer to them as j, k and l. Also note the test subject [1,34] and measurment number [1,4]

```
j = 8378 ; s_j = 3; Mn_j = 1,
k = 25227; s_k = 7; Mn_k = 1,
l = 99395; s_l = 36; Mn_l = 1
```

```
start = df[(df['Measure number'] == Mn_j) & (df['Participant_no'] == s_j)].index[0]
end = df[(df['Measure number'] == Mn_j) & (df['Participant_no'] == s_j)].index[-1]
```

```
print('start =', start)
print("end =",end)
```

```
start = 6287
end = 10450
```

```
S = df.columns.get_loc('Participant_no')
T = df.columns.get_loc('Timestamp')
M = df.columns.get_loc('MotionDeg')
```

```
print("Start_new =", "",  df.iloc[end, [S,T,M]])
print("End_old =", "",  df.iloc[end+1, [S,T,M]])
```

```
Start_new =  Participant_no     3.00000
Timestamp          8.04000
MotionDeg         21.42444
Name: 10450, dtype: float64
End_old =  Participant_no    4.000000
Timestamp         0.000000
MotionDeg         0.042772
Name: 10451, dtype: float64
```

End arguments return correxct index. Notice again that Timestamp and MotionDeg resets at nan-value incident.

```
#Display fallout for all three incidents
```

```
Mn = df.columns.get_loc('Measure number')
S = df.columns.get_loc('Participant_no')
MD = df.columns.get_loc('MotionDeg')
T = df.columns.get_loc('Timestamp')
```

```
df.iloc[j-5:j+5, [Mn, S, MD, T]]
```

```
      Measure number  Participant_no  MotionDeg  Timestamp
8373               1               3   6.679461      8.832
8374               1               3   6.784261      8.844
8375               1               3   7.000156      8.856
8376               1               3   7.224145      8.868
8377               1               3   7.359512      8.880
8378               1               3        NaN      8.892
8379               1               3   0.068633      0.000
8380               1               3   0.108082      0.012
8381               1               3   0.115724      0.024
8382               1               3   0.173888      0.036
```

```
df.iloc[k-5:k+13, [Mn, S, MD, T]]
```

```
       Measure number  Participant_no  MotionDeg  Timestamp
25222               1               7   5.170828      7.224
25223               1               7   5.193101      7.236
25224               1               7   5.216119      7.248
25225               1               7   5.238813      7.260
25226               1               7   5.256419      7.272
25227               1               7        NaN      7.284
25228               1               7   0.000004      0.000
25229               1               7   0.000004      0.012
25230               1               7   0.000004      0.024
25231               1               7   0.000004      0.036
25232               1               7   0.000004      0.048
25233               1               7   0.000004      0.060
25234               1               7   0.000004      0.072
25235               1               7   0.000004      0.084
25236               1               7   0.000004      0.096
25237               1               7   0.019635      0.108
25238               1               7   0.034278      0.120
25239               1               7   0.034278      0.132
```

```
df.iloc[l-5:l+5, [Mn, S, MD, T]]
```

```
       Measure number  Participant_no  MotionDeg  Timestamp
99390               1              36   9.997043      8.748
99391               1              36  10.142671      8.760
99392               1              36  10.305382      8.772
99393               1              36  10.487338      8.784
99394               1              36  10.664954      8.796
99395               1              36        NaN      8.808
99396               1              36   0.016182      0.000
99397               1              36   0.016182      0.012
99398               1              36   0.022912      0.024
99399               1              36   0.036238      0.036
```

Nan values detected at three places in dataset. This could be solved using interpolate.

Timestamp(2) and MotionDeg(7) seems to reset at nan-value incidents. To fix this issue, one can possibly use cumsum pandas function.

Defining function to fix all three nan-instances as well as the reset of Timestamp(2) and MotionDeg(7).

Making a note that each subject has mulitple measurements throughout the dataset, only do accumulative sum until either Participant_no or Measure number changes.

```
'''
j = 8378 ; s_j = 3; Mn_j = 1,
k = 25227; s_k = 7; Mn_k = 1,
l = 99395; s_l = 36; Mn_l = 1
'''


# Need Participant_no, row index for nan-incident (j,k,l),
# Measure number
# end index for subject recording,
# column index Timestamp = T
# column index Motiondeg = M

# Participant = subject = s ,incident_row_index = x , Measure number = M_n

def fixnan(s, x, Mn, T, M):

    end = df[(df['Measure number'] == Mn) & (df['Participant_no'] == s)].index[-1]


    #for Timestamp and MotionDeg fix acumulative sum throughout array

    df.iloc[x:end+1 , T] = df.iloc[x:end+1 , T].cumsum()

    # Motiondeg has nan value, start accumulative at x-1

    #df.iloc[x-1:end+1 , M] = df.iloc[x-1:end+1 , M].cumsum()

    # Timestamp: x + 1 = zero due to reset, interpolate x and x+2

    df.iloc[x+1, T] = ( df.iloc[x, T] + df.iloc[x+2, T] )  / 2

    # MotionDeg: Reset count

    df.iloc[x, M] = 0



T = 2
M = 7

fixnan(s_j, j, Mn_j, T, M)
fixnan(s_k, k, Mn_k, T, M)
fixnan(s_l, l, Mn_l, T, M)

'''
j = 8378 ; s_j = 3; Mn_j = 1,
k = 25227; s_k = 7; Mn_k = 1,
l = 99395; s_l = 36; Mn_l = 1
'''

'\nj = 8378 ; s_j = 3; Mn_j = 1,\nk = 25227; s_k = 7; Mn_k = 1,\nl = 99395; s_l = 36; Mn_l =

df.iloc[j-5:j+5, [Mn, S, MD, T]]

      Measure number  Participant_no  MotionDeg  Timestamp
```

```
8373                1                3   6.679461      8.832
8374                1                3   6.784261      8.844
8375                1                3   7.000156      8.856
8376                1                3   7.224145      8.868
8377                1                3   7.359512      8.880
8378                1                3   0.000000      8.892
8379                1                3   0.068633      8.898
8380                1                3   0.108082      8.904
8381                1                3   0.115724      8.928
8382                1                3   0.173888      8.964
```

```
df.iloc[k-5:k+13, [Mn, S, MD, T]]
```

```
       Measure number  Participant_no  MotionDeg  Timestamp
25222                1               7   5.170828      7.224
25223                1               7   5.193101      7.236
25224                1               7   5.216119      7.248
25225                1               7   5.238813      7.260
25226                1               7   5.256419      7.272
25227                1               7   0.000000      7.284
25228                1               7   0.000004      7.290
25229                1               7   0.000004      7.296
25230                1               7   0.000004      7.320
25231                1               7   0.000004      7.356
25232                1               7   0.000004      7.404
25233                1               7   0.000004      7.464
25234                1               7   0.000004      7.536
25235                1               7   0.000004      7.620
25236                1               7   0.000004      7.716
25237                1               7   0.019635      7.824
25238                1               7   0.034278      7.944
25239                1               7   0.034278      8.076
```

```
df.iloc[l-5:l+5, [Mn, S, MD, T]]
```

```
       Measure number  Participant_no  MotionDeg  Timestamp
99390                1              36   9.997043      8.748
99391                1              36  10.142671      8.760
99392                1              36  10.305382      8.772
99393                1              36  10.487338      8.784
99394                1              36  10.664954      8.796
99395                1              36   0.000000      8.808
99396                1              36   0.016182      8.814
99397                1              36   0.016182      8.820
99398                1              36   0.022912      8.844
99399                1              36   0.036238      8.880
```

Tis fix isn't ideal, as the data is somewhat skewed by the artificial interpolation. However, we will consider this acceptable due to only three instances in this fairly large dataset.

```
df.to_csv('leg_train_raw_clean.csv',index=False)
```

C.2 Running Gait Event labeling algorithm for Gait Detection on the TopMan dataset

```python
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/lim-movement-dataset/leg_test_features.csv
/kaggle/input/lim-movement-dataset/leg_test_raw_labeled_4_11_complete.csv
/kaggle/input/lim-movement-dataset/TopMan_complete_ID_DetLabel_Ready_12_12.parquet
/kaggle/input/lim-movement-dataset/df_TopMan_635_steps__Detection_and_ID_Labeled_Original_Index.csv
/kaggle/input/lim-movement-dataset/leg_train_raw_unlabeled.csv
/kaggle/input/lim-movement-dataset/leg_train_raw_labeled_4_11_complete.csv
/kaggle/input/lim-movement-dataset/df_GE_Tags_13_12.csv
/kaggle/input/lim-movement-dataset/TopMan_ML_ID_0_29.csv
/kaggle/input/lim-movement-dataset/TopMan_complete_ID_DetLabel_Ready_8_12.csv
/kaggle/input/lim-movement-dataset/leg_test_raw.csv
/kaggle/input/lim-movement-dataset/leg_train_raw_clean.csv
/kaggle/input/lim-movement-dataset/Leg_Raw_Full_W_Engineered_Feats_26_11.csv
/kaggle/input/lim-movement-dataset/leg_Gait_Detection_complete_raw_labeled_20_11.csv
/kaggle/input/lim-movement-dataset/leg_train_features.csv
/kaggle/input/lim-movement-dataset/TopMan_complete_12_8_ID_Det_Ready_12_8.csv
/kaggle/input/lim-movement-dataset/leg_train_raw_labeled_12_11_complete.csv
/kaggle/input/lim-movement-dataset/TopMan_635_steps.csv
```

```
/kaggle/input/lim-movement-dataset/df_TopMan_635_steps__Detection_and_ID_Labeled_Reset_Index.csv
/kaggle/input/lim-movement-dataset/leg_train_raw.csv
```

```
pd.set_option('display.max_columns', None)
```

```
df = pd.read_parquet('/kaggle/input/lim-movement-dataset/TopMan_complete_ID_DetLabel_Ready_12_12.parquet')
```

```
df.dtypes
```

```
Gait_Identification      int16
Gait_Events              int64
Measure number           int16
Timestamp              float64
q0                     float32
q1                     float32
q2                     float32
q3                     float32
MotionDeg              float32
Roll                   float32
Pitch                  float32
Yaw                    float32
accX                   float32
accY                   float32
accZ                   float32
gyrX                   float32
gyrY                   float32
gyrZ                   float32
magX                   float32
magY                   float32
magZ                   float32
Detection_50             int64
dtype: object
```

```
df.fillna(0, inplace=True)
```

```
df
```

```
        Gait_Identification  Gait_Events  Measure number     Timestamp  \
```

```
0                      0           0              1      0.000000
1                      0           0              1      0.012000
2                      0           0              1      0.024000
3                      0           0              1      0.036000
4                      0           0              1      0.048000
...                  ...         ...            ...           ...
147152                29           0              1  13215.228067
147153                29           0              1  13225.572067
147154                29           0              1  13235.928067
147155                29           0              1  13246.296067
147156                29           0              1  13256.676067

            q0      q1      q2      q3  MotionDeg        Roll       Pitch  \
0       0.5605 -0.7824 -0.2432  0.1200   0.228584  -99.915062  165.284698
1       0.5601 -0.7831 -0.2429  0.1182   0.521143  -99.903725  164.881226
2       0.5596 -0.7839 -0.2423  0.1159   0.788835  -99.901802  164.399826
3       0.5591 -0.7847 -0.2419  0.1138   0.983235  -99.892815  163.932587
4       0.5587 -0.7852 -0.2419  0.1122   1.158354  -99.853317  163.511810
...        ...     ...     ...     ...        ...         ...         ...
147152  0.5152 -0.6502  0.3419 -0.4413   4.964061 -103.596619  152.927551
147153  0.5155 -0.6501  0.3419 -0.4411   4.943203 -103.560097  153.000198
147154  0.5158 -0.6501  0.3418 -0.4408   4.923748 -103.530479  153.060638
147155  0.5160 -0.6502  0.3418 -0.4404   4.900333 -103.505928  153.119370
147156  0.5163 -0.6502  0.3417 -0.4400   4.886818 -103.472488  153.190247

                Yaw    accX    accY    accZ   gyrX   gyrY   gyrZ    magX  \
0       143.521698  0.2021 -0.8843 -0.3394 -0.158  0.101 -0.017  30.000
1       143.192734  0.1967 -0.8867 -0.3228 -0.217  0.311 -0.057  30.000
2       142.817688  0.1860 -0.9210 -0.3300 -0.287  0.414 -0.086  30.000
3       142.439377  0.1674 -0.9470 -0.3413 -0.276  0.367 -0.101  30.000
4       142.103729  0.1699 -0.9478 -0.3438 -0.202  0.255 -0.116  30.000
...            ...     ...     ...     ...    ...    ...    ...     ...
147152  178.485168  0.1655 -0.9850 -0.0717  0.017 -0.028  0.033   6.734
147153  178.467392  0.1678 -0.9870 -0.0708  0.012 -0.047  0.034   6.734
```

```
147154  178.455612  0.1708 -0.9907 -0.0708  0.018 -0.061  0.030   6.734
147155  178.502029  0.1761 -0.9990 -0.0766  0.005 -0.072  0.034   6.734
147156  178.505783  0.1743 -0.9995 -0.0747  0.016 -0.089  0.031   6.734

         magY       magZ  Detection_50
0      -1.983 -29.299999             0
1      -1.983 -29.299999             0
2      -1.983 -29.299999             0
3      -1.983 -29.299999             0
4      -1.983 -29.299999             0
...       ...        ...           ...
147152 -5.990  26.719999             0
147153 -5.990  26.719999             0
147154 -5.990  26.719999             0
147155 -5.990  26.719999             0
147156 -5.990  26.719999             0

[147157 rows x 22 columns]
```



```
list(df.columns)
```

```
['Gait_Identification',
 'Gait_Events',
 'Measure number',
 'Timestamp',
 'q0',
 'q1',
 'q2',
 'q3',
 'MotionDeg',
 'Roll',
 'Pitch',
 'Yaw',
 'accX',
 'accY',
```

```
 'accZ',
 'gyrX',
 'gyrY',
 'gyrZ',
 'magX',
 'magY',
 'magZ',
 'Detection_50']

df = df[['Gait_Identification',
         'Gait_Events',
 'Measure number',
 'Timestamp',
 'q0',
 'q1',
 'q2',
 'q3',
 'MotionDeg',
 'Roll',
 'Pitch',
 'Yaw',
 'accX',
 'accY',
 'accZ',
 'gyrX',
 'gyrY',
 'gyrZ',
 'magX',
 'magY',
 'magZ'
 ]]

Cols = ['q0',
 'q1',
 'q2',
```

```
 'q3',
 'MotionDeg',
 'Roll',
 'Pitch',
 'Yaw',
 'accX',
 'accY',
 'accZ',
 'gyrX',
 'gyrY',
 'gyrZ',
 'magX',
 'magY',
 'magZ']
```

```
df[Cols] = df[Cols].astype('float32')
```

```
/tmp/ipykernel_361/4171098590.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[Cols] = df[Cols].astype('float32')
```

```
df.dtypes
```

```
Gait_Identification      int16
Gait_Events              int64
Measure number           int16
Timestamp              float64
q0                     float32
q1                     float32
q2                     float32
q3                     float32
MotionDeg              float32
```

```
Roll                    float32
Pitch                   float32
Yaw                     float32
accX                    float32
accY                    float32
accZ                    float32
gyrX                    float32
gyrY                    float32
gyrZ                    float32
magX                    float32
magY                    float32
magZ                    float32
dtype: object
```

```
df
```

```
        Gait_Identification  Gait_Events  Measure number     Timestamp  \
0                         0            0               1      0.000000
1                         0            0               1      0.012000
2                         0            0               1      0.024000
3                         0            0               1      0.036000
4                         0            0               1      0.048000
...                     ...          ...             ...           ...
147152                   29            0               1  13215.228067
147153                   29            0               1  13225.572067
147154                   29            0               1  13235.928067
147155                   29            0               1  13246.296067
147156                   29            0               1  13256.676067

            q0      q1      q2      q3  MotionDeg        Roll       Pitch  \
0       0.5605 -0.7824 -0.2432  0.1200   0.228584  -99.915062  165.284698
1       0.5601 -0.7831 -0.2429  0.1182   0.521143  -99.903725  164.881226
2       0.5596 -0.7839 -0.2423  0.1159   0.788835  -99.901802  164.399826
3       0.5591 -0.7847 -0.2419  0.1138   0.983235  -99.892815  163.932587
4       0.5587 -0.7852 -0.2419  0.1122   1.158354  -99.853317  163.511810
```

```
...        ...     ...     ...     ...        ...          ...          ...
147152  0.5152 -0.6502  0.3419 -0.4413   4.964061 -103.596619  152.927551
147153  0.5155 -0.6501  0.3419 -0.4411   4.943203 -103.560097  153.000198
147154  0.5158 -0.6501  0.3418 -0.4408   4.923748 -103.530479  153.060638
147155  0.5160 -0.6502  0.3418 -0.4404   4.900333 -103.505928  153.119370
147156  0.5163 -0.6502  0.3417 -0.4400   4.886818 -103.472488  153.190247

                Yaw    accX    accY    accZ   gyrX   gyrY   gyrZ    magX  \
0        143.521698  0.2021 -0.8843 -0.3394 -0.158  0.101 -0.017  30.000
1        143.192734  0.1967 -0.8867 -0.3228 -0.217  0.311 -0.057  30.000
2        142.817688  0.1860 -0.9210 -0.3300 -0.287  0.414 -0.086  30.000
3        142.439377  0.1674 -0.9470 -0.3413 -0.276  0.367 -0.101  30.000
4        142.103729  0.1699 -0.9478 -0.3438 -0.202  0.255 -0.116  30.000
...             ...     ...     ...     ...    ...    ...    ...     ...
147152   178.485168  0.1655 -0.9850 -0.0717  0.017 -0.028  0.033   6.734
147153   178.467392  0.1678 -0.9870 -0.0708  0.012 -0.047  0.034   6.734
147154   178.455612  0.1708 -0.9907 -0.0708  0.018 -0.061  0.030   6.734
147155   178.502029  0.1761 -0.9990 -0.0766  0.005 -0.072  0.034   6.734
147156   178.505783  0.1743 -0.9995 -0.0747  0.016 -0.089  0.031   6.734

          magY       magZ
0       -1.983 -29.299999
1       -1.983 -29.299999
2       -1.983 -29.299999
3       -1.983 -29.299999
4       -1.983 -29.299999
...        ...        ...
147152 -5.990  26.719999
147153 -5.990  26.719999
147154 -5.990  26.719999
147155 -5.990  26.719999
147156 -5.990  26.719999

[147157 rows x 21 columns]
```

Gait_Events holds objects for gait phase detection

Making sure column 'Gait_Events' can hold both numbers and strings

```
df['Gait_Events'] = 0
```

```
df['Gait_Events'] = df['Gait_Events'].astype('object')
```

```
/tmp/ipykernel_361/1512221221.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Gait_Events'] = 0
/tmp/ipykernel_361/1512221221.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Gait_Events'] = df['Gait_Events'].astype('object')
```

df['Gait_Identification'] = df['Gait_Identification'].astype('int16')

df['Measure number'] = df['Measure number'].astype('int16')

```
df.dtypes
```

```
Gait_Identification      int16
Gait_Events             object
Measure number           int16
Timestamp              float64
q0                     float32
q1                     float32
q2                     float32
q3                     float32
MotionDeg              float32
Roll                   float32
Pitch                  float32
```

```
Yaw                     float32
accX                    float32
accY                    float32
accZ                    float32
gyrX                    float32
gyrY                    float32
gyrZ                    float32
magX                    float32
magY                    float32
magZ                    float32
dtype: object
```

df.to_csv('TopMan_complete_ID_DetLabel_Ready_8_12.csv')

df.to_parquet('TopMan_complete_ID_DetLabel_Ready_12_12.parquet')

```
import matplotlib.pyplot as plt
```

```
list(df.columns)
```

```
['Gait_Identification',
 'Gait_Events',
 'Measure number',
 'Timestamp',
 'q0',
 'q1',
 'q2',
 'q3',
 'MotionDeg',
 'Roll',
 'Pitch',
 'Yaw',
 'accX',
 'accY',
 'accZ',
 'gyrX',
 'gyrY',
```

```
 'gyrZ',
 'magX',
 'magY',
 'magZ']

df['Detection_50'] = 0

df = df[['Gait_Identification',
 'Gait_Events',
 'Measure number',
 'Timestamp',
 'q0',
 'q1',
 'q2',
 'q3',
 'MotionDeg',
 'Roll',
 'Pitch',
 'Yaw',
 'accX',
 'accY',
 'accZ',
 'gyrX',
 'gyrY',
 'gyrZ',
 'magX',
 'magY',
 'magZ',
 'Detection_50',]]

df_Labeling_Copy = df.copy()

#df_Labeling_Copy['Detection_50'] = 0

df_Labeling_Copy

        Gait_Identification Gait_Events  Measure number     Timestamp      q0  \
```

```
0                      0          0              1      0.000000  0.5605
1                      0          0              1      0.012000  0.5601
2                      0          0              1      0.024000  0.5596
3                      0          0              1      0.036000  0.5591
4                      0          0              1      0.048000  0.5587
...                  ...        ...            ...           ...     ...
147152                29          0              1  13215.228067  0.5152
147153                29          0              1  13225.572067  0.5155
147154                29          0              1  13235.928067  0.5158
147155                29          0              1  13246.296067  0.5160
147156                29          0              1  13256.676067  0.5163

            q1      q2      q3  MotionDeg        Roll       Pitch         Yaw  \
0      -0.7824 -0.2432  0.1200   0.228584  -99.915062  165.284698  143.521698
1      -0.7831 -0.2429  0.1182   0.521143  -99.903725  164.881226  143.192734
2      -0.7839 -0.2423  0.1159   0.788835  -99.901802  164.399826  142.817688
3      -0.7847 -0.2419  0.1138   0.983235  -99.892815  163.932587  142.439377
4      -0.7852 -0.2419  0.1122   1.158354  -99.853317  163.511810  142.103729
...        ...     ...     ...        ...         ...         ...         ...
147152 -0.6502  0.3419 -0.4413   4.964061 -103.596619  152.927551  178.485168
147153 -0.6501  0.3419 -0.4411   4.943203 -103.560097  153.000198  178.467392
147154 -0.6501  0.3418 -0.4408   4.923748 -103.530479  153.060638  178.455612
147155 -0.6502  0.3418 -0.4404   4.900333 -103.505928  153.119370  178.502029
147156 -0.6502  0.3417 -0.4400   4.886818 -103.472488  153.190247  178.505783

          accX    accY    accZ   gyrX   gyrY   gyrZ    magX   magY       magZ  \
0       0.2021 -0.8843 -0.3394 -0.158  0.101 -0.017  30.000 -1.983 -29.299999
1       0.1967 -0.8867 -0.3228 -0.217  0.311 -0.057  30.000 -1.983 -29.299999
2       0.1860 -0.9210 -0.3300 -0.287  0.414 -0.086  30.000 -1.983 -29.299999
3       0.1674 -0.9470 -0.3413 -0.276  0.367 -0.101  30.000 -1.983 -29.299999
4       0.1699 -0.9478 -0.3438 -0.202  0.255 -0.116  30.000 -1.983 -29.299999
...        ...     ...     ...    ...    ...    ...     ...    ...        ...
147152  0.1655 -0.9850 -0.0717  0.017 -0.028  0.033   6.734 -5.990  26.719999
147153  0.1678 -0.9870 -0.0708  0.012 -0.047  0.034   6.734 -5.990  26.719999
```

```
147154  0.1708 -0.9907 -0.0708  0.018 -0.061  0.030   6.734 -5.990  26.719999
147155  0.1761 -0.9990 -0.0766  0.005 -0.072  0.034   6.734 -5.990  26.719999
147156  0.1743 -0.9995 -0.0747  0.016 -0.089  0.031   6.734 -5.990  26.719999

        Detection_50
0                  0
1                  0
2                  0
3                  0
4                  0
...              ...
147152             0
147153             0
147154             0
147155             0
147156             0

[147157 rows x 22 columns]
```

'''

Tagging_Run_13_12

'''

import matplotlib.pyplot as plt import numpy as np from scipy.signal import butter, filtfilt, argrelextrema

def butter_lowpass(data, cutoff_freq, fs, order=4): nyquist = 0.5 * fs normal_cutoff = cutoff_freq / nyquist b, a = butter(order, normal_cutoff, btype='low', analog=False) y = filtfilt(b, a, data, axis=0) return y

def normalize_window(window): # Normalize each column (feature) independently normalized_window = (window - window.mean(axis=0)) / window.std(axis=0) return normalized_window

def detect_local_minima(data, order): minima_indices = argrelextrema(data, np.less_equal, order=order)[0] return minima_indices

def detect_local_maxima(data, order): maxima_indices = argrelextrema(data, np.greater_equal, order=order)[0] return maxima_indices

def plot_around_event(event_type, event_pos_in_df, zoom, features, df=df): # Determine start and end indices based on the zoom and event position start = int(event_pos_in_df - zoom) end = int(event_pos_in_df + zoom)

```
# Extract the relevant portion of the DataFrame
df_plot = df.loc[start:end, :]

# Extract the subject, for use in plot name
# column name is in 'Gait_Identification'

name = df.loc[start, 'Gait_Identification']


# Create a figure and primary axis
fig, ax1 = plt.subplots(figsize=(10, 6))

# Plot each feature on the primary axis with twinx for secondary and tertiary axes
for i, feature in enumerate(features):
    ax = ax1.twinx() if i > 0 else ax1
    color = ['blue', 'orange', 'green'][i]
    ax.plot(df_plot[feature], label=feature, color=color)
    ax.set_ylabel(feature, color=color)
    ax.tick_params(axis='y', labelcolor=color)

    if i == 2:  # Make some space for y_ax for the third feature (accX)
        ax.spines['right'].set_position(('outward', 60))

# Add a vertical line at the event position
ax1.axvline(x=event_pos_in_df, color='red', linestyle='--', label=f'{event_type}')

# Add a title and legend based on the event type
if event_type == 'TO':
    plt.title(f'Original DataFrame around Toe Off, subject{name}')
elif event_type == 'HS':
    plt.title(f'Original DataFrame around Heel-strike, subject{name}')
```

```
fig.legend(loc='upper right', bbox_to_anchor=(0.85, 0.85))

plt.savefig(f'{event_type}_subject_{name}_pos_{start}_{end}_in_dataframe', format='eps')         #fstrings are nice


plt.shodef plot_around_event(event_type, event_pos_in_df, zoom, features, df=df):
# Determine start and end indices based on the zoom and event position
start = int(event_pos_in_df - zoom)
end = int(event_pos_in_df + zoom)

# Extract the relevant portion of the DataFrame
df_plot = df.loc[start:end, :]

# Extract the subject, for use in plot name
# column name is in 'Gait_Identification'

name = df.loc[start, 'Gait_Identification']


# Create a figure and primary axis
fig, ax1 = plt.subplots(figsize=(10, 6))

# Plot each feature on the primary axis with twinx for secondary and tertiary axes
for i, feature in enumerate(features):
    ax = ax1.twinx() if i > 0 else ax1
    color = ['blue', 'orange', 'green'][i]
    ax.plot(df_plot[feature], label=feature, color=color)
    ax.set_ylabel(feature, color=color)
    ax.tick_params(axis='y', labelcolor=color)

    if i == 2:  # Make some space for y_ax for the third feature (accX)
        ax.spines['right'].set_position(('outward', 60))

# Add a vertical line at the event position
```

```
ax1.axvline(x=event_pos_in_df, color='red', linestyle='--', label=f'{event_type}')

# Add a title and legend based on the event type
if event_type == 'TO':
    plt.title(f'Original DataFrame around Toe Off at {start}:{end} in df, subject {name}')
elif event_type == 'HS':
    plt.title(f'Original DataFrame around Heel-strike at {start}:{end} in df, subject {name}')

fig.legend(loc='upper right', bbox_to_anchor=(0.85, 0.85))

plt.savefig(f'{event_type}_pos_{start}_{end}_subject_{name}_in_dataframe', format='eps') #f strings are nice


plt.show()w()
```

def detect_heel_strike(Debug, Index_Loc_Max_Pitch, windowed_maxima_indices_dict, non_W_maxima_indices_dict, windowed_minima_indices_dict, non_W_minima_indices_dict, start_index_subject, start_index_window, HS_lower, HS_upper_Max, HS_upper_Min, HS_Impact_Search_Lower, HS_Impact_Search_Upper, W_zoom_lower, W_zoom_upper, df_Labeling_Copy, df): if Debug == True:

```
    print('Running HS detection')

Event_Detected = False



Index_Loc_Min_accX = [index for index in windowed_minima_indices_dict['accX'] if
                      Index_Loc_Max_Pitch - HS_lower <= index <= Index_Loc_Max_Pitch + HS_upper_Min]


if not any(Index_Loc_Min_accX):
    # If no candidates in windowed_maxima_indices_dict['accX'], check non_W_maxima_indices_dict['accX']
    Index_Loc_Min_accX_nonW = [index for index in non_W_minima_indices_dict['accX'] if
                               Index_Loc_Max_Pitch - HS_lower <= index <= Index_Loc_Max_Pitch + HS_upper_Min]

    if any(Index_Loc_Min_accX_nonW):
```

```
        Window_Index_Loc_Min_accX = Index_Loc_Min_accX_nonW[0]

        if Debug == True:

            print('accX MIN component of HS found inside nonW:', Window_Index_Loc_Min_accX)
    else:
        if Debug == True:
            print('Not able to find local min accX near Pitch Max')
        return None

else:
    Window_Index_Loc_Min_accX = Index_Loc_Min_accX[0]
    if Debug == True:
        print('accX MIN component of HS found inside WIN:', Window_Index_Loc_Min_accX)


if Window_Index_Loc_Min_accX is not None:

    # Check for local maxima of accX within a range around Pitch maxima
    Index_Loc_Max_accX = [index for index in windowed_maxima_indices_dict['accX'] if
                      Window_Index_Loc_Min_accX <= index <= Index_Loc_Max_Pitch + HS_upper_Max]

    if not any(Index_Loc_Max_accX):
        # If no candidates in windowed_maxima_indices_dict['accX'], check non_W_maxima_indices_dict['accX']
        Index_Loc_Max_accX_nonW = [index for index in non_W_maxima_indices_dict['accX'] if
                                   Index_Loc_Max_Pitch <= index <= Index_Loc_Max_Pitch + HS_upper_Max]

        if any(Index_Loc_Max_accX_nonW):
            Window_Index_Loc_Max_accX = Index_Loc_Max_accX_nonW[0]

            if Debug == True:

                print('accX MAX component of HS found inside nonW:', Window_Index_Loc_Max_accX)
        else:
```

```
            if Debug == True:
                print('Not able to find local max accX near accX min')
            return None

    else:
        Window_Index_Loc_Max_accX = Index_Loc_Max_accX[0]

        if Debug == True:
            print('accX MAX component of HS found inside WIN:', Window_Index_Loc_Max_accX)


if Window_Index_Loc_Max_accX is not None and Window_Index_Loc_Min_accX is not None:


    if Debug == True:
        print('Found both min and max for accX component near max for pitch. Probable HS. Search for accZ min.')


    Cand_Loc_Min_accZ = [index for index in windowed_minima_indices_dict['accZ'] if
                    Window_Index_Loc_Max_accX - HS_Impact_Search_Lower <= index <= Window_Index_Loc_Max_accX + HS_Impact_Search_Upper]

    if Debug == True:

        print('these are accZ within range of accX max:', Cand_Loc_Min_accZ)

    if any(Cand_Loc_Min_accZ):

        if Debug == True:

            print('Found candidates for accZ: ', Cand_Loc_Min_accZ)

        Cand_Loc_Min_accZ = Cand_Loc_Min_accZ[0]

        if Debug == True:
```

```
            print('Heel Strike at w index: ', Cand_Loc_Min_accZ)
            print('Heel Strike at Index: ', Cand_Loc_Min_accZ + start_index_subject + start_index_window)

        Heel_strike = Cand_Loc_Min_accZ + start_index_subject + start_index_window

        if not any(df_Labeling_Copy.loc[max(Heel_strike - 20, 0):min(Heel_strike + 20, len(df_Labeling_Copy) - 1), 'Gait_Events'] == 'HS')

            #Condition stops search from going out of bounds near start/end of labeling_df


                Zoom = 50

                #plot_around_event('HS', Heel_strike, zoom=Zoom, features=['Pitch', 'accZ', 'accX'], df=df)



                df_Labeling_Copy.loc[Heel_strike, 'Gait_Events'] = 'HS'

                Event_Detected = True

        else:
            if Debug == True:

                print("Heel Strike already exists within ±20 positions; skip tag.")

    else:
        if Debug == True:

            print('Not able to find local min accZ near Pitch Max')


    if Event_Detected:
```

```
        return True

return False
```

def detect_toe_off(Debug, Index_Loc_Min_Pitch, W_zoom_lower, W_zoom_upper, TO_lower, TO_upper, start_index_subject, start_index_window, windowed_maxima_indices_dict, non_W_minima_indices_dict, df_Labeling_Copy, Event_Tagged, df, Zoom=50):

```
if Debug == True:
    print('Running Toe_Off detection')

Event_Tagged = False

if any(Index_Loc_Min_Pitch - TO_lower < index < Index_Loc_Min_Pitch + TO_upper for index in
       windowed_maxima_indices_dict['accZ']):

    matching_indices = np.where(
        (Index_Loc_Min_Pitch - TO_lower < windowed_maxima_indices_dict['accZ']) &
        (windowed_maxima_indices_dict['accZ'] < Index_Loc_Min_Pitch + TO_upper))[0]

    Index_Loc_Max_W_accZ = windowed_maxima_indices_dict['accZ'][matching_indices[0]]

    if Debug == True:

        print('all instances', windowed_maxima_indices_dict['accZ'])
        print('index in list that meets criteria: ', matching_indices)
        print('Toe off at: ',
              Index_Loc_Max_W_accZ,
              'convert to global coordinates:',
              Index_Loc_Max_W_accZ + start_index_subject + start_index_window)

    Toe_Off = Index_Loc_Max_W_accZ + start_index_subject + start_index_window

    Event_Tagged = True

    if Debug == True:
```

```
        print('toe off at: ', Toe_Off)

    if not any(df_Labeling_Copy.loc[max(Toe_Off - 20, 0):min(Toe_Off + 20, len(df_Labeling_Copy) - 1), 'Gait_Events'] == 'TO'):
        #Condition stops search from going out of bounds near start/end of labeling_df

        df_Labeling_Copy.loc[Toe_Off, 'Gait_Events'] = 'TO'

        #plot_around_event('TO', Toe_Off, zoom=Zoom, features=['Pitch', 'accZ', 'accX'], df=df)

    else:
        if Debug == True:

            print("Toe_off already exists within ±20 positions; skip tag.")

else:
    if Debug == True:

        print(f'Toe_Off_Detection: Could not find accZ MAX in range [ {Index_Loc_Min_Pitch - TO_lower} , {Index_Loc_Min_Pitch + TO_upper}

if Event_Tagged:

    return True

return False
```

def windowed_subject_data_plotter(df, df_Labeling_Copy, subject_column, subject_number, Debug=True, features=None, window_size=100, overlap=50, cutoff_freq=None, fs=None, order=None, non_windowed_features=None, NWF_minmax_orders=None, minmax_windowed_features=None):

```
# Filter the DataFrame based on the specified subject and subject column
subject_data = df[df[subject_column] == subject_number]

start_index_subject = subject_data.index[0]

# Extract the selected features
```

```
selected_data = subject_data[features]

# Apply Hamming windowing and Butterworth lowpass filtering to the data
hamming_window = np.hamming(window_size)[:, None]  # Reshape to column vector
windows = [
    butter_lowpass(selected_data[i:i+window_size].values * hamming_window, cutoff_freq, fs=fs, order=order)
    for i in range(0, len(selected_data)-window_size+1, overlap)
]

# Normalize each window
normalized_windows = [normalize_window(window) for window in windows]

windowed_minima_indices_dict = {} #re-initialized for every window
windowed_maxima_indices_dict = {} #re-initialized for every window

Continue_Pitch_TH = False

# Loop over non_windowed_features and NWF_minmax_orders simultaneously
for j, (window, unfiltered_window) in enumerate(zip(normalized_windows, windows)):



    '''
    Event_Tagged is used to skip to next window as soon as either HS or TO is tagged
    This mediates the problem of processing events to near the edge of the window,
    where the characteristics of the measurands are dirstorted.

    '''

    Event_Tagged = False
```

```
start_index_window = j * (window_size - overlap)

'''
Remove all output to save ram

'''

'''

fig, axs = plt.subplots(len(features) + len(non_windowed_features), 1, figsize=(10, 2*(len(features) + len(non_windowed_features))))
'''


# Plot the normalized and filtered windowed data in separate plots for each feature and window
for i, feature in enumerate(features):


    #axs[i].plot(range(len(window)), window[:, i], label=f'{feature} (Filtered and Normalized)')

    min_order, max_order = minmax_windowed_features[i] # fetch orders from function call list minmax_windowed_features=[[8,8],[35,9]]

    minima_indices = detect_local_minima(window[:, i], min_order)
    minima_indices_original = minima_indices + start_index_window

    #axs[i].vlines(minima_indices, ymin=window[:, i].min(), ymax=window[:, i].max(), color='red', linestyle='dashed', label='accZ Mini

    maxima_indices = detect_local_maxima(window[:, i], max_order)
    #axs[i].vlines(maxima_indices, ymin=window[:, i].min(), ymax=window[:, i].max(), color='blue', linestyle='dashed', label='accZ Max

    windowed_minima_indices_dict[feature] = minima_indices
    windowed_maxima_indices_dict[feature] = maxima_indices

    '''
```

```
    axs[i].set_xlabel('Index')
    axs[i].set_ylabel(f'{feature} Values')
    axs[i].set_title(f'Window {j+1}')
    axs[i].legend()

    '''

# Add a separate subplot for the features in non_windowed_features within the current window


print_statements = [] #re-initialized for every window

non_W_minima_indices_dict = {} #re-initialized for every window
non_W_maxima_indices_dict = {} #re-initialized for every window

for i, (non_windowed_feature, minmax_orders) in enumerate(zip(non_windowed_features, NWF_minmax_orders)):

    index_for_orders = i % len(NWF_minmax_orders)  # Ensure that the index wraps around for the length of NWF_minmax_orders
    globals()[f'{non_windowed_feature}_Minima_order_global'] = minmax_orders[0]
    globals()[f'{non_windowed_feature}_Maxima_order_global'] = minmax_orders[1]


    # Plot the non-windowed features
    feature_data = subject_data[non_windowed_feature].values[start_index_window:start_index_window + window_size]
    filtered_feature_data = butter_lowpass(feature_data, cutoff_freq=25, fs=100, order=2)

    '''
    axs[len(features) + i].plot(range(start_index_window,
                                      start_index_window + len(feature_data)),
                                filtered_feature_data, label=f'{non_windowed_feature} (Filtered)')

    axs[len(features) + i].set_xlabel('Index')
    axs[len(features) + i].set_ylabel(f'{non_windowed_feature} Values')
    axs[len(features) + i].set_title(f'Window {j + 1} - Filtered {non_windowed_feature}')
```

```
axs[len(features) + i].legend()

xticks = np.arange(start_index_window,
                   start_index_window + len(feature_data), 20)  # Adjust the step size as needed
axs[len(features) + i].set_xticks(xticks)
axs[len(features) + i].set_xticklabels(xticks - start_index_window)

#axs[len(features) + i].set_xlim([0, len(feature_data)])

'''




'''

# Check if the function call feeds the right arguments to argrelextrema


print(non_windowed_feature)
print(f'NWF_minmax_orders: {NWF_minmax_orders}')
print(f'minmax_orders: {minmax_orders}')


'''




# Detect local minima and maxima for the current non_windowed_feature
minima_indices = detect_local_minima(filtered_feature_data, minmax_orders[0])
```

```
minima_indices_original = minima_indices + start_index_window  # Use start_index_window

'''
axs[len(features) + i].vlines(minima_indices_original,
                              ymin=filtered_feature_data.min(), ymax=filtered_feature_data.max(),
                              color='red', linestyle='dashed', label=f'{non_windowed_feature} Minima')
'''
maxima_indices = detect_local_maxima(filtered_feature_data, minmax_orders[1])



maxima_indices_original = maxima_indices + start_index_window  # Use start_index_window

'''
axs[len(features) + i].vlines(maxima_indices_original,
                              ymin=filtered_feature_data.min(), ymax=filtered_feature_data.max(),
                              color='blue', linestyle='dashed', label=f'{non_windowed_feature} Maxima')

axs[len(features) + i].legend()
'''
'''
minima_indices_dict[non_windowed_feature] = minima_indices
maxima_indices_dict[non_windowed_feature] = maxima_indices
'''

non_W_minima_indices_dict[non_windowed_feature] = minima_indices
non_W_maxima_indices_dict[non_windowed_feature] = maxima_indices



'''

# append print statements to display where local minimas are (non-windowed feats)
```

```
    txt = f'Window {j + 1} -
    Minima indicies for Filtered {non_windowed_feature}: {minima_indices}'

    print_statements.append(txt)

    txt = f'Window {j + 1} -
    Minima indicies for Filtered {non_windowed_feature} as they appear in original df index: {minima_indices+start_index_subject+start

    print_statements.append(txt)

    # append print statements to display where local maximas are (non-windowed feats)

    txt = f'Window {j + 1} - Maxima indicies for Filtered {non_windowed_feature}: {maxima_indices}'

    print_statements.append(txt)

    txt = f'Window {j + 1} - Maxima indicies for Filtered {non_windowed_feature} as they appear in original df index: {maxima_indices+

    print_statements.append(txt)

    '''



    '''
    Setting threshold for pitch magnitude to rule out inactivity form the labeling process
    '''
if Debug == True:

    print(f'Maxima indices for Pitch window {j+1}:', non_W_maxima_indices_dict['Pitch'])
    print(f'Minima indices for Pitchwindow {j+1}:', non_W_minima_indices_dict['Pitch'])

    Max_pitch_value = subject_data["Pitch"].iloc[start_index_window + non_W_maxima_indices_dict["Pitch"]].max()
    Min_pitch_value = subject_data["Pitch"].iloc[start_index_window + non_W_minima_indices_dict["Pitch"]].min()
```

```
    print(f'Max Pitch: {Max_pitch_value}')
    print(f'Min Pitch: {Min_pitch_value}')

    '''
    print(f'Maxima values for Pitch window {j+1}:', pitch_values_at_maxima)
    print(f'Minima values for Pitch window {j+1}:', pitch_values_at_minima)
    '''

    pitch_magnitude = np.abs(Max_pitch_value - Min_pitch_value)

    print(f'Pitch Mag window {j+1} :',pitch_magnitude)

'''
pitch_magnitude_threshold = 20

if pitch_magnitude < pitch_magnitude_threshold:
    print(f'Skipping window {j+1} due to low pitch magnitude: {pitch_magnitude}')

    Continue_Pitch_TH = True

    break

if Continue_Pitch_TH == True:

    continue

    '''


'''
plt.tight_layout()
plt.show()
'''
```

```
#print('minimas for WIN Pitch:',windowed_minima_indices_dict['Pitch'])
#print('maximas for WIN Pitch:',windowed_maxima_indices_dict['Pitch'])


if Debug == True:


    print('HEEL_STRIKE parameters:')

    print('maximas for Pitch:',non_W_maxima_indices_dict['Pitch'])

    print('maximas for WIN accX:',windowed_maxima_indices_dict['accX'])
    print('maximas for accX:',non_W_maxima_indices_dict['accX'])

    print('minimas for WIN accX:',windowed_minima_indices_dict['accX'])
    print('minimas forr accX:',windowed_minima_indices_dict['accX'])

    print('minimas for WIN accZ:',windowed_minima_indices_dict['accZ'])

    print('maximas for WIN accZ:',windowed_maxima_indices_dict['accZ'])









    '''
    print('minimas for Pitch:',minima_indices_dict['Pitch']+start_index_subject+start_index_window)
    print('maximas for Pitch:',maxima_indices_dict['Pitch']+start_index_subject+start_index_window)
    '''
```

```
    '''
    non_W_minima_indices_dict[non_windowed_feature] = minima_indices
    non_W_maxima_indices_dict[non_windowed_feature] = maxima_indices
    '''

    print('minimas for Pitch:',non_W_minima_indices_dict['Pitch'])

    print('maximas for WIN accZ:',windowed_maxima_indices_dict['accZ'])

    print('maximas for accZ:',non_W_maxima_indices_dict['accZ'])


    print('minimas for accZ:',non_W_minima_indices_dict['accZ'])



    print(f'Window {j+1} - Start Index subject: {start_index_subject}')
    print(f'Window {j+1} - Start Index window: {start_index_window}')


W_zoom_lower = 10
W_zoom_upper = 70

TO_lower = 5
TO_upper = 20

HS_accX_Max = 30
HS_accX_Min = 20

HS_lower = 4
HS_upper_Max = HS_accX_Max
HS_upper_Min = HS_accX_Min
```

```
HS_Impact_accZ_Min = 20



HS_Impact_Search_Lower = 0
HS_Impact_Search_Upper = HS_Impact_accZ_Min

Index_Loc_Max_Pitch_Bool = False

Index_Loc_Min_Pitch_Bool = False


 # HEEL STRIKE incidents
Pitch_Maxs = [index for index in non_W_maxima_indices_dict['Pitch'] if W_zoom_lower < index < W_zoom_upper]

if any(Pitch_Maxs):

    if Debug == True:
        print(f'Condition met: There are indices greater than {W_zoom_lower} and less than {W_zoom_upper} for loc MAX pitch. Possible

    Index_Loc_Max_Pitch = Pitch_Maxs[0]  # if multiple instances, handle the first one in the window first
    Index_Loc_Max_Pitch_Bool = True

    if Debug == True:

        print('index loc max pitch:', Index_Loc_Max_Pitch)

else:

    if Debug == True:

        print(f'No indices found for loc MAX pitch in the specified range {W_zoom_lower}-{W_zoom_upper}.')
```

```
 # Toe Off incidents
Pitch_Mins = [index for index in non_W_minima_indices_dict['Pitch'] if W_zoom_lower < index < W_zoom_upper]

if any(Pitch_Mins):
    if Debug == True:

        print(f'Condition met: There are indices greater than {W_zoom_lower} and less than {W_zoom_upper} for loc MIN pitch. Possible

    Index_Loc_Min_Pitch = Pitch_Mins[0]
    Index_Loc_Min_Pitch_Bool = True

    if Debug == True:
        print('index loc min pitch:', Index_Loc_Min_Pitch)

else:
    if Debug == True:
        print(f'No indices found for loc MIN pitch in the specified range {W_zoom_lower}-{W_zoom_upper}.')



if Index_Loc_Min_Pitch_Bool == True or Index_Loc_Max_Pitch_Bool == True:


    if Index_Loc_Min_Pitch_Bool == False and Index_Loc_Max_Pitch_Bool == True:


        HS_Detected = detect_heel_strike(Debug, Index_Loc_Max_Pitch, windowed_maxima_indices_dict, non_W_maxima_indices_dict,
               windowed_minima_indices_dict, non_W_minima_indices_dict,
               start_index_subject, start_index_window,
               HS_lower, HS_upper_Max, HS_upper_Min, HS_Impact_Search_Lower, HS_Impact_Search_Upper,
               W_zoom_lower, W_zoom_upper,
               df_Labeling_Copy, df)
```

```
if Index_Loc_Min_Pitch_Bool == True and Index_Loc_Max_Pitch_Bool == False:


    TO_Detected = detect_toe_off(Debug, Index_Loc_Min_Pitch, W_zoom_lower, W_zoom_upper, TO_lower, TO_upper, start_index_subject,
     windowed_maxima_indices_dict, non_W_minima_indices_dict, df_Labeling_Copy, Event_Tagged, df)



if (Index_Loc_Min_Pitch_Bool == True and Index_Loc_Max_Pitch_Bool == True) and Index_Loc_Min_Pitch < Index_Loc_Max_Pitch:

    if Debug == True:

        print ('Possibility of both TO and HS, in that order. Running TO first:')


    TO_Detected = detect_toe_off(Debug, Index_Loc_Min_Pitch, W_zoom_lower, W_zoom_upper, TO_lower, TO_upper, start_index_subject,
         windowed_maxima_indices_dict, non_W_minima_indices_dict, df_Labeling_Copy, Event_Tagged, df)

    if Debug == True:
        print ('TO finished, now we want HS to run: ')

    HS_Detected = detect_heel_strike(Debug, Index_Loc_Max_Pitch, windowed_maxima_indices_dict, non_W_maxima_indices_dict,
           windowed_minima_indices_dict, non_W_minima_indices_dict,
           start_index_subject, start_index_window,
           HS_lower, HS_upper_Max, HS_upper_Min, HS_Impact_Search_Lower, HS_Impact_Search_Upper,
           W_zoom_lower, W_zoom_upper,
           df_Labeling_Copy, df)


if (Index_Loc_Min_Pitch_Bool == True and Index_Loc_Max_Pitch_Bool == True) and Index_Loc_Min_Pitch > Index_Loc_Max_Pitch:
```

```
            HS_Detected = detect_heel_strike(Debug, Index_Loc_Max_Pitch, windowed_maxima_indices_dict, non_W_maxima_indices_dict,
                   windowed_minima_indices_dict, non_W_minima_indices_dict,
                   start_index_subject, start_index_window,
                   HS_lower, HS_upper_Max, HS_upper_Min, HS_Impact_Search_Lower, HS_Impact_Search_Upper,
                   W_zoom_lower, W_zoom_upper,
                   df_Labeling_Copy, df)

            TO_Detected = detect_toe_off(Debug, Index_Loc_Min_Pitch, W_zoom_lower, W_zoom_upper, TO_lower, TO_upper, start_index_subject,
                 windowed_maxima_indices_dict, non_W_minima_indices_dict, df_Labeling_Copy, Event_Tagged, df)

return df_Labeling_Copy



    #for subject in range (0 , Subjects) :

    # Go through subjects [0,29], adjust parameters to detect as many TO and HS as possible.
```

W_zoom_lower = 10 W_zoom_upper = 70

TO_lower = 5 TO_upper = 10

HS_accX_Max = 30 HS_accX_Min = 20

HS_lower = 4 HS_upper_Max = HS_accX_Max HS_upper_Min = HS_accX_Min

HS_Impact_accZ_Min = 20

HS_Impact_Search_Lower = 0 HS_Impact_Search_Upper = HS_Impact_accZ_Min

Index_Loc_Max_Pitch_Bool = False

Index_Loc_Min_Pitch_Bool = False

Subjects = 30

for subject in range (0 , Subjects) :

```
print(f'Subject: {subject}')
```

```
windowed_subject_data_plotter(df, df_Labeling_Copy, 'Gait_Identification', subject, Debug=False, features=['accZ', 'accX'],
                              minmax_windowed_features=[[12,15],[5,5]],
                              cutoff_freq=15, fs=100, order=3,
                              non_windowed_features=['Pitch', 'accZ', 'accX'],
                              NWF_minmax_orders=[[35, 30], [int(HS_Impact_accZ_Min*0.8), 8], [9, 10]])
```

print('Saving df')

df_GE_Tags = df_Labeling_Copy.copy() df_GE_Tags.to_csv('df_GE_Tags_13_12')

```
# df_GE_Tags.loc[600:700]

# df_GE_Tags.dtypes

#df_GE_Tags.to_parquet('df_GE_Tags_13_12.parquet')

#df_GE_Tags
'''

Upgrade HS 13_12


'''
```



```
import matplotlib.pyplot as plt
import numpy as np
from scipy.signal import butter, filtfilt, argrelextrema

def butter_lowpass(data, cutoff_freq, fs, order=4):
    nyquist = 0.5 * fs
    normal_cutoff = cutoff_freq / nyquist
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    y = filtfilt(b, a, data, axis=0)
```

```
    return y

def normalize_window(window):
    # Normalize each column (feature) independently
    normalized_window = (window - window.mean(axis=0)) / window.std(axis=0)
    return normalized_window

def detect_local_minima(data, order):
    minima_indices = argrelextrema(data, np.less_equal, order=order)[0]
    return minima_indices

def detect_local_maxima(data, order):
    maxima_indices = argrelextrema(data, np.greater_equal, order=order)[0]
    return maxima_indices

def plot_around_event(event_type, event_pos_in_df, zoom, features, df=df):
    # Determine start and end indices based on the zoom and event position
    start = int(event_pos_in_df - zoom)
    end = int(event_pos_in_df + zoom)

    # Extract the relevant portion of the DataFrame
    df_plot = df.loc[start:end, :]

    # Extract the subject, for use in plot name
    # column name is in 'Gait_Identification'

    name = df.loc[start, 'Gait_Identification']


    # Create a figure and primary axis
    fig, ax1 = plt.subplots(figsize=(10, 6))

    # Plot each feature on the primary axis with twinx for secondary and tertiary axes
    for i, feature in enumerate(features):
```

```
        ax = ax1.twinx() if i > 0 else ax1
        color = ['blue', 'orange', 'green'][i]
        ax.plot(df_plot[feature], label=feature, color=color)
        ax.set_ylabel(feature, color=color)
        ax.tick_params(axis='y', labelcolor=color)

        if i == 2:  # Make some space for y_ax for the third feature (accX)
            ax.spines['right'].set_position(('outward', 60))

    # Add a vertical line at the event position
    ax1.axvline(x=event_pos_in_df, color='red', linestyle='--', label=f'{event_type}')

    # Add a title and legend based on the event type
    if event_type == 'TO':
        plt.title(f'Original DataFrame around Toe Off at {start}:{end} in df, subject {name}')
    elif event_type == 'HS':
        plt.title(f'Original DataFrame around Heel-strike at {start}:{end} in df, subject {name}')

    fig.legend(loc='upper right', bbox_to_anchor=(0.85, 0.85))

    plt.savefig(f'{event_type}_pos_{start}_{end}_subject_{name}_in_dataframe', format='eps') #f strings are nice


    plt.show()



def detect_heel_strike(Debug, Index_Loc_Max_Pitch, windowed_maxima_indices_dict, non_W_maxima_indices_dict,
                       windowed_minima_indices_dict, non_W_minima_indices_dict,
                       start_index_subject, start_index_window,
                       HS_lower, HS_upper_Max, HS_upper_Min, HS_Impact_Search_Lower, HS_Impact_Search_Upper,
                       W_zoom_lower, W_zoom_upper,
                       df_Labeling_Copy, df):
    if Debug == True:
```

```
    print('Running HS detection')

Event_Detected = False



Index_Loc_Min_accX = [index for index in windowed_minima_indices_dict['accX'] if
                      Index_Loc_Max_Pitch - HS_lower <= index <= Index_Loc_Max_Pitch + HS_upper_Min]


if not any(Index_Loc_Min_accX):
    # If no candidates in windowed_maxima_indices_dict['accX'], check non_W_maxima_indices_dict['accX']
    Index_Loc_Min_accX_nonW = [index for index in non_W_minima_indices_dict['accX'] if
                               Index_Loc_Max_Pitch - HS_lower <= index <= Index_Loc_Max_Pitch + HS_upper_Min]

    if any(Index_Loc_Min_accX_nonW):
        Window_Index_Loc_Min_accX = Index_Loc_Min_accX_nonW[0]

        if Debug == True:

            print('accX MIN component of HS found inside nonW:', Window_Index_Loc_Min_accX)
    else:
        if Debug == True:
            print('Not able to find local min accX near Pitch Max')
        return None

else:
    Window_Index_Loc_Min_accX = Index_Loc_Min_accX[0]
    if Debug == True:
        print('accX MIN component of HS found inside WIN:', Window_Index_Loc_Min_accX)


if Window_Index_Loc_Min_accX is not None:
```

```
    # Check for local maxima of accX within a range around Pitch maxima
    Index_Loc_Max_accX = [index for index in windowed_maxima_indices_dict['accX'] if
                      Window_Index_Loc_Min_accX <= index <= Index_Loc_Max_Pitch + HS_upper_Max]

    if not any(Index_Loc_Max_accX):
        # If no candidates in windowed_maxima_indices_dict['accX'], check non_W_maxima_indices_dict['accX']
        Index_Loc_Max_accX_nonW = [index for index in non_W_maxima_indices_dict['accX'] if
                                   Index_Loc_Max_Pitch <= index <= Index_Loc_Max_Pitch + HS_upper_Max]

        if any(Index_Loc_Max_accX_nonW):
            Window_Index_Loc_Max_accX = Index_Loc_Max_accX_nonW[0]

            if Debug == True:

                print('accX MAX component of HS found inside nonW:', Window_Index_Loc_Max_accX)
        else:
            if Debug == True:
                print('Not able to find local max accX near accX min')
            return None

    else:
        Window_Index_Loc_Max_accX = Index_Loc_Max_accX[0]

        if Debug == True:
            print('accX MAX component of HS found inside WIN:', Window_Index_Loc_Max_accX)


if Window_Index_Loc_Max_accX is not None and Window_Index_Loc_Min_accX is not None:


    if Debug == True:
        print('Found both min and max for accX component near max for pitch. Probable HS. Search for accZ min.')
```

```
Cand_Loc_Min_accZ = [index for index in windowed_minima_indices_dict['accZ'] if
                    Window_Index_Loc_Max_accX - HS_Impact_Search_Lower <= index <= Window_Index_Loc_Max_accX + HS_Impact_Search_Upper

if Debug == True:

    print('these are accZ within range of accX max:', Cand_Loc_Min_accZ)

if any(Cand_Loc_Min_accZ):

    if Debug == True:

        print('Found candidates for accZ: ', Cand_Loc_Min_accZ)

    Cand_Loc_Min_accZ = Cand_Loc_Min_accZ[0]

    if Debug == True:

        print('Heel Strike at w index: ', Cand_Loc_Min_accZ)
        print('Heel Strike at Index: ', Cand_Loc_Min_accZ + start_index_subject + start_index_window)

    Heel_strike = Cand_Loc_Min_accZ + start_index_subject + start_index_window

    if not any(df_Labeling_Copy.loc[max(Heel_strike - 20, 0):min(Heel_strike + 20, len(df_Labeling_Copy) - 1), 'Gait_Events'] == '

        #Condition stops search from going out of bounds near start/end of labeling_df


            Zoom = 50

            plot_around_event('HS', Heel_strike, zoom=Zoom, features=['Pitch', 'accZ', 'accX'], df=df)
```

```
                    df_Labeling_Copy.loc[Heel_strike, 'Gait_Events'] = 'HS'

                    Event_Detected = True

            else:
                if Debug == True:

                    print("Heel Strike already exists within ±20 positions; skip tag.")

        else:
            if Debug == True:

                print('Not able to find local min accZ near Pitch Max')


        if Event_Detected:

            return True

    return False


def detect_toe_off(Debug, Index_Loc_Min_Pitch, W_zoom_lower, W_zoom_upper, TO_lower, TO_upper, start_index_subject, start_index_window,
                   windowed_maxima_indices_dict, non_W_minima_indices_dict, df_Labeling_Copy, Event_Tagged,
                   df, Zoom=50):

    if Debug == True:
        print('Running Toe_Off detection')

    Event_Tagged = False

    if any(Index_Loc_Min_Pitch - TO_lower < index < Index_Loc_Min_Pitch + TO_upper for index in
           windowed_maxima_indices_dict['accZ']):
```

```
matching_indices = np.where(
    (Index_Loc_Min_Pitch - TO_lower < windowed_maxima_indices_dict['accZ']) &
    (windowed_maxima_indices_dict['accZ'] < Index_Loc_Min_Pitch + TO_upper))[0]

Index_Loc_Max_W_accZ = windowed_maxima_indices_dict['accZ'][matching_indices[0]]

if Debug == True:

    print('all instances', windowed_maxima_indices_dict['accZ'])
    print('index in list that meets criteria: ', matching_indices)
    print('Toe off at: ',
          Index_Loc_Max_W_accZ,
          'convert to global coordinates:',
          Index_Loc_Max_W_accZ + start_index_subject + start_index_window)

Toe_Off = Index_Loc_Max_W_accZ + start_index_subject + start_index_window

Event_Tagged = True

if Debug == True:

    print('toe off at: ', Toe_Off)

if not any(df_Labeling_Copy.loc[max(Toe_Off - 20, 0):min(Toe_Off + 20, len(df_Labeling_Copy) - 1), 'Gait_Events'] == 'TO'):
    #Condition stops search from going out of bounds near start/end of labeling_df

    df_Labeling_Copy.loc[Toe_Off, 'Gait_Events'] = 'TO'

    plot_around_event('TO', Toe_Off, zoom=Zoom, features=['Pitch', 'accZ', 'accX'], df=df)

else:
    if Debug == True:

        print("Toe_off already exists within ±20 positions; skip tag.")
```

```
    else:
        if Debug == True:

            print(f'Toe_Off_Detection: Could not find accZ MAX in range [ {Index_Loc_Min_Pitch - TO_lower} , {Index_Loc_Min_Pitch + TO_upp

    if Event_Tagged:

        return True

    return False


def windowed_subject_data_plotter(df, df_Labeling_Copy, subject_column, subject_number, Debug=True, features=None, window_size=100,
                                  overlap=50, cutoff_freq=None, fs=None, order=None,
                                  non_windowed_features=None, NWF_minmax_orders=None,
                                  minmax_windowed_features=None):

    # Filter the DataFrame based on the specified subject and subject column
    subject_data = df[df[subject_column] == subject_number]

    start_index_subject = subject_data.index[0]

    # Extract the selected features
    selected_data = subject_data[features]

    # Apply Hamming windowing and Butterworth lowpass filtering to the data
    hamming_window = np.hamming(window_size)[:, None]  # Reshape to column vector
    windows = [
        butter_lowpass(selected_data[i:i+window_size].values * hamming_window, cutoff_freq, fs=fs, order=order)
        for i in range(0, len(selected_data)-window_size+1, overlap)
    ]

    # Normalize each window
```

```
normalized_windows = [normalize_window(window) for window in windows]

windowed_minima_indices_dict = {} #re-initialized for every window
windowed_maxima_indices_dict = {} #re-initialized for every window

Continue_Pitch_TH = False

# Loop over non_windowed_features and NWF_minmax_orders simultaneously
for j, (window, unfiltered_window) in enumerate(zip(normalized_windows, windows)):



    '''
    Event_Tagged is used to skip to next window as soon as either HS or TO is tagged
    This mediates the problem of processing events to near the edge of the window,
    where the characteristics of the measurands are dirstorted.

    '''

    Event_Tagged = False




    start_index_window = j * (window_size - overlap)


    fig, axs = plt.subplots(len(features) + len(non_windowed_features), 1, figsize=(10, 2*(len(features) + len(non_windowed_features))



    # Plot the normalized and filtered windowed data in separate plots for each feature and window
    for i, feature in enumerate(features):
        axs[i].plot(range(len(window)), window[:, i], label=f'{feature} (Filtered and Normalized)')
```

```
        min_order, max_order = minmax_windowed_features[i] # fetch orders from function call list minmax_windowed_features=[[8,8],[35,

        minima_indices = detect_local_minima(window[:, i], min_order)
        minima_indices_original = minima_indices + start_index_window

        axs[i].vlines(minima_indices, ymin=window[:, i].min(), ymax=window[:, i].max(), color='red', linestyle='dashed', label='accZ M

        maxima_indices = detect_local_maxima(window[:, i], max_order)
        axs[i].vlines(maxima_indices, ymin=window[:, i].min(), ymax=window[:, i].max(), color='blue', linestyle='dashed', label='accZ

        windowed_minima_indices_dict[feature] = minima_indices
        windowed_maxima_indices_dict[feature] = maxima_indices

        axs[i].set_xlabel('Index')
        axs[i].set_ylabel(f'{feature} Values')
        axs[i].set_title(f'Window {j+1}')
        axs[i].legend()

    # Add a separate subplot for the features in non_windowed_features within the current window


    print_statements = [] #re-initialized for every window

    non_W_minima_indices_dict = {} #re-initialized for every window
    non_W_maxima_indices_dict = {} #re-initialized for every window

    for i, (non_windowed_feature, minmax_orders) in enumerate(zip(non_windowed_features, NWF_minmax_orders)):

        index_for_orders = i % len(NWF_minmax_orders)  # Ensure that the index wraps around for the length of NWF_minmax_orders
        globals()[f'{non_windowed_feature}_Minima_order_global'] = minmax_orders[0]
        globals()[f'{non_windowed_feature}_Maxima_order_global'] = minmax_orders[1]
```

```
# Plot the non-windowed features
feature_data = subject_data[non_windowed_feature].values[start_index_window:start_index_window + window_size]
filtered_feature_data = butter_lowpass(feature_data, cutoff_freq=25, fs=100, order=2)
axs[len(features) + i].plot(range(start_index_window,
                                  start_index_window + len(feature_data)),
                            filtered_feature_data, label=f'{non_windowed_feature} (Filtered)')

axs[len(features) + i].set_xlabel('Index')
axs[len(features) + i].set_ylabel(f'{non_windowed_feature} Values')
axs[len(features) + i].set_title(f'Window {j + 1} - Filtered {non_windowed_feature}')
axs[len(features) + i].legend()
xticks = np.arange(start_index_window,
                   start_index_window + len(feature_data), 20)  # Adjust the step size as needed
axs[len(features) + i].set_xticks(xticks)
axs[len(features) + i].set_xticklabels(xticks - start_index_window)

#axs[len(features) + i].set_xlim([0, len(feature_data)])



'''

# Check if the function call feeds the right arguments to argrelextrema


print(non_windowed_feature)
print(f'NWF_minmax_orders: {NWF_minmax_orders}')
print(f'minmax_orders: {minmax_orders}')


'''
```

```
# Detect local minima and maxima for the current non_windowed_feature
minima_indices = detect_local_minima(filtered_feature_data, minmax_orders[0])



minima_indices_original = minima_indices + start_index_window  # Use start_index_window
axs[len(features) + i].vlines(minima_indices_original,
                              ymin=filtered_feature_data.min(), ymax=filtered_feature_data.max(),
                              color='red', linestyle='dashed', label=f'{non_windowed_feature} Minima')

maxima_indices = detect_local_maxima(filtered_feature_data, minmax_orders[1])



maxima_indices_original = maxima_indices + start_index_window  # Use start_index_window
axs[len(features) + i].vlines(maxima_indices_original,
                              ymin=filtered_feature_data.min(), ymax=filtered_feature_data.max(),
                              color='blue', linestyle='dashed', label=f'{non_windowed_feature} Maxima')

axs[len(features) + i].legend()

'''
minima_indices_dict[non_windowed_feature] = minima_indices
maxima_indices_dict[non_windowed_feature] = maxima_indices
'''

non_W_minima_indices_dict[non_windowed_feature] = minima_indices
non_W_maxima_indices_dict[non_windowed_feature] = maxima_indices



'''
```

```
    # append print statements to display where local minimas are (non-windowed feats)

    txt = f'Window {j + 1} -
    Minima indicies for Filtered {non_windowed_feature}: {minima_indices}'

    print_statements.append(txt)

    txt = f'Window {j + 1} -
    Minima indicies for Filtered {non_windowed_feature} as they appear in original df index: {minima_indices+start_index_subject+s

    print_statements.append(txt)

    # append print statements to display where local maximas are (non-windowed feats)

    txt = f'Window {j + 1} - Maxima indicies for Filtered {non_windowed_feature}: {maxima_indices}'

    print_statements.append(txt)

    txt = f'Window {j + 1} - Maxima indicies for Filtered {non_windowed_feature} as they appear in original df index: {maxima_indi

    print_statements.append(txt)

    '''



    '''
    Setting threshold for pitch magnitude to rule out inactivity form the labeling process
    '''


print(f'Maxima indices for Pitch window {j+1}:', non_W_maxima_indices_dict['Pitch'])
print(f'Minima indices for Pitchwindow {j+1}:', non_W_minima_indices_dict['Pitch'])
```

```
Max_pitch_value = subject_data["Pitch"].iloc[start_index_window + non_W_maxima_indices_dict["Pitch"]].max()
Min_pitch_value = subject_data["Pitch"].iloc[start_index_window + non_W_minima_indices_dict["Pitch"]].min()

print(f'Max Pitch: {Max_pitch_value}')
print(f'Min Pitch: {Min_pitch_value}')

'''
print(f'Maxima values for Pitch window {j+1}:', pitch_values_at_maxima)
print(f'Minima values for Pitch window {j+1}:', pitch_values_at_minima)
'''

pitch_magnitude = np.abs(Max_pitch_value - Min_pitch_value)

print(f'Pitch Mag window {j+1} :',pitch_magnitude)

'''
pitch_magnitude_threshold = 20

if pitch_magnitude < pitch_magnitude_threshold:
    print(f'Skipping window {j+1} due to low pitch magnitude: {pitch_magnitude}')

    Continue_Pitch_TH = True

    break

if Continue_Pitch_TH == True:

    continue

    '''
```

```
plt.tight_layout()
plt.show()



#print('minimas for WIN Pitch:',windowed_minima_indices_dict['Pitch'])
#print('maximas for WIN Pitch:',windowed_maxima_indices_dict['Pitch'])


if Debug == True:


    print('HEEL_STRIKE parameters:')

    print('maximas for Pitch:',non_W_maxima_indices_dict['Pitch'])

    print('maximas for WIN accX:',windowed_maxima_indices_dict['accX'])
    print('maximas for accX:',non_W_maxima_indices_dict['accX'])

    print('minimas for WIN accX:',windowed_minima_indices_dict['accX'])
    print('minimas forr accX:',windowed_minima_indices_dict['accX'])

    print('minimas for WIN accZ:',windowed_minima_indices_dict['accZ'])

    print('maximas for WIN accZ:',windowed_maxima_indices_dict['accZ'])









    '''
```

```
        print('minimas for Pitch:',minima_indices_dict['Pitch']+start_index_subject+start_index_window)
        print('maximas for Pitch:',maxima_indices_dict['Pitch']+start_index_subject+start_index_window)
        '''

        '''
        non_W_minima_indices_dict[non_windowed_feature] = minima_indices
        non_W_maxima_indices_dict[non_windowed_feature] = maxima_indices
        '''

        print('minimas for Pitch:',non_W_minima_indices_dict['Pitch'])

        print('maximas for WIN accZ:',windowed_maxima_indices_dict['accZ'])

        print('maximas for accZ:',non_W_maxima_indices_dict['accZ'])


        print('minimas for accZ:',non_W_minima_indices_dict['accZ'])



        print(f'Window {j+1} - Start Index subject: {start_index_subject}')
        print(f'Window {j+1} - Start Index window: {start_index_window}')


W_zoom_lower = 10
W_zoom_upper = 70

TO_lower = 5
TO_upper = 20

HS_accX_Max = 30
HS_accX_Min = 20
```

```
HS_lower = 4
HS_upper_Max = HS_accX_Max
HS_upper_Min = HS_accX_Min

HS_Impact_accZ_Min = 20



HS_Impact_Search_Lower = 0
HS_Impact_Search_Upper = HS_Impact_accZ_Min

Index_Loc_Max_Pitch_Bool = False

Index_Loc_Min_Pitch_Bool = False


 # HEEL STRIKE incidents
Pitch_Maxs = [index for index in non_W_maxima_indices_dict['Pitch'] if W_zoom_lower < index < W_zoom_upper]

if any(Pitch_Maxs):

    if Debug == True:
        print(f'Condition met: There are indices greater than {W_zoom_lower} and less than {W_zoom_upper} for loc MAX pitch. Possi

    Index_Loc_Max_Pitch = Pitch_Maxs[0]  # if multiple instances, handle the first one in the window first
    Index_Loc_Max_Pitch_Bool = True

    if Debug == True:

        print('index loc max pitch:', Index_Loc_Max_Pitch)

else:

    if Debug == True:
```

```
        print(f'No indices found for loc MAX pitch in the specified range {W_zoom_lower}-{W_zoom_upper}.')

 # Toe Off incidents
Pitch_Mins = [index for index in non_W_minima_indices_dict['Pitch'] if W_zoom_lower < index < W_zoom_upper]

if any(Pitch_Mins):
    if Debug == True:

        print(f'Condition met: There are indices greater than {W_zoom_lower} and less than {W_zoom_upper} for loc MIN pitch. Possi

    Index_Loc_Min_Pitch = Pitch_Mins[0]
    Index_Loc_Min_Pitch_Bool = True

    if Debug == True:
        print('index loc min pitch:', Index_Loc_Min_Pitch)

else:
    if Debug == True:
        print(f'No indices found for loc MIN pitch in the specified range {W_zoom_lower}-{W_zoom_upper}.')



if Index_Loc_Min_Pitch_Bool == True or Index_Loc_Max_Pitch_Bool == True:


    if Index_Loc_Min_Pitch_Bool == False and Index_Loc_Max_Pitch_Bool == True:


        HS_Detected = detect_heel_strike(Debug, Index_Loc_Max_Pitch, windowed_maxima_indices_dict, non_W_maxima_indices_dict,
               windowed_minima_indices_dict, non_W_minima_indices_dict,
               start_index_subject, start_index_window,
               HS_lower, HS_upper_Max, HS_upper_Min, HS_Impact_Search_Lower, HS_Impact_Search_Upper,
               W_zoom_lower, W_zoom_upper,
```

```
            df_Labeling_Copy, df)



if Index_Loc_Min_Pitch_Bool == True and Index_Loc_Max_Pitch_Bool == False:


    TO_Detected = detect_toe_off(Debug, Index_Loc_Min_Pitch, W_zoom_lower, W_zoom_upper, TO_lower, TO_upper, start_index_subje
     windowed_maxima_indices_dict, non_W_minima_indices_dict, df_Labeling_Copy, Event_Tagged, df)



if (Index_Loc_Min_Pitch_Bool == True and Index_Loc_Max_Pitch_Bool == True) and Index_Loc_Min_Pitch < Index_Loc_Max_Pitch:

    if Debug == True:

        print ('Possibility of both TO and HS, in that order. Running TO first:')


    TO_Detected = detect_toe_off(Debug, Index_Loc_Min_Pitch, W_zoom_lower, W_zoom_upper, TO_lower, TO_upper, start_index_subje
         windowed_maxima_indices_dict, non_W_minima_indices_dict, df_Labeling_Copy, Event_Tagged, df)

    if Debug == True:
        print ('TO finished, now we want HS to run: ')

    HS_Detected = detect_heel_strike(Debug, Index_Loc_Max_Pitch, windowed_maxima_indices_dict, non_W_maxima_indices_dict,
           windowed_minima_indices_dict, non_W_minima_indices_dict,
           start_index_subject, start_index_window,
           HS_lower, HS_upper_Max, HS_upper_Min, HS_Impact_Search_Lower, HS_Impact_Search_Upper,
           W_zoom_lower, W_zoom_upper,
           df_Labeling_Copy, df)
```

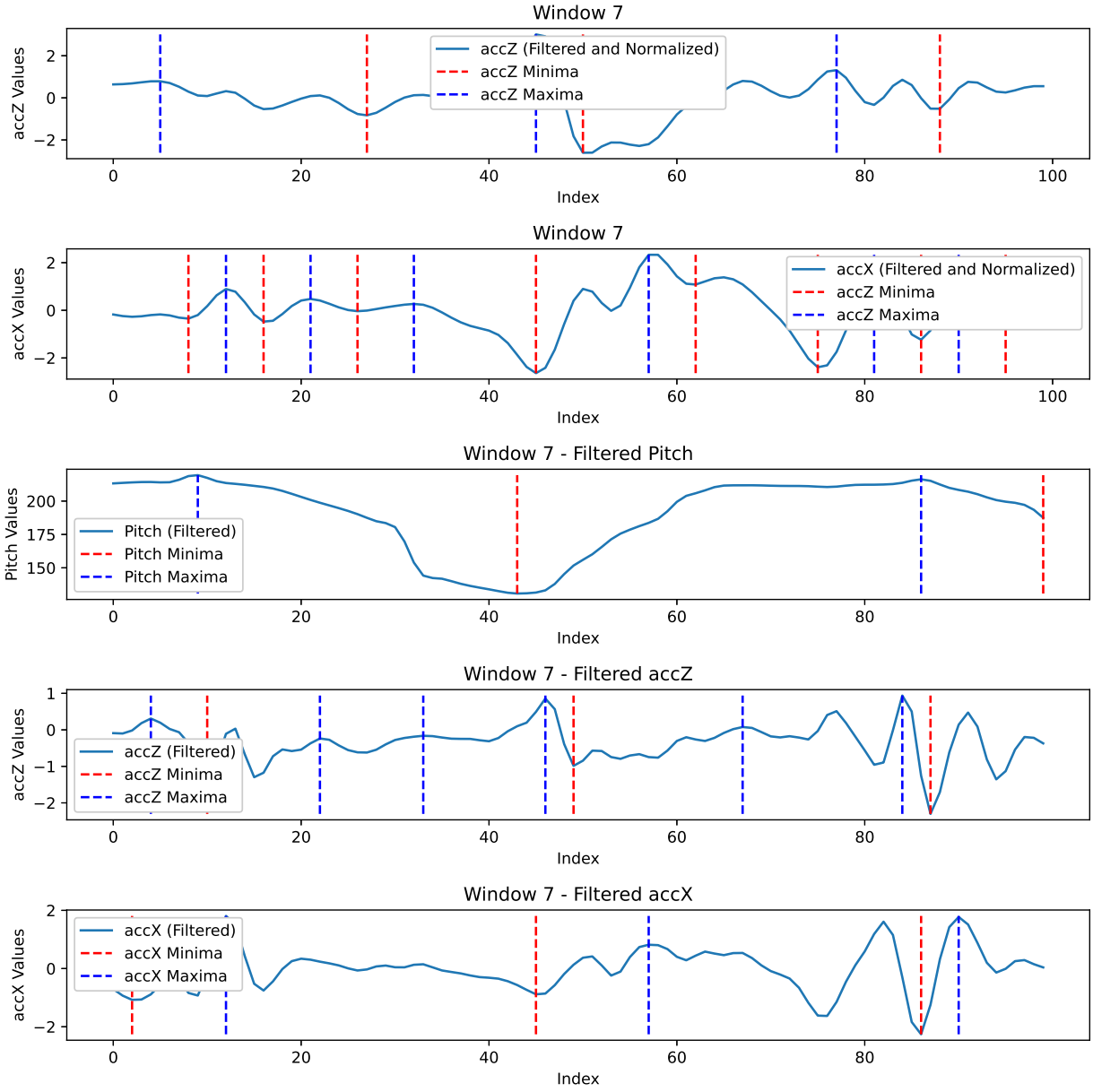
```
            if (Index_Loc_Min_Pitch_Bool == True and Index_Loc_Max_Pitch_Bool == True) and Index_Loc_Min_Pitch > Index_Loc_Max_Pitch:

                HS_Detected = detect_heel_strike(Debug, Index_Loc_Max_Pitch, windowed_maxima_indices_dict, non_W_maxima_indices_dict,
                       windowed_minima_indices_dict, non_W_minima_indices_dict,
                       start_index_subject, start_index_window,
                       HS_lower, HS_upper_Max, HS_upper_Min, HS_Impact_Search_Lower, HS_Impact_Search_Upper,
                       W_zoom_lower, W_zoom_upper,
                       df_Labeling_Copy, df)

                TO_Detected = detect_toe_off(Debug, Index_Loc_Min_Pitch, W_zoom_lower, W_zoom_upper, TO_lower, TO_upper, start_index_subje
                     windowed_maxima_indices_dict, non_W_minima_indices_dict, df_Labeling_Copy, Event_Tagged, df)

    return df_Labeling_Copy


        #for subject in range (0 , Subjects) :

        # Go through subjects [0,29], adjust parameters to detect as many TO and HS as possible.


W_zoom_lower = 10
W_zoom_upper = 70

TO_lower = 5
TO_upper = 10

HS_accX_Max = 30
HS_accX_Min = 20

HS_lower = 4
HS_upper_Max = HS_accX_Max
```

```
HS_upper_Min = HS_accX_Min

HS_Impact_accZ_Min = 20



HS_Impact_Search_Lower = 0
HS_Impact_Search_Upper = HS_Impact_accZ_Min

Index_Loc_Max_Pitch_Bool = False

Index_Loc_Min_Pitch_Bool = False

'''
Subjects = 30

for subject in range (0 , Subjects) :

    print(f'Subject: {subject}')
'''

windowed_subject_data_plotter(df, df_Labeling_Copy, 'Gait_Identification', 1, Debug=True, features=['accZ', 'accX'],
                          minmax_windowed_features=[[12,15],[5,5]],
                          cutoff_freq=15, fs=100, order=3,
                          non_windowed_features=['Pitch', 'accZ', 'accX'],
                          NWF_minmax_orders=[[35, 30], [int(HS_Impact_accZ_Min*0.8), 8], [9, 10]])
```

Important notice: The output has been greatly reduced to better display how the labeling algorithm works.

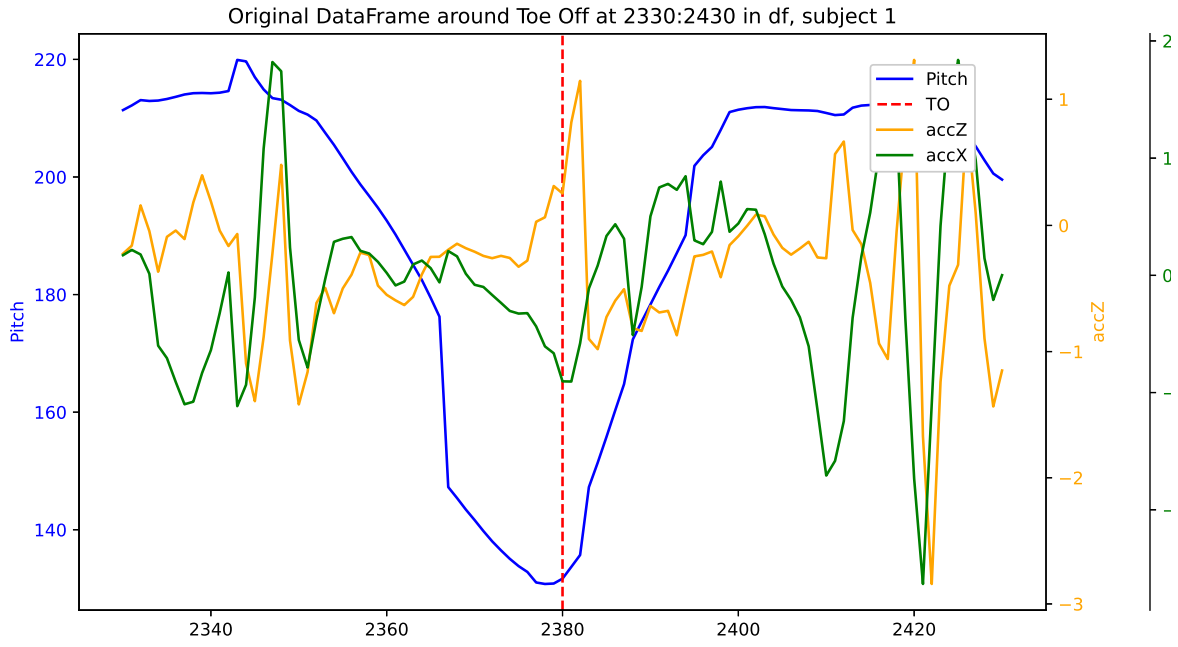
The events are tagged in the df, in preparation for labeling gait progression later.

```
Maxima indices for Pitch window 7: [ 9 86]
Minima indices for Pitchwindow 7: [43 99]
Max Pitch: 219.66281127929688
Min Pitch: 130.77691650390625
Pitch Mag window 7 : 88.885895
```


Window 7
accZ Values
accZ (Filtered and Normalized)
accZ Minima
accZ Maxima
Index
Window 7
accX Values
accX (Filtered and Normalized)
accZ Minima
accZ Maxima
Index
Window 7 - Filtered Pitch
Pitch Values
Pitch (Filtered)
Pitch Minima
Pitch Maxima
Index
Window 7 - Filtered accZ
accZ Values
accZ (Filtered)
accZ Minima
accZ Maxima
Index
Window 7 - Filtered accX
accX Values
accX (Filtered)
accX Minima
accX Maxima
Index

figure

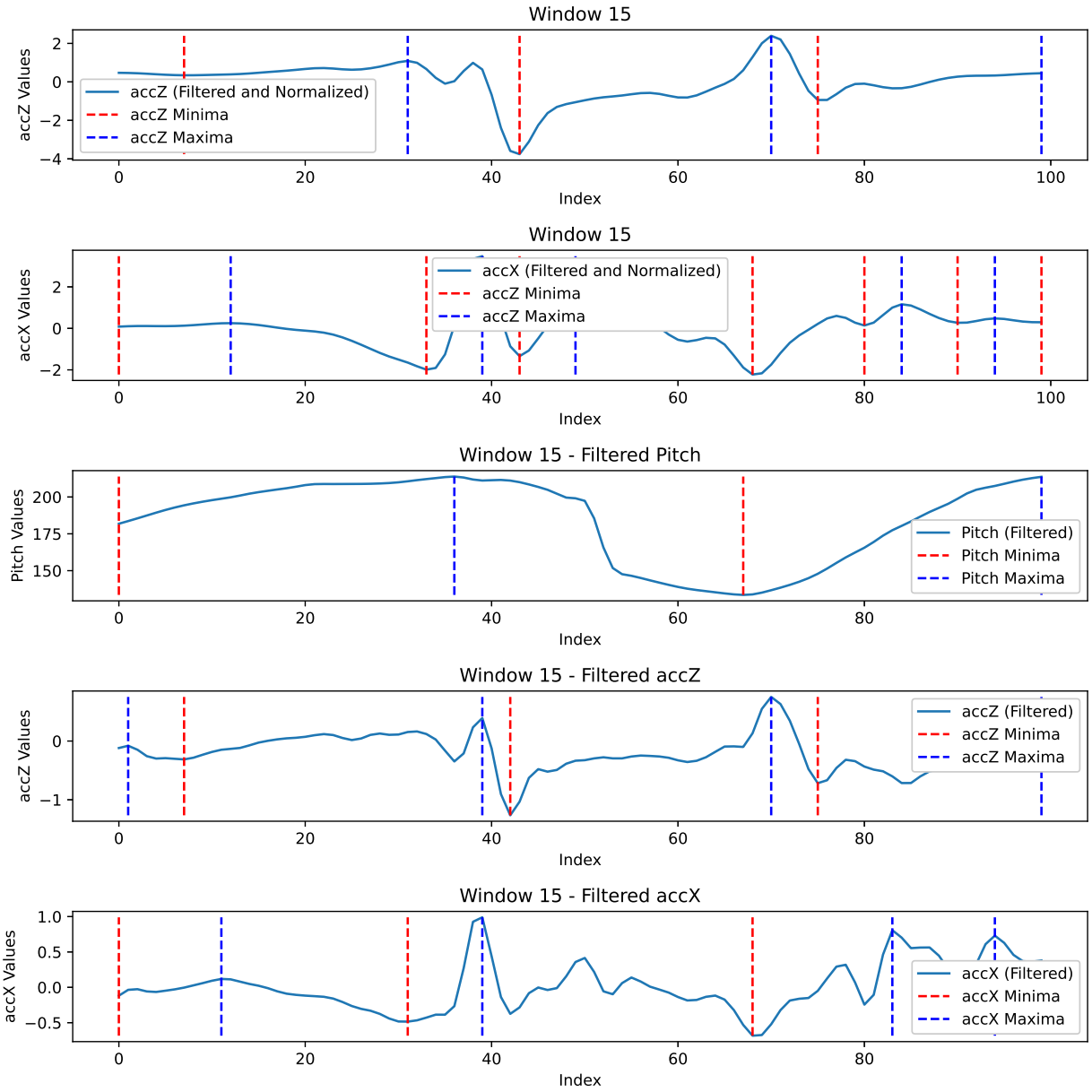
```
HEEL_STRIKE parameters:
maximas for Pitch: [ 9 86]
maximas for WIN accX: [12 21 32 57 81 90]
maximas for accX: [12 57 90]
minimas for WIN accX: [ 8 16 26 45 62 75 86 95]
minimas forr accX: [ 8 16 26 45 62 75 86 95]
minimas for WIN accZ: [27 50 88]
maximas for WIN accZ: [ 5 45 77]
minimas for Pitch: [43 99]
maximas for WIN accZ: [ 5 45 77]
maximas for accZ: [ 4 22 33 46 67 84]
minimas for accZ: [10 49 87]
Window 7 - Start Index subject: 2035
Window 7 - Start Index window: 300
No indices found for loc MAX pitch in the specified range 10-70.
Condition met: There are indices greater than 10 and less than 70 for loc MIN pitch. Possible toe-off
index loc min pitch: 43
Running Toe_Off detection
all instances [ 5 45 77]
index in list that meets criteria:  [1]
Toe off at:  45 convert to global coordinates: 2380
toe off at:  2380
```


Original DataFrame around Toe Off at 2330:2430 in df, subject 1
Pitch
TO
accZ
accX
220
200
180
160
140
2340
2360
2380
2400
2420
1
0
−1
−2
−3
2
1
0

.....

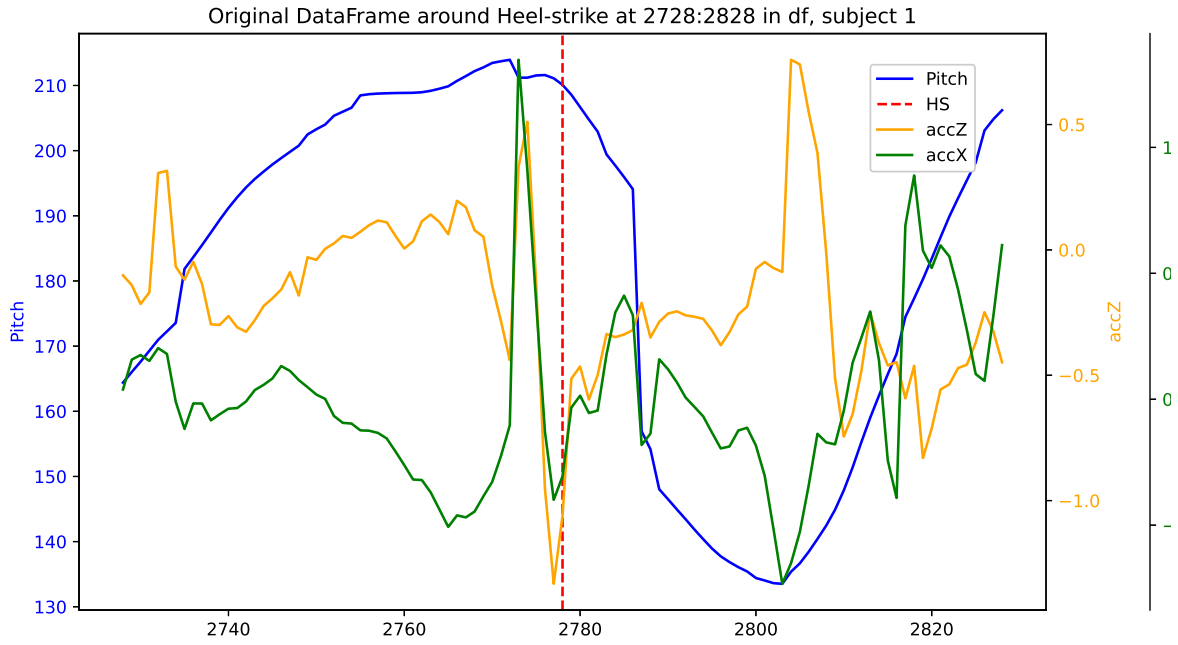
```
Maxima indices for Pitch window 15: [36 99]
Minima indices for Pitchwindow 15: [ 0 67]
Max Pitch: 213.71609497070312
Min Pitch: 133.62432861328125
Pitch Mag window 15 : 80.09177
```

figure


Window 15
accZ Values
accZ (Filtered and Normalized)
accZ Minima
accZ Maxima
Index
Window 15
accX Values
accX (Filtered and Normalized)
accZ Minima
accZ Maxima
Index
Window 15 - Filtered Pitch
Pitch Values
Pitch (Filtered)
Pitch Minima
Pitch Maxima
Index
Window 15 - Filtered accZ
accZ Values
accZ (Filtered)
accZ Minima
accZ Maxima
Index
Window 15 - Filtered accX
accX Values
accX (Filtered)
accX Minima
accX Maxima
Index

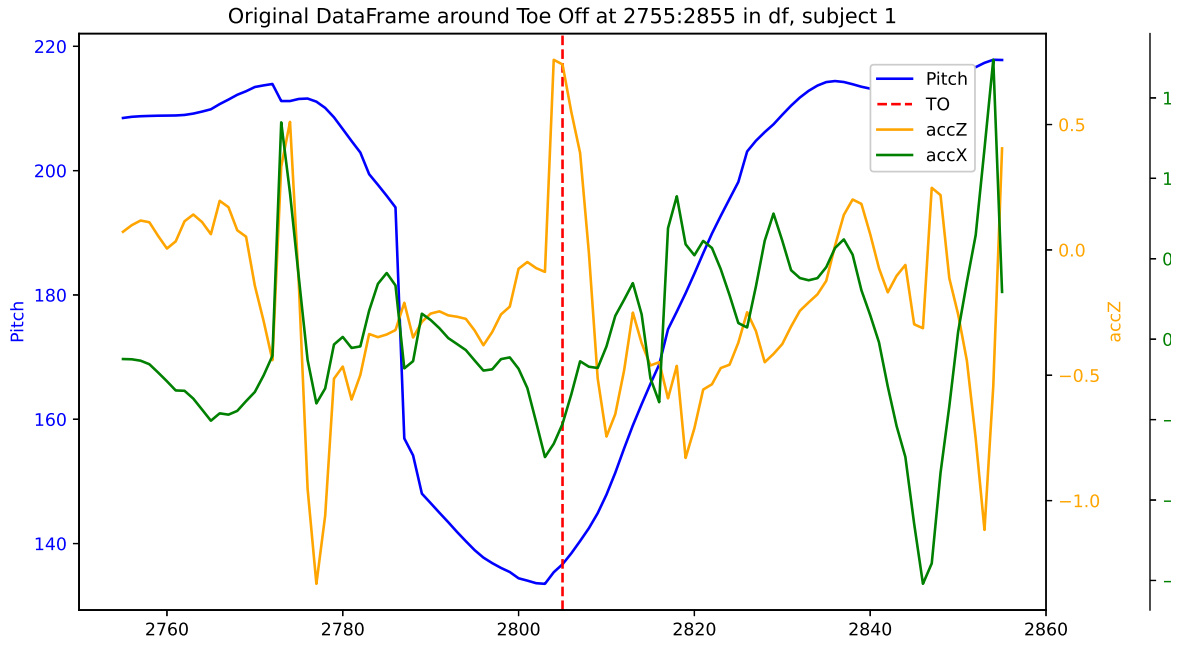
```
HEEL_STRIKE parameters:
maximas for Pitch: [36 99]
maximas for WIN accX: [12 39 49 84 94]
maximas for accX: [11 39 83 94]
minimas for WIN accX: [ 0 33 43 68 80 90 99]
minimas forr accX: [ 0 33 43 68 80 90 99]
minimas for WIN accZ: [ 7 43 75]
maximas for WIN accZ: [31 70 99]
minimas for Pitch: [ 0 67]
maximas for WIN accZ: [31 70 99]
maximas for accZ: [ 1 39 70 99]
minimas for accZ: [ 7 42 75]
Window 15 - Start Index subject: 2035
Window 15 - Start Index window: 700
Condition met: There are indices greater than 10 and less than 70 for loc MAX pitch. Possible Heel-strike!
index loc max pitch: 36
Condition met: There are indices greater than 10 and less than 70 for loc MIN pitch. Possible toe-off
index loc min pitch: 67
Running HS detection
accX MIN component of HS found inside WIN: 33
accX MAX component of HS found inside WIN: 39
Found both min and max for accX component near max for pitch. Probable HS. Search for accZ min.
these are accZ within range of accX max: [43]
Found candidates for accZ:  [43]
Heel Strike at w index:  43
Heel Strike at Index:  2778
```

figure


Original DataFrame around Heel-strike at 2728:2828 in df, subject 1
Pitch
HS
accZ
accX
Pitch
accZ
210
200
190
180
170
160
150
140
130
0.5
0.0
−0.5
−1.0
2740
2760
2780
2800
2820

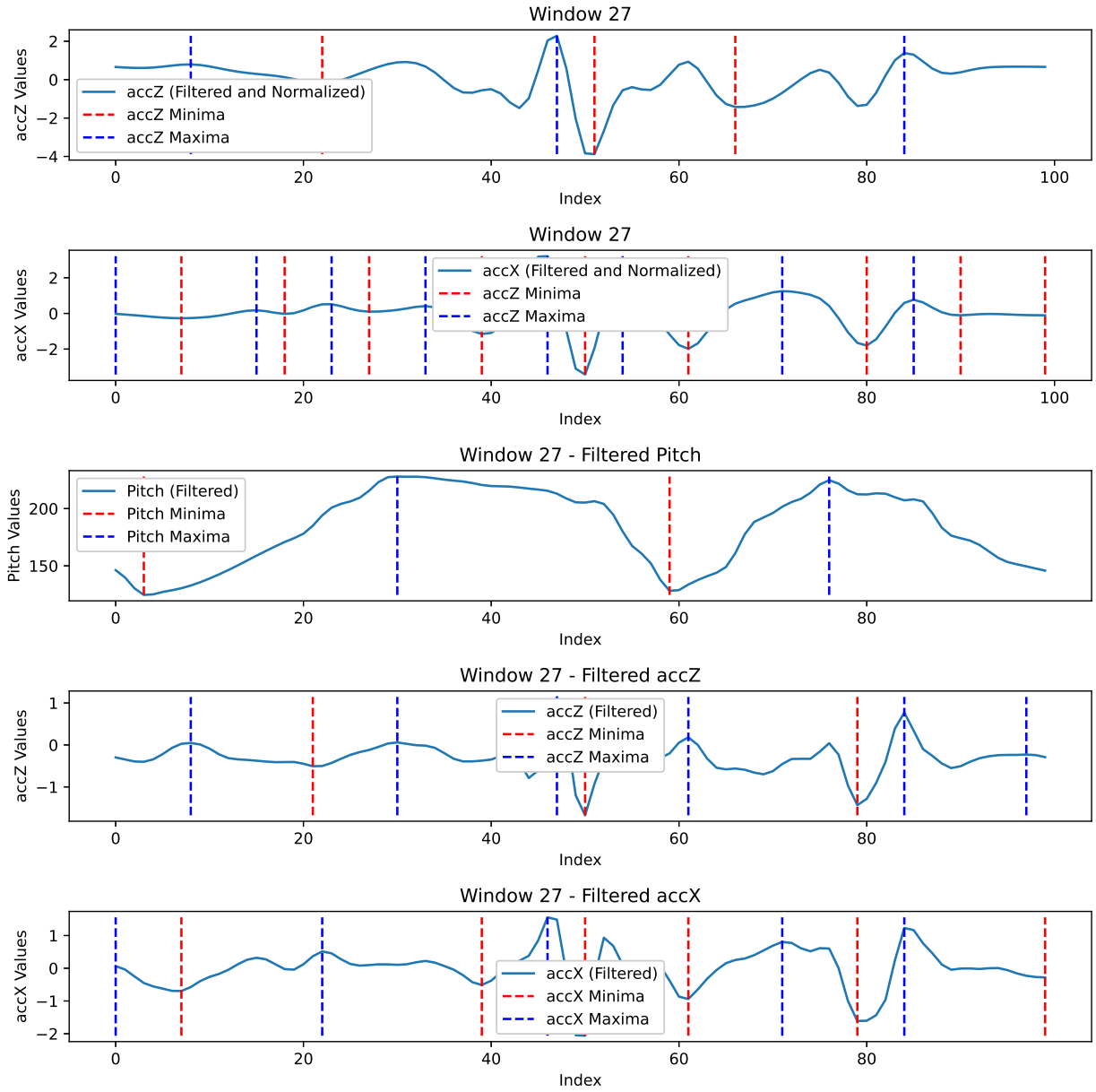
```
Running Toe_Off detection
all instances [31 70 99]
index in list that meets criteria:  [1]
Toe off at:  70 convert to global coordinates: 2805
toe off at:  2805
```

figure


Original DataFrame around Toe Off at 2755:2855 in df, subject 1
Pitch
TO
accZ
accX
220
200
180
160
140
0.5
0.0
−0.5
−1.0
2760
2780
2800
2820
2840
2860

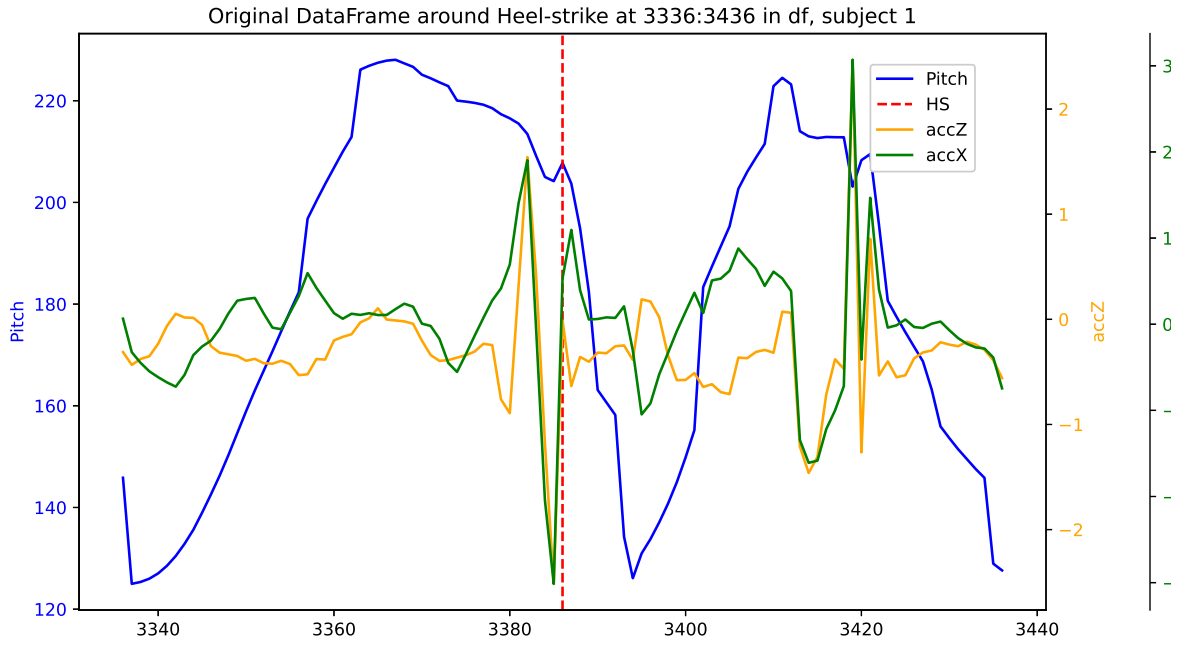
```
Maxima indices for Pitch window 27: [30 76]
Minima indices for Pitchwindow 27: [ 3 59]
Max Pitch: 227.4761962890625
Min Pitch: 125.34751892089844
Pitch Mag window 27 : 102.12868
```

figure


Window 27
accZ Values
accZ (Filtered and Normalized)
accZ Minima
accZ Maxima
Index
Window 27
accX Values
accX (Filtered and Normalized)
accZ Minima
accZ Maxima
Index
Window 27 - Filtered Pitch
Pitch Values
Pitch (Filtered)
Pitch Minima
Pitch Maxima
Index
Window 27 - Filtered accZ
accZ Values
accZ (Filtered)
accZ Minima
accZ Maxima
Index
Window 27 - Filtered accX
accX Values
accX (Filtered)
accX Minima
accX Maxima
Index

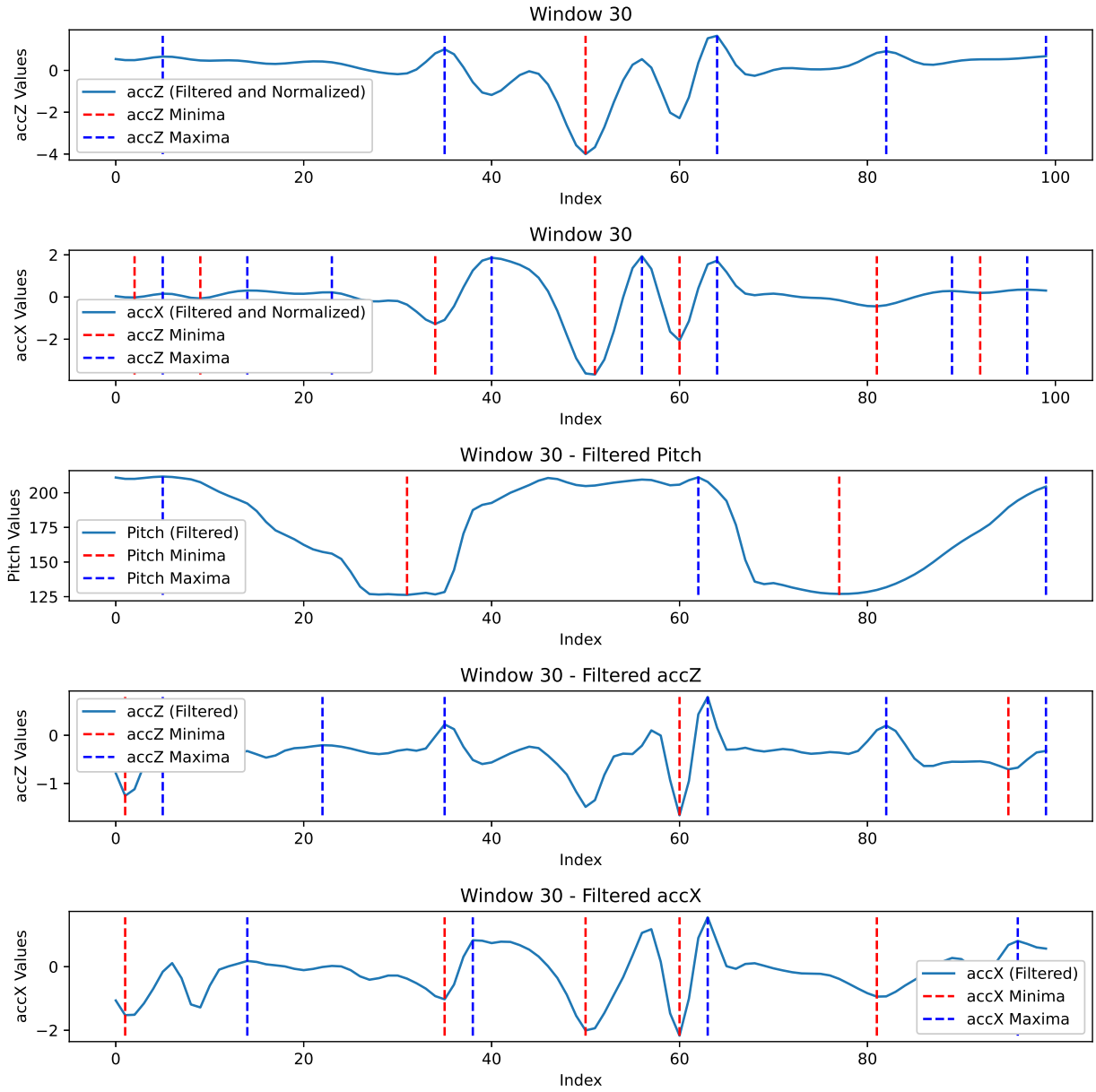
```
HEEL_STRIKE parameters:
maximas for Pitch: [30 76]
maximas for WIN accX: [ 0 15 23 33 46 54 71 85]
maximas for accX: [ 0 22 46 71 84]
minimas for WIN accX: [ 7 18 27 39 50 61 80 90 99]
minimas forr accX: [ 7 18 27 39 50 61 80 90 99]
minimas for WIN accZ: [22 51 66]
maximas for WIN accZ: [ 8 47 84]
minimas for Pitch: [ 3 59]
maximas for WIN accZ: [ 8 47 84]
maximas for accZ: [ 8 30 47 61 84 97]
minimas for accZ: [21 50 79]
Window 27 - Start Index subject: 2035
Window 27 - Start Index window: 1300
Condition met: There are indices greater than 10 and less than 70 for loc MAX pitch. Possible Heel-strike!
index loc max pitch: 30
Condition met: There are indices greater than 10 and less than 70 for loc MIN pitch. Possible toe-off
index loc min pitch: 59
Running HS detection
accX MIN component of HS found inside WIN: 27
accX MAX component of HS found inside WIN: 33
Found both min and max for accX component near max for pitch. Probable HS. Search for accZ min.
these are accZ within range of accX max: [51]
Found candidates for accZ:  [51]
Heel Strike at w index:  51
Heel Strike at Index:  3386
```

figure


Original DataFrame around Heel-strike at 3336:3436 in df, subject 1
Pitch
HS
accZ
accX
Pitch
accZ
220
200
180
160
140
120
2
1
0
−1
−2
3
2
1
0
3340
3360
3380
3400
3420
3440

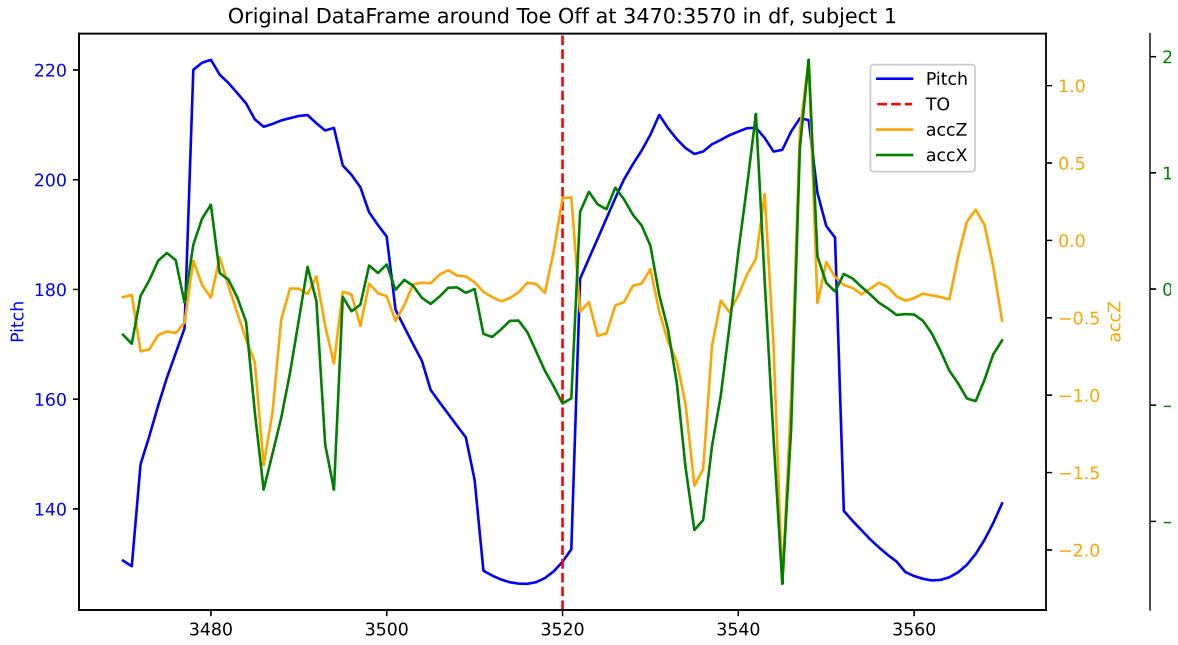
```
Running Toe_Off detection
Toe_Off_Detection: Could not find accZ MAX in range [ 54 , 79 ]

.......

Maxima indices for Pitch window 30: [ 5 62 99]
Minima indices for Pitchwindow 30: [31 77]
Max Pitch: 211.622314453125
Min Pitch: 126.34613800048828
Pitch Mag window 30 : 85.27618
```

figure


Window 30
accZ Values
accZ (Filtered and Normalized)
accZ Minima
accZ Maxima
Index
Window 30
accX Values
accX (Filtered and Normalized)
accZ Minima
accZ Maxima
Index
Window 30 - Filtered Pitch
Pitch Values
Pitch (Filtered)
Pitch Minima
Pitch Maxima
Index
Window 30 - Filtered accZ
accZ Values
accZ (Filtered)
accZ Minima
accZ Maxima
Index
Window 30 - Filtered accX
accX Values
accX (Filtered)
accX Minima
accX Maxima
Index

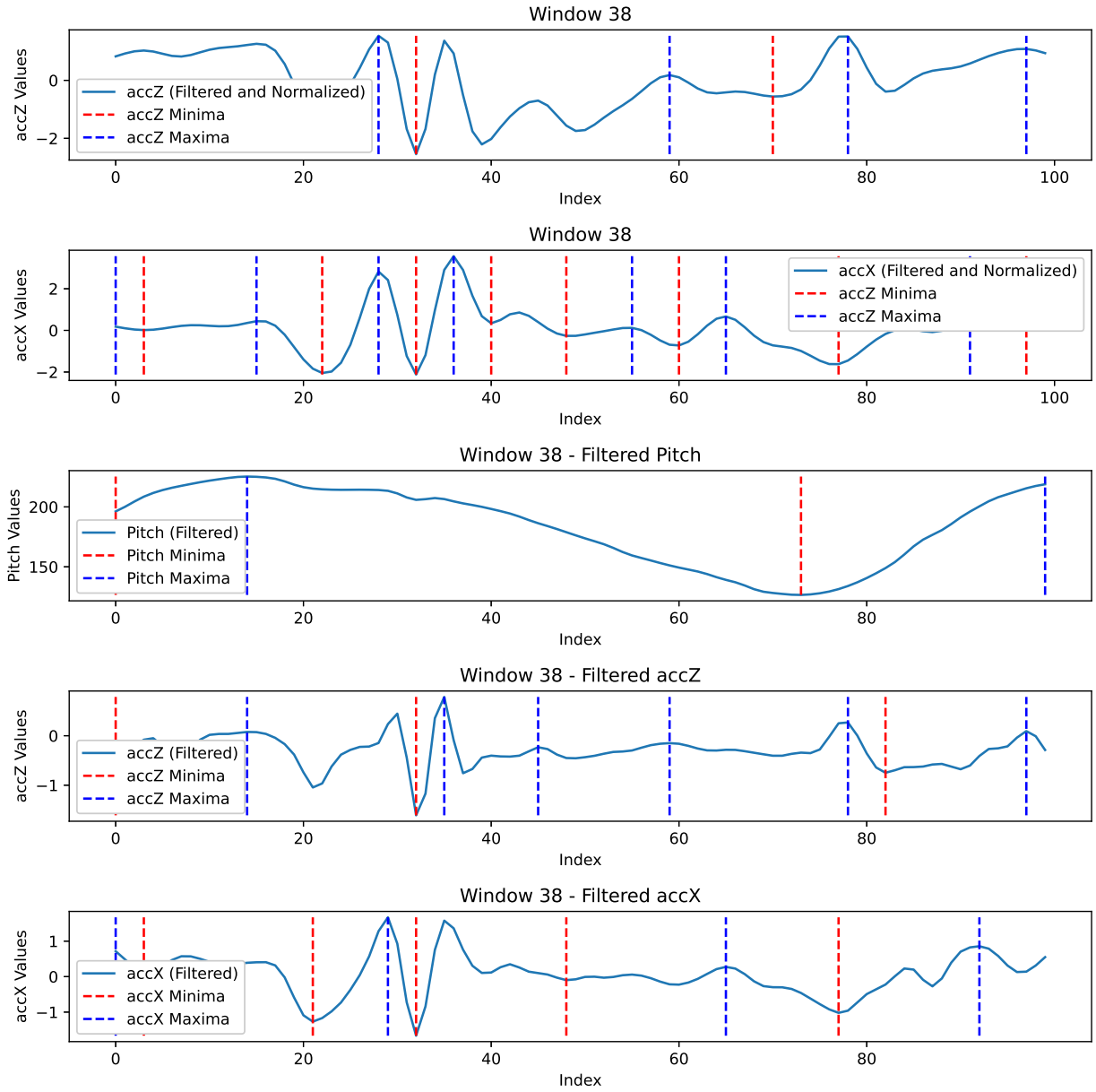
```
HEEL_STRIKE parameters:
maximas for Pitch: [ 5 62 99]
maximas for WIN accX: [ 5 14 23 40 56 64 89 97]
maximas for accX: [14 38 63 96]
minimas for WIN accX: [ 2  9 34 51 60 81 92]
minimas forr accX: [ 2  9 34 51 60 81 92]
minimas for WIN accZ: [50]
maximas for WIN accZ: [ 5 35 64 82 99]
minimas for Pitch: [31 77]
maximas for WIN accZ: [ 5 35 64 82 99]
maximas for accZ: [ 5 22 35 63 82 99]
minimas for accZ: [ 1 60 95]
Window 30 - Start Index subject: 2035
Window 30 - Start Index window: 1450
Condition met: There are indices greater than 10 and less than 70 for loc MAX pitch. Possible Heel-strike!
index loc max pitch: 62
Condition met: There are indices greater than 10 and less than 70 for loc MIN pitch. Possible toe-off
index loc min pitch: 31
Possibility of both TO and HS, in that order. Running TO first:
Running Toe_Off detection
all instances [ 5 35 64 82 99]
index in list that meets criteria:  [1]
Toe off at:  35 convert to global coordinates: 3520
toe off at:  3520
```

figure


Original DataFrame around Toe Off at 3470:3570 in df, subject 1
Pitch
TO
accZ
accX
220
200
180
160
140
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
2
1
0
3480
3500
3520
3540
3560

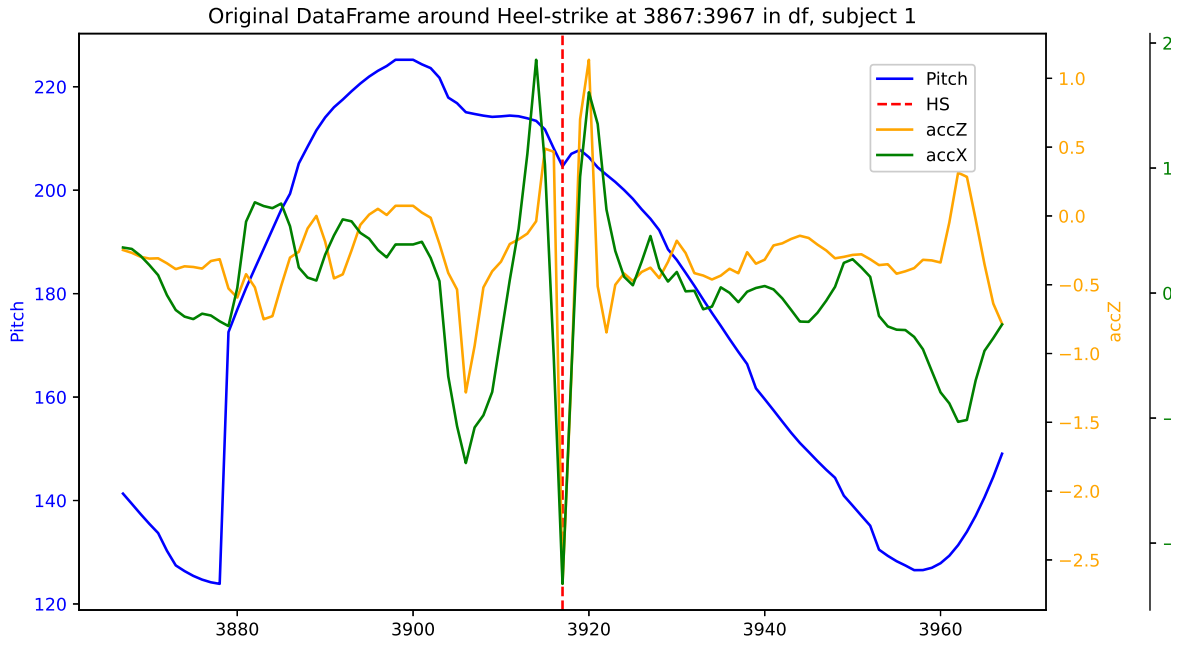
```
TO finished, now we want HS to run:
Running HS detection
accX MIN component of HS found inside WIN: 60
accX MAX component of HS found inside WIN: 64
Found both min and max for accX component near max for pitch. Probable HS. Search for accZ min.
these are accZ within range of accX max: []
Not able to find local min accZ near Pitch Max

.......

Maxima indices for Pitch window 38: [14 99]
Minima indices for Pitchwindow 38: [ 0 73]
Max Pitch: 225.21783447265625
Min Pitch: 126.54901123046875
Pitch Mag window 38 : 98.66882
```

figure


Window 38
accZ Values
accZ (Filtered and Normalized)
accZ Minima
accZ Maxima
Index
Window 38
accX Values
accX (Filtered and Normalized)
accZ Minima
accZ Maxima
Index
Window 38 - Filtered Pitch
Pitch Values
Pitch (Filtered)
Pitch Minima
Pitch Maxima
Index
Window 38 - Filtered accZ
accZ Values
accZ (Filtered)
accZ Minima
accZ Maxima
Index
Window 38 - Filtered accX
accX Values
accX (Filtered)
accX Minima
accX Maxima
Index

```
HEEL_STRIKE parameters:
maximas for Pitch: [14 99]
maximas for WIN accX: [ 0 15 28 36 55 65 91]
maximas for accX: [ 0 29 65 92]
minimas for WIN accX: [ 3 22 32 40 48 60 77 97]
minimas forr accX: [ 3 22 32 40 48 60 77 97]
minimas for WIN accZ: [32 70]
maximas for WIN accZ: [28 59 78 97]
minimas for Pitch: [ 0 73]
maximas for WIN accZ: [28 59 78 97]
maximas for accZ: [14 35 45 59 78 97]
minimas for accZ: [ 0 32 82]
Window 38 - Start Index subject: 2035
Window 38 - Start Index window: 1850
Condition met: There are indices greater than 10 and less than 70 for loc MAX pitch. Possible Heel-strike!
index loc max pitch: 14
No indices found for loc MIN pitch in the specified range 10-70.
Running HS detection
accX MIN component of HS found inside WIN: 22
accX MAX component of HS found inside WIN: 28
Found both min and max for accX component near max for pitch. Probable HS. Search for accZ min.
these are accZ within range of accX max: [32]
Found candidates for accZ:  [32]
Heel Strike at w index:  32
Heel Strike at Index:  3917
```

figure


Original DataFrame around Heel-strike at 3867:3967 in df, subject 1
Pitch
HS
accZ
accX
220
200
180
160
140
120
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
2
1
0
3880
3900
3920
3940
3960

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
```

```
df_Labeling_Copy = df.copy()
```

```
df_Labeling_Copy = df
```

df_Labeling_Copy = df

df_Labeling_Copy['Detection_50'] = 0

Remember to re-iterate labeling df after each subject. Remember to download Parquet

df.to_parquet('.parquet')

```
pd.set_option('display.max_rows', 100)
```

```
display(df_Labeling_Copy_0)
```

Quickly scroll through output and list obvious mistakes

```
def remove_events(df, bad_tags=None):

    for df_index in bad_tags:

        df.loc[df_index, 'Gait_Events'] = 0


BAD_ = [

18155,
18151,
18308,
18589,
19082,
19078,
19339,
19800,
```

```
19902,
20021,
20049,
20305,
20276,
20426,
21295,
21449,
21985,
22132,
22227,
22222,
22944,
23053,
23100,
23517,
23695,
23772,
23802,
23879,
23922,
24049,
25014,
25321



]

remove_events(df_Labeling_Copy_2,

              BAD_)

df_Labeling_Copy_1.loc[22218, 'Gait_Events'] = 'HS'
```

```
df_Labeling_Copy.loc[25490, 'Gait_Events']

df_Labeling_Copy_0.loc[2492, 'Gait_Events']

df_Labeling_Copy = df

df_Labeling_Copy_3 = df_Labeling_Copy_2

df_Labeling_Copy_1.loc[9037]

df.columns.dtypes

list(df_Labeling_Copy_1.columns)

df_Labeling_Copy_1 = df_Labeling_Copy_1[['Gait_Identification',
 'Gait_Events',
 'Measure number',
 'Timestamp',
 'q0',
 'q1',
 'q2',
 'q3',
 'MotionDeg',
 'Roll',
 'Pitch',
 'Yaw',
 'accX',
 'accY',
 'accZ',
 'gyrX',
 'gyrY',
 'gyrZ',
 'magX',
 'magY',
 'magZ',
 'Detection_50',]]

#df_Labeling_Copy_1['Gait_Events'] = df_Labeling_Copy_1['Gait_Events'].astype(object)
```

```
df_Labeling_Copy_3.to_csv('df_Labeling_Copy_3.csv')
```

Appendix D

Appendix: Code and Machine learning, Gait ID

D.1 UIA IMU: Feature importance analysis / feature selection using random forest and Support Vector Machine (SVM) models

```
    # This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a ...
   version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_ID_Walking_Gait_Dataset_8_1_2024
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_IMU_9ax_WG_Dataset_W_Calibration_Data_U_21_Des_23.csv
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_IMU_9ax_WG_Dataset_U_19_Des_23
/kaggle/input/lim-movement-dataset/leg_test_features.csv
/kaggle/input/lim-movement-dataset/leg_test_raw_labeled_4_11_complete.csv
/kaggle/input/lim-movement-dataset/TopMan_complete_ID_DetLabel_Ready_12_12.parquet
/kaggle/input/lim-movement-dataset/df_TopMan_635_steps__Detection_and_ID_Labeled_Original_Index.csv
/kaggle/input/lim-movement-dataset/leg_train_raw_unlabeled.csv
/kaggle/input/lim-movement-dataset/leg_train_raw_labeled_4_11_complete.csv
/kaggle/input/lim-movement-dataset/df_GE_Tags_13_12.csv
/kaggle/input/lim-movement-dataset/TopMan_ML_ID_0_29.csv
/kaggle/input/lim-movement-dataset/TopMan_complete_ID_DetLabel_Ready_8_12.csv
/kaggle/input/lim-movement-dataset/leg_test_raw.csv
/kaggle/input/lim-movement-dataset/leg_train_raw_clean.csv
/kaggle/input/lim-movement-dataset/Leg_Raw_Full_W_Engineered_Feats_26_11.csv
/kaggle/input/lim-movement-dataset/leg_Gait_Detection_complete_raw_labeled_20_11.csv
/kaggle/input/lim-movement-dataset/leg_train_features.csv
/kaggle/input/lim-movement-dataset/TopMan_complete_12_8_ID_Det_Ready_12_8.csv
/kaggle/input/lim-movement-dataset/leg_train_raw_labeled_12_11_complete.csv
/kaggle/input/lim-movement-dataset/TopMan_635_steps.csv
/kaggle/input/lim-movement-dataset/df_TopMan_635_steps__Detection_and_ID_Labeled_Reset_Index.csv
```

```
/kaggle/input/lim-movement-dataset/leg_train_raw.csv
```

```
pd.set_option('display.max_columns', 500)

df = pd.read_csv('/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_ID_Walking_Gait_Dataset_8_1_2024')

#df_1 = pd.read_csv('/kaggle/input/lim-movement-dataset/leg_test_raw_labeled_4_11_complete.csv')
#df_train.head()
```

Concatenate train and test

```
list(df.columns)
```

```
['Unnamed: 0',
 'Timestamp',
 'dt',
 'Subject_no',
 'q0',
 'q1',
 'q2',
 'q3',
 'Pitch',
 'Roll',
 'Yaw',
 'accX',
 'accY',
 'accZ',
 'gyrX',
 'gyrY',
 'gyrZ',
 'magX',
 'magY',
 'magZ']
```

```
df.isna().sum()
```

```
Unnamed: 0    0
```

```
Timestamp     0
dt            0
Subject_no    0
q0            0
q1            0
q2            0
q3            0
Pitch         0
Roll          0
Yaw           0
accX          0
accY          0
accZ          0
gyrX          0
gyrY          0
gyrZ          0
magX          0
magY          0
magZ          0
dtype: int64

df

       Unnamed: 0   Timestamp        dt  Subject_no        q0        q1  \
0               0    0.034056  0.034056           0  0.999981 -0.005295
1               1    0.068112  0.034056           0  0.999967 -0.007995
2               2    0.102168  0.034056           0  0.999951 -0.009922
3               3    0.142416  0.040248           0  0.999923 -0.012365
4               4    0.182664  0.040248           0  0.999878 -0.015495
...           ...         ...       ...         ...       ...       ...
38754       38754  132.178021  0.039903          12 -0.978575 -0.198041
38755       38755  132.217924  0.039903          12 -0.980173 -0.194980
38756       38756  132.257827  0.039903          12 -0.982184 -0.185914
38757       38757  132.297729  0.039902          12 -0.984022 -0.175340
38758       38758  132.337632  0.039903          12 -0.985002 -0.166676
```

```
              q2        q3     Pitch      Roll         Yaw      accX      accY  \
0     -0.003050 -0.000229 -0.349617 -0.024413   -0.606688  0.001269 -0.007415
1     -0.001483 -0.000205 -0.170067 -0.022105   -0.916112 -0.014983 -0.010159
2     -0.000083 -0.000221 -0.009767 -0.025253   -1.136978 -0.024849 -0.013850
3      0.000831 -0.000712  0.094222 -0.082716   -1.417035 -0.025351 -0.018286
4      0.000539 -0.001899  0.058380 -0.218530   -1.775726 -0.018526 -0.021769
...         ...       ...       ...       ...         ...       ...       ...
38754 -0.020191  0.052557  3.458979 -5.437485 -337.283108  0.181484  0.031365
38755 -0.004102  0.035017  1.243237 -3.843961 -337.540503  0.114871 -0.046202
38756  0.004001  0.027092  0.126920 -3.136616 -338.566529  0.002566 -0.098474
38757  0.004847  0.030541  0.067100 -3.544386 -339.795425 -0.110981 -0.107848
38758  0.002149  0.044560  0.608490 -5.078503 -340.818421 -0.178788 -0.079801

           accZ      gyrX      gyrY      gyrZ  magX  magY  magZ
0      0.007225 -0.015511 -0.128565 -0.013460    72   -42     7
1      0.003744  0.009013 -0.156497 -0.001248    73   -43     7
2     -0.001352  0.032911 -0.179508 -0.004759    72   -43     8
3     -0.007889  0.054869 -0.196698 -0.028749    72   -43     7
4     -0.014033  0.074107 -0.206459 -0.063698    72   -43     5
...         ...       ...       ...       ...   ...   ...   ...
38754  0.091784  0.130059 -0.464335  1.226585    56   -27    41
38755  0.108562  0.023426 -0.404651  0.938618    55   -27    40
38756  0.088519 -0.176852 -0.326740  0.471631    55   -28    41
38757  0.041200 -0.331125 -0.271021 -0.084528    56   -30    39
38758 -0.012090 -0.334927 -0.259580 -0.616271    56   -31    36

[38759 rows x 20 columns]

Columns = ['Subject_no',
 'q0',
 'q1',
 'q2',
 'q3',
 'Pitch',
```

```
 'Roll',
 'Yaw',
 'accX',
 'accY',
 'accZ',
 'gyrX',
 'gyrY',
 'gyrZ',
 'magX',
 'magY',
 'magZ']

df['Gait_Identification'] = df['Subject_no']

Columns_ID = [
 'Gait_Identification',
 'q0',
 'q1',
 'q2',
 'q3',
 'Roll',
 'Pitch',
 'Yaw',
 'accX',
 'accY',
 'accZ',
 'gyrX',
 'gyrY',
 'gyrZ',
 'magX',
 'magY',
 'magZ']

df_ID = df[Columns_ID]

df_ID
```

```
       Gait_Identification        q0        q1        q2        q3      Roll  \
0                        0  0.999981 -0.005295 -0.003050 -0.000229 -0.024413
1                        0  0.999967 -0.007995 -0.001483 -0.000205 -0.022105
2                        0  0.999951 -0.009922 -0.000083 -0.000221 -0.025253
3                        0  0.999923 -0.012365  0.000831 -0.000712 -0.082716
4                        0  0.999878 -0.015495  0.000539 -0.001899 -0.218530
...                    ...       ...       ...       ...       ...       ...
38754                   12 -0.978575 -0.198041 -0.020191  0.052557 -5.437485
38755                   12 -0.980173 -0.194980 -0.004102  0.035017 -3.843961
38756                   12 -0.982184 -0.185914  0.004001  0.027092 -3.136616
38757                   12 -0.984022 -0.175340  0.004847  0.030541 -3.544386
38758                   12 -0.985002 -0.166676  0.002149  0.044560 -5.078503

          Pitch         Yaw      accX      accY      accZ      gyrX      gyrY  \
0     -0.349617   -0.606688  0.001269 -0.007415  0.007225 -0.015511 -0.128565
1     -0.170067   -0.916112 -0.014983 -0.010159  0.003744  0.009013 -0.156497
2     -0.009767   -1.136978 -0.024849 -0.013850 -0.001352  0.032911 -0.179508
3      0.094222   -1.417035 -0.025351 -0.018286 -0.007889  0.054869 -0.196698
4      0.058380   -1.775726 -0.018526 -0.021769 -0.014033  0.074107 -0.206459
...         ...         ...       ...       ...       ...       ...       ...
38754  3.458979 -337.283108  0.181484  0.031365  0.091784  0.130059 -0.464335
38755  1.243237 -337.540503  0.114871 -0.046202  0.108562  0.023426 -0.404651
38756  0.126920 -338.566529  0.002566 -0.098474  0.088519 -0.176852 -0.326740
38757  0.067100 -339.795425 -0.110981 -0.107848  0.041200 -0.331125 -0.271021
38758  0.608490 -340.818421 -0.178788 -0.079801 -0.012090 -0.334927 -0.259580

           gyrZ  magX  magY  magZ
0     -0.013460    72   -42     7
1     -0.001248    73   -43     7
2     -0.004759    72   -43     8
3     -0.028749    72   -43     7
4     -0.063698    72   -43     5
...         ...   ...   ...   ...
38754  1.226585    56   -27    41
```

```
38755  0.938618    55   -27    40
38756  0.471631    55   -28    41
38757 -0.084528    56   -30    39
38758 -0.616271    56   -31    36

[38759 rows x 17 columns]
```

```
Feats_ID = [
 'q0',
 'q1',
 'q2',
 'q3',
 'Roll',
 'Pitch',
 'Yaw',
 'accX',
 'accY',
 'accZ',
 'gyrX',
 'gyrY',
 'gyrZ',
 'magX',
 'magY',
 'magZ']
```

```
Feats_ID
```

```
['q0',
 'q1',
 'q2',
 'q3',
 'Roll',
 'Pitch',
 'Yaw',
 'accX',
```

```
 'accY',
 'accZ',
 'gyrX',
 'gyrY',
 'gyrZ',
 'magX',
 'magY',
 'magZ']
```

Need to normalize feats

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
#df[Feats] = scaler.fit_transform(df[Feats])

df_ID[Feats_ID] = scaler.fit_transform(df_ID[Feats_ID])
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is ...
   required for this version of SciPy (detected version 1.24.3
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
/tmp/ipykernel_42/4253417415.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: ...
   https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_ID[Feats_ID] = scaler.fit_transform(df_ID[Feats_ID])
```

```
df_ID
```

	Gait_Identification	q0	q1	q2	q3	Roll	\
0	0	0.999997	0.519988	0.497156	0.501271	0.499981	
1	0	0.999990	0.518577	0.498107	0.501283	0.499988	
2	0	0.999982	0.517569	0.498957	0.501275	0.499979	

```
3                     0  0.999968  0.516292  0.499512  0.501027  0.499820
4                     0  0.999945  0.514656  0.499334  0.500427  0.499442
...                 ...       ...       ...       ...       ...       ...
38754                12  0.010397  0.419213  0.486752  0.527960  0.484943
38755                12  0.009598  0.420814  0.496517  0.519092  0.489370
38756                12  0.008592  0.425554  0.501435  0.515085  0.491335
38757                12  0.007673  0.431083  0.501949  0.516829  0.490202
38758                12  0.007183  0.435612  0.500312  0.523917  0.485940

          Pitch       Yaw      accX      accY      accZ      gyrX      gyrY  \
0      0.519295  0.499256  0.511461  0.481736  0.466147  0.405400  0.390924
1      0.520344  0.498969  0.503746  0.480409  0.463970  0.410807  0.386520
2      0.521280  0.498765  0.499062  0.478624  0.460782  0.416075  0.382893
3      0.521887  0.498505  0.498824  0.476479  0.456693  0.420916  0.380183
4      0.521678  0.498173  0.502064  0.474794  0.452850  0.425157  0.378644
...         ...       ...       ...       ...       ...       ...       ...
38754  0.541533  0.187301  0.597018  0.500492  0.519040  0.437492  0.337990
38755  0.528596  0.187063  0.565394  0.462977  0.529535  0.413984  0.347399
38756  0.522078  0.186112  0.512077  0.437695  0.516998  0.369831  0.359682
38757  0.521729  0.184973  0.458171  0.433162  0.487399  0.335820  0.368466
38758  0.524890  0.184026  0.425979  0.446727  0.454065  0.334982  0.370269

          gyrZ      magX      magY      magZ
0      0.419031  0.579545  0.388060  0.222222
1      0.421010  0.590909  0.373134  0.222222
2      0.420441  0.579545  0.373134  0.233333
3      0.416553  0.579545  0.373134  0.222222
4      0.410889  0.579545  0.373134  0.200000
...         ...       ...       ...       ...
38754  0.619995  0.397727  0.611940  0.600000
38755  0.573327  0.386364  0.611940  0.588889
38756  0.497646  0.386364  0.597015  0.600000
38757  0.407513  0.397727  0.567164  0.577778
38758  0.321337  0.397727  0.552239  0.544444
```

```
[38759 rows x 17 columns]
```

```
df_ID.shape
```

```
(38759, 17)
```

Function to segregate into test and training sets

```
def create_train_test_sets_ID(df, test_percentage):
    # Sort the DataFrame based on 'Gait_Identification'
    df_sorted = df.sort_values(by='Gait_Identification')

    # Get unique labels
    unique_labels = np.unique(df_sorted['Gait_Identification'].values)

    train_sequences, train_labels, test_sequences, test_labels = [], [], [], []

    for label in unique_labels:
        label_df = df_sorted[df_sorted['Gait_Identification'] == label]

        # Calculate the number of sequences for the test set
        test_size = int(len(label_df) * (test_percentage / 100))

        # Split the label-specific DataFrame into training and testing sets
        label_df_test = label_df.iloc[:test_size]
        label_df_train = label_df.iloc[test_size:]

        # Extract sequences and labels from the DataFrames
        train_sequences.append(label_df_train.drop(columns=['Gait_Identification']).values)
        train_labels.append(label_df_train['Gait_Identification'].values)
        test_sequences.append(label_df_test.drop(columns=['Gait_Identification']).values)
        test_labels.append(label_df_test['Gait_Identification'].values)

    # Concatenate sequences and labels
    x_train = np.concatenate(train_sequences)
```

```
    y_train = np.concatenate(train_labels)
    x_test = np.concatenate(test_sequences)
    y_test = np.concatenate(test_labels)

    return x_train, y_train, x_test, y_test

x_train_ID_Unseq, y_train_ID_Unseq, x_test_ID_Unseq, y_test_ID_Unseq = create_train_test_sets_ID(df_ID, 25)

unique_labels_train = np.unique(y_train_ID_Unseq)
unique_labels_test = np.unique(y_test_ID_Unseq)

print("Unique labels in training set:", unique_labels_train)
print("Unique labels in testing set:", unique_labels_test)

print(x_test_ID_Unseq.shape)
print(x_train_ID_Unseq.shape)
print(y_test_ID_Unseq.shape)
print(y_train_ID_Unseq.shape)
```



```
Unique labels in training set: [ 0  1  2  3  4  5  6  7  8  9 10 11 12]
Unique labels in testing set: [ 0  1  2  3  4  5  6  7  8  9 10 11 12]
(9686, 16)
(29073, 16)
(9686,)
(29073,)
```

Want long sequences for the ID task. Want sequences to have uniform label.

```
def create_sequences_with_constant_label(x_data, y_data, sequence_length):
    x_seqs, y_seqs = [], []

    if len(y_data) >= sequence_length:
        for i in range(0, len(y_data) - sequence_length + 1, sequence_length):
            sequence_x = x_data[i:i + sequence_length]
            sequence_y = y_data[i:i + sequence_length]
```

```
            # Check if all y values in the sequence are the same
            if len(set(sequence_y)) == 1:
                x_seqs.append(sequence_x)
                y_seqs.append(sequence_y[0])  # Take the constant y value

        print(f"Number of sequences created: {len(x_seqs)}")
        if len(x_seqs) > 0:
            print(f"Example sequence_x: {x_seqs[0]}")
            print(f"Example sequence_y: {y_seqs[0]}")

    return np.array(x_seqs), np.array(y_seqs)


# run function
sequence_length = 50 #The function takes 1/2 sequence length as step, creating seqs with 50% overlap
x_train_ID, y_train_ID = create_sequences_with_constant_label(x_train_ID_Unseq, y_train_ID_Unseq, sequence_length)
x_test_ID, y_test_ID = create_sequences_with_constant_label(x_test_ID_Unseq, y_test_ID_Unseq, sequence_length)

# Print the shapes of the resulting sequences
print("Number of sequences in x_train_ID:", len(x_train_ID))
print("Number of sequences in y_train_ID:", len(y_train_ID))
print("Number of sequences in x_test_ID:", len(x_test_ID))
print("Number of sequences in y_test_ID:", len(y_test_ID))
```

```
Number of sequences created: 569
Example sequence_x: [[0.69621901 0.87066201 0.6430768  0.20366617 0.46106243 0.96470997
  0.58282576 0.5135397  0.45130431 0.45656518 0.42154693 0.50155211
  0.39407969 0.625      0.26865672 0.48888889]
 [0.62976679 0.88079044 0.42430882 0.16280109 0.23108286 0.86326585
  0.57996734 0.45857105 0.47279934 0.53407329 0.391842   0.33188633
  0.44138108 0.64772727 0.40298507 0.56666667]
 [0.62796305 0.87569043 0.426741   0.15677849 0.22178998 0.86712356
  0.57693853 0.49651272 0.42509056 0.51826335 0.37312726 0.31176079
  0.38286915 0.65909091 0.3880597  0.53333333]
```

```
[0.62537804 0.86977937 0.43097974 0.14966687 0.21124266 0.87242201
 0.57315021 0.54878976 0.40147007 0.48600698 0.34969288 0.28984026
 0.34977987 0.67045455 0.3880597  0.53333333]
[0.62332406 0.86319583 0.43413466 0.14256495 0.20003106 0.87530544
 0.5690605  0.59353872 0.41707251 0.45094473 0.34672185 0.273557
 0.35641204 0.67045455 0.37313433 0.5       ]
[0.62212469 0.85691701 0.43351775 0.13675044 0.18955452 0.87372326
 0.56575162 0.61427218 0.46510785 0.42457395 0.38336057 0.27609779
 0.39102175 0.67045455 0.3880597  0.47777778]
[0.62093271 0.85296756 0.42811677 0.13369288 0.18197698 0.8690126
 0.56450001 0.60786474 0.52099901 0.40783156 0.45889822 0.3109669
 0.42401913 0.68181818 0.3880597  0.48888889]
[0.61811416 0.85369499 0.41968109 0.13456376 0.17922504 0.86547133
 0.56597693 0.58418262 0.55636294 0.39132455 0.54947244 0.38409804
 0.42746093 0.67045455 0.41791045 0.48888889]
[0.61254657 0.86083212 0.41187942 0.14012959 0.18330404 0.86722202
 0.57036239 0.55770146 0.55464704 0.36433593 0.61790715 0.48781722
 0.39431364 0.68181818 0.44776119 0.52222222]
[0.60457488 0.87441242 0.40859171 0.15051167 0.19698023 0.87509871
 0.57770109 0.53637873 0.51969844 0.3262905  0.6318985  0.60051932
 0.34562615 0.68181818 0.44776119 0.53333333]
[0.59691541 0.89172756 0.41132142 0.16469514 0.22298329 0.88394513
 0.58750239 0.51534015 0.47286085 0.29213907 0.58087185 0.69318671
 0.32019797 0.64772727 0.46268657 0.61111111]
[0.59331349 0.90908372 0.41770753 0.18114926 0.26025819 0.88547625
 0.5980346  0.48062998 0.44070463 0.28673855 0.48223514 0.74027156
 0.35143941 0.625      0.46268657 0.63333333]
[0.59608065 0.92305751 0.42342543 0.19748155 0.29680351 0.87601914
 0.60599996 0.4219417  0.44051298 0.330491   0.3733548  0.72996456
 0.4443793  0.60227273 0.43283582 0.67777778]
[0.6052238  0.9309857  0.42486653 0.21047507 0.31805436 0.85932991
 0.60910469 0.3464365  0.4715119  0.42459576 0.2933575  0.66890557
 0.56705446 0.59090909 0.44776119 0.73333333]
[0.61976786 0.93290667 0.41905882 0.2199799  0.32302365 0.83721858
```

```
 0.60815452 0.28322288 0.51629462 0.54528282 0.26447805 0.57901434
 0.6632342  0.56818182 0.46268657 0.73333333]
[0.633845   0.9309331  0.40970872 0.22579791 0.31999854 0.8165036
 0.60579876 0.27252641 0.55110891 0.65157493 0.2829104  0.48797783
 0.68111958 0.54545455 0.47761194 0.74444444]
[0.64332445 0.92735497 0.40222027 0.2275     0.31480169 0.80347382
 0.60354722 0.34320534 0.55892037 0.70335845 0.32376882 0.4179992
 0.60245843 0.54545455 0.47761194 0.76666667]
[0.64659907 0.92422398 0.40110514 0.2252486  0.31057529 0.80206059
 0.60197455 0.49115202 0.53798543 0.68048044 0.35684583 0.37809336
 0.45502976 0.53409091 0.47761194 0.76666667]
[0.6441862  0.92266522 0.40801581 0.22023899 0.30927322 0.81226421
 0.60125146 0.67289104 0.501539   0.59277752 0.36391544 0.36319731
 0.30032021 0.54545455 0.46268657 0.74444444]
[0.63864151 0.92191286 0.42092157 0.21376758 0.31100365 0.82997601
 0.60102178 0.82107994 0.46934574 0.47575953 0.34779305 0.35978382
 0.20255617 0.55681818 0.46268657 0.75555556]
[0.63332453 0.92067377 0.43459863 0.20731154 0.31381071 0.84808712
 0.60068944 0.87563985 0.45632563 0.37427084 0.32846706 0.35446095
 0.19635964 0.54545455 0.37313433 0.76666667]
[0.63043857 0.91834541 0.44368205 0.2018178  0.31391596 0.86056831
 0.59970498 0.8139895  0.46478899 0.32256746 0.32926736 0.34088232
 0.27114621 0.54545455 0.34328358 0.74444444]
[0.63022637 0.91526162 0.44495272 0.19779593 0.30850647 0.86448132
 0.59804729 0.6632202  0.48440582 0.33066309 0.36167827 0.32168286
 0.38034404 0.56818182 0.34328358 0.71111111]
[0.63023017 0.91275312 0.43957014 0.19548129 0.29984555 0.86219893
 0.59671917 0.48692506 0.49941509 0.38303819 0.41779232 0.30516929
 0.46887725 0.59090909 0.37313433 0.7       ]
[0.62850685 0.90846712 0.43278969 0.19065807 0.28624834 0.86161457
 0.59463496 0.35384754 0.49847329 0.44918855 0.47451833 0.29935736
 0.50225532 0.59090909 0.3880597  0.71111111]
[0.62554112 0.90450813 0.4260129  0.18593472 0.27309011 0.86162527
 0.59287328 0.30562414 0.48126837 0.49966823 0.50661556 0.30707154
```

```
 0.48101417 0.61363636 0.3880597  0.66666667]
[0.6212464  0.90141922 0.42223166 0.18138975 0.26257366 0.86463115
 0.59155051 0.34062962 0.45819348 0.51920415 0.50066144 0.3247222
 0.43422594 0.625      0.35820896 0.66666667]
[0.61713614 0.89901145 0.42198546 0.17723518 0.25503268 0.86978915
 0.59037911 0.42094593 0.44358708 0.51087728 0.46194116 0.34489353
 0.39847164 0.63636364 0.3880597  0.63333333]
[0.62931115 0.88559918 0.42526169 0.16729491 0.23949415 0.86392048
 0.58247035 0.45021466 0.51605197 0.52838193 0.39058856 0.3491667
 0.49296474 0.64772727 0.40298507 0.57777778]
[0.62590675 0.8904053  0.42924544 0.17029796 0.24738091 0.86963618
 0.58482213 0.47070731 0.52951598 0.50971317 0.37283516 0.36546163
 0.50693912 0.64772727 0.40298507 0.58888889]
[0.62060128 0.89477473 0.43388061 0.17219169 0.25425678 0.8775719
 0.5871125  0.50214388 0.5061941  0.49533788 0.35738727 0.38103334
 0.47418651 0.63636364 0.40298507 0.6       ]
[0.61611805 0.89814785 0.43589938 0.17385465 0.25919489 0.88257398
 0.58914867 0.518106   0.46188697 0.49943211 0.36510653 0.39236662
 0.41483994 0.63636364 0.41791045 0.58888889]
[0.63183396 0.88064052 0.44420858 0.16083488 0.24040638 0.87737111
 0.57810129 0.4906717  0.45508745 0.53259204 0.38977778 0.34122103
 0.39612026 0.625      0.3880597  0.55555556]
[0.63037578 0.87564139 0.44691719 0.15513455 0.23143178 0.88098817
 0.57489659 0.51784847 0.43724464 0.50840865 0.36858561 0.31846301
 0.37769659 0.63636364 0.3880597  0.53333333]
[0.62905537 0.86958859 0.44921693 0.1487334  0.22070214 0.88336084
 0.57106627 0.55700393 0.44422511 0.47468358 0.35871527 0.29225634
 0.37254675 0.63636364 0.3880597  0.51111111]
[0.62769975 0.86329027 0.44913435 0.14255053 0.20894129 0.88300496
 0.56739286 0.59845581 0.47614037 0.44103908 0.3816981  0.27802605
 0.37882103 0.65909091 0.3880597  0.51111111]
[0.62530163 0.85854444 0.44578126 0.13786785 0.19814502 0.88090386
 0.56504418 0.62893547 0.51963277 0.41089253 0.44485634 0.2940174
 0.38827668 0.64772727 0.37313433 0.48888889]
```

```
[0.62083868 0.85771186 0.439543   0.13627797 0.19106092 0.87973468
 0.56513371 0.6371829  0.55334697 0.38149283 0.53400957 0.35193313
 0.38998426 0.67045455 0.43283582 0.47777778]
[0.61397707 0.86281091 0.43228928 0.13931058 0.19086486 0.88220785
 0.56843504 0.61910984 0.55839297 0.34963313 0.61684479 0.44910394
 0.37701373 0.65909091 0.44776119 0.51111111]
[0.60527036 0.8745529  0.42695879 0.14791509 0.20134432 0.88898424
 0.57543292 0.57883416 0.52895212 0.31827405 0.65636702 0.56639053
 0.35358955 0.64772727 0.46268657 0.53333333]
[0.59632598 0.89122843 0.42660375 0.16150476 0.22630687 0.8970746
 0.58588652 0.52488665 0.47720015 0.29877241 0.62857127 0.67342611
 0.33769639 0.63636364 0.49253731 0.58888889]
[0.59036044 0.90898997 0.43126907 0.17800222 0.26565706 0.89948318
 0.59788318 0.46490092 0.42888306 0.30655814 0.53563138 0.73938631
 0.35464124 0.60227273 0.49253731 0.65555556]
[0.59059127 0.92401979 0.43772613 0.19472061 0.30754006 0.8916538
 0.6076042  0.40356971 0.41065109 0.35252886 0.40730936 0.74481887
 0.4220451  0.59090909 0.47761194 0.66666667]
[0.5987435  0.9333016  0.44125239 0.20862604 0.33360616 0.87549125
 0.61179088 0.34589774 0.43527553 0.43515333 0.28878452 0.68939501
 0.53372157 0.55681818 0.49253731 0.71111111]
[0.61429409 0.93590374 0.43741242 0.21855114 0.33903332 0.85350114
 0.61059369 0.3027065  0.4930588  0.53780043 0.22019623 0.59208927
 0.65365174 0.54545455 0.49253731 0.75555556]
[0.61497543 0.89676952 0.42351244 0.17378627 0.2497423  0.8741557
 0.58907322 0.49634115 0.44678205 0.4906069  0.4107289  0.36061191
 0.39612313 0.63636364 0.37313433 0.62222222]
[0.63228623 0.93316031 0.42692325 0.2246261  0.33228219 0.82956934
 0.60685446 0.29218582 0.5552048  0.63288249 0.21808704 0.48357186
 0.72760369 0.51136364 0.52238806 0.76666667]
[0.65527166 0.92129621 0.40525903 0.2251314  0.31156553 0.79852445
 0.5999648  0.43237243 0.5735524  0.69188544 0.33611747 0.34204998
 0.58966743 0.51136364 0.55223881 0.75555556]
[0.65550064 0.91740518 0.40472592 0.22020897 0.30493885 0.80144694
```

```
  0.59830209 0.57292762 0.51671506 0.63169431 0.38333585 0.32862675
  0.4055105  0.52272727 0.52238806 0.74444444]
 [0.64914528 0.91616781 0.41412372 0.21307818 0.30293379 0.817778
  0.59778816 0.71461311 0.44728993 0.5263797  0.3908733  0.34000572
  0.22875922 0.53409091 0.50746269 0.72222222]]
Example sequence_y: 0
Number of sequences created: 181
Example sequence_x: [[0.99999681 0.51998848 0.4971561  0.50127086 0.49998149 0.51929545
  0.49925617 0.51146131 0.48173613 0.46614732 0.4054002  0.39092372
  0.41903051 0.57954545 0.3880597  0.22222222]
 [0.7528712  0.84244759 0.74465735 0.27168237 0.50592658 0.95814129
  0.59869779 0.49637072 0.48263491 0.39477749 0.41683965 0.34577659
  0.45022384 0.625      0.31343284 0.36666667]
 [0.75327726 0.84288751 0.739642   0.26904405 0.50339642 0.95742627
  0.59624222 0.37809017 0.49023701 0.4433044  0.46466554 0.34878516
  0.49408945 0.61363636 0.31343284 0.36666667]
 [0.75345499 0.84368613 0.73311931 0.26567274 0.50017689 0.95613432
  0.59340447 0.3325057  0.48379656 0.47882477 0.50039595 0.36001351
  0.48961965 0.61363636 0.34328358 0.34444444]
 [0.75209348 0.84428253 0.72879412 0.26201484 0.4979679  0.95698012
  0.59185458 0.35795625 0.46664994 0.49074845 0.50500547 0.3777236
  0.4559713  0.61363636 0.32835821 0.31111111]
 [0.74993409 0.84403918 0.72771103 0.25869701 0.49705985 0.9602801
  0.59137195 0.42226988 0.45051802 0.48233936 0.47457454 0.39596008
  0.42335523 0.61363636 0.32835821 0.28888889]
 [0.74836211 0.84306736 0.72913302 0.25678581 0.49718199 0.96403019
  0.59151284 0.48413111 0.44789358 0.46623404 0.42247786 0.40825184
  0.41392982 0.60227273 0.31343284 0.28888889]
 [0.74861269 0.84201195 0.7310828  0.25703419 0.49761625 0.96543816
  0.59164448 0.51502861 0.46358057 0.45622744 0.37166997 0.41129462
  0.43071631 0.60227273 0.31343284 0.26666667]
 [0.75268336 0.84258816 0.72932632 0.26088563 0.49713091 0.95881826
  0.59066456 0.5101781  0.4907593  0.45995039 0.34197117 0.4062253
  0.45932085 0.60227273 0.29850746 0.26666667]
```

```
[0.75820136 0.84192103 0.72596959 0.26385351 0.49501117 0.95173603
 0.58793811 0.48507792 0.5142739  0.4759356  0.33989909 0.39684942
 0.47907867 0.57954545 0.28358209 0.24444444]
[0.76322772 0.8405571  0.72226047 0.26534988 0.49224795 0.94597702
 0.58491236 0.46280535 0.51941374 0.4958381  0.35670944 0.38629977
 0.47568792 0.56818182 0.29850746 0.24444444]
[0.76673873 0.83807107 0.71959695 0.26452584 0.48919067 0.94375509
 0.58173658 0.46076385 0.50100123 0.50981765 0.37530615 0.37468104
 0.44832499 0.56818182 0.29850746 0.23333333]
[0.76859458 0.83423887 0.7188024  0.26143599 0.4860912  0.9456523
 0.57823146 0.48346202 0.46725409 0.51172517 0.38161549 0.35957323
 0.40868724 0.56818182 0.31343284 0.24444444]
[0.76969218 0.82909365 0.71926362 0.25683778 0.482696   0.94992482
 0.57385959 0.52294262 0.43590734 0.5013302  0.37370856 0.33926192
 0.37391146 0.55681818 0.29850746 0.21111111]
[0.77096765 0.8228749  0.71970223 0.2514031  0.47850108 0.95407382
 0.56815372 0.56444421 0.42464646 0.48287106 0.36345517 0.31654874
 0.35757134 0.55681818 0.31343284 0.18888889]
[0.77269244 0.81637014 0.71877706 0.2455986  0.47324194 0.95584587
 0.56144617 0.59359267 0.44127355 0.46144213 0.36973412 0.300307
 0.36287826 0.54545455 0.29850746 0.2       ]
[0.77407231 0.81176727 0.71567105 0.24041791 0.46785883 0.9543714
 0.5559476  0.60219538 0.47920265 0.43985641 0.40683156 0.30309041
 0.38091824 0.54545455 0.31343284 0.18888889]
[0.77357303 0.81110061 0.71098034 0.23654815 0.46403036 0.95189509
 0.55369122 0.59090438 0.52080512 0.41803546 0.47427259 0.33542482
 0.3950162  0.54545455 0.32835821 0.16666667]
[0.76943383 0.81531173 0.70664241 0.23415451 0.4636393  0.95219529
 0.55536832 0.56789172 0.54666872 0.39512426 0.55373309 0.39952562
 0.39006348 0.55681818 0.35820896 0.17777778]
[0.76051303 0.82399682 0.70549944 0.23352004 0.46855873 0.95818075
 0.56139846 0.5437658  0.54560994 0.37261865 0.61523156 0.48582681
 0.36292861 0.57954545 0.35820896 0.2       ]
[0.74741097 0.83520202 0.70973289 0.23511445 0.47928573 0.96908284
```

```
 0.57284189 0.52479415 0.52006645 0.35605918 0.63017067 0.57449851
 0.32810594 0.59090909 0.40298507 0.24444444]
[0.73250875 0.84656647 0.71933435 0.23950854 0.49393981 0.97860202
 0.59128465 0.50831675 0.48424713 0.35375043 0.585187   0.64173796
 0.31358178 0.60227273 0.41791045 0.26666667]
[0.71975027 0.85635415 0.73160944 0.24721105 0.50854079 0.97731917
 0.61244892 0.4841678  0.45640641 0.37289997 0.48995651 0.66831782
 0.34653067 0.625      0.41791045 0.3       ]
[0.71313761 0.86367282 0.74215194 0.25759254 0.51916328 0.96532618
 0.62513516 0.44280415 0.44956892 0.41526458 0.37465175 0.6467936
 0.43514011 0.61363636 0.43283582 0.35555556]
[0.71624935 0.86854194 0.74527061 0.26925927 0.52321588 0.94994749
 0.62546867 0.38592178 0.46547806 0.47476844 0.27762875 0.58434306
 0.55733868 0.60227273 0.37313433 0.4       ]
[0.72685501 0.87258283 0.74014152 0.28187937 0.52251664 0.93192244
 0.61916683 0.33208312 0.49451335 0.53845531 0.22885468 0.50006799
 0.6654406  0.625      0.37313433 0.41111111]
[0.74031059 0.87509972 0.72939415 0.29253426 0.51869287 0.91357644
 0.61140742 0.31146635 0.52113982 0.59035513 0.23703246 0.41787073
 0.70731115 0.61363636 0.37313433 0.43333333]
[0.75196824 0.87536818 0.71785414 0.29847894 0.5134701  0.89967441
 0.604971   0.3503356  0.53183177 0.61652489 0.28685074 0.3577908
 0.65407502 0.61363636 0.34328358 0.44444444]
[0.75234464 0.84188274 0.74718603 0.2722426  0.50693756 0.95945326
 0.59982391 0.65127798 0.46809289 0.3551673  0.37989584 0.34738479
 0.36568295 0.61363636 0.28358209 0.36666667]
[0.75192478 0.84526257 0.74602381 0.27534503 0.50853248 0.95401225
 0.60144476 0.78257244 0.4594615  0.3483704  0.36716508 0.34719428
 0.27617516 0.625      0.29850746 0.41111111]
[0.7533211  0.8515236  0.74053254 0.28133733 0.50979586 0.94130417
 0.60199742 0.83449071 0.4668822  0.38808231 0.37459991 0.34090378
 0.22897368 0.63636364 0.34328358 0.41111111]
[0.7556951  0.8587405  0.73348061 0.28918653 0.51102413 0.92518333
 0.60217758 0.78389569 0.49081879 0.46897797 0.38389612 0.33070813
```

```
 0.25684356 0.63636364 0.34328358 0.4       ]
[0.75887803 0.84128751 0.7465008  0.2784912  0.50663941 0.95130197
 0.59749338 0.41187424 0.48408115 0.4091454  0.45853242 0.34567668
 0.49267092 0.60227273 0.31343284 0.33333333]
[0.75790802 0.84214693 0.74115118 0.27445074 0.5040898  0.9519405
 0.59558906 0.36659447 0.48300568 0.455927   0.50630193 0.35697081
 0.4754291  0.60227273 0.29850746 0.32222222]
[0.75562472 0.84275124 0.73720078 0.26975274 0.5020481  0.95449889
 0.59443525 0.37528871 0.47257812 0.49178225 0.52208604 0.37922928
 0.4345379  0.59090909 0.34328358 0.3       ]
[0.75261146 0.84281738 0.73597924 0.26562532 0.50119803 0.95870229
 0.59437277 0.41523929 0.45961265 0.50699079 0.49979361 0.40390951
 0.40354481 0.61363636 0.34328358 0.28888889]
[0.75050755 0.84233772 0.73671597 0.26324798 0.50119912 0.9624075
 0.59478388 0.45825344 0.453707   0.50239907 0.45112811 0.42161824
 0.40253503 0.59090909 0.32835821 0.27777778]
[0.75046775 0.84152189 0.73779402 0.26293124 0.50126504 0.96377394
 0.59474724 0.48474336 0.46126213 0.48824463 0.39839018 0.42691099
 0.42933099 0.60227273 0.31343284 0.25555556]
[0.75309279 0.84043252 0.73739916 0.26413476 0.50041018 0.96190591
 0.59317989 0.48972465 0.48088309 0.47809596 0.36229232 0.42026581
 0.46491245 0.59090909 0.32835821 0.24444444]
[0.75808254 0.83990579 0.73418769 0.2667626  0.49848975 0.9556338
 0.59036589 0.48030249 0.50315339 0.4810593  0.35168686 0.40637256
 0.48717283 0.57954545 0.32835821 0.24444444]
[0.76307547 0.83899162 0.7302574  0.2685748  0.49596578 0.94956928
 0.58728677 0.46883102 0.51537576 0.4967293  0.36057477 0.39019263
 0.48365044 0.59090909 0.32835821 0.23333333]
[0.76703242 0.8369107  0.7269727  0.26829864 0.49293748 0.94639538
 0.5839645  0.46647906 0.50888394 0.51588484 0.37351481 0.37357873
 0.45662494 0.56818182 0.32835821 0.21111111]
[0.76990086 0.8332045  0.72505929 0.2657537  0.48943081 0.94678689
 0.58006332 0.47947153 0.48475055 0.5263871  0.37602785 0.35480756
 0.41971695 0.55681818 0.32835821 0.21111111]
```

```
 [0.77229794 0.8277998  0.72431466 0.26151559 0.48532023 0.94952502
  0.57510409 0.50774668 0.45432876 0.52041614 0.36409634 0.33144312
  0.38962122 0.54545455 0.32835821 0.2       ]
 [0.77507301 0.8210739  0.72361425 0.2564402  0.48029972 0.95200337
  0.56879251 0.54517366 0.43386187 0.49838832 0.34748109 0.30469161
  0.37774508 0.55681818 0.31343284 0.16666667]
 [0.75896306 0.87311456 0.71048887 0.29841351 0.50852344 0.89520118
  0.60060185 0.45335294 0.52152364 0.60932208 0.34780358 0.32928215
  0.51860021 0.60227273 0.31343284 0.44444444]
 [0.77849867 0.8140264  0.72164313 0.25127202 0.47430632 0.95173936
  0.56176753 0.58134422 0.43582808 0.46726785 0.34485425 0.28221216
  0.38513289 0.54545455 0.34328358 0.16666667]
 [0.78354879 0.80570054 0.71199916 0.24246311 0.46241815 0.94314902
  0.55223884 0.61040936 0.50064331 0.40504943 0.4358256  0.30204612
  0.413115   0.54545455 0.34328358 0.16666667]
 [0.78143098 0.80802796 0.70680289 0.23957501 0.46016135 0.94117193
  0.55242352 0.59733349 0.53419479 0.37634693 0.51863774 0.36333175
  0.40452041 0.54545455 0.37313433 0.15555556]
 [0.77411003 0.81542342 0.70483083 0.23818944 0.46318863 0.94571634
  0.55653497 0.57285483 0.54618448 0.34700533 0.59238259 0.45443706
  0.37409619 0.57954545 0.40298507 0.16666667]]
Example sequence_y: 0
Number of sequences in x_train_ID: 569
Number of sequences in y_train_ID: 569
Number of sequences in x_test_ID: 181
Number of sequences in y_test_ID: 181
```

```python
import pandas as pd

# Function to convert sequences to DataFrame
def sequences_to_dataframe(x_seqs, y_seqs):
    # Create a list to hold dictionaries representing each sequence
    data = []

    # Iterate through each sequence
```

```
    for i in range(len(x_seqs)):
        sequence_dict = {'Sequence': x_seqs[i], 'Label': y_seqs[i]}
        data.append(sequence_dict)

    # Create a DataFrame from the list of dictionaries
    df = pd.DataFrame(data)

    return df

# Convert training sequences to DataFrame
df_train = sequences_to_dataframe(x_train_ID, y_train_ID)

# Convert testing sequences to DataFrame
df_test = sequences_to_dataframe(x_test_ID, y_test_ID)

# Display the DataFrames
print("Training Data:")
print(df_train.head())

print("\nTesting Data:")
print(df_test.head())
```

```
Training Data:
                                            Sequence  Label
0  [[0.6962190141248129, 0.8706620057654118, 0.64...      0
1  [[0.639922935787536, 0.9160756914667891, 0.430...      0
2  [[0.6199807719716374, 0.8535009287781409, 0.42...      0
3  [[0.6368026338400108, 0.8989425711944159, 0.46...      0
4  [[0.6186779666022791, 0.93403101127627, 0.4525...      0

Testing Data:
                                            Sequence  Label
0  [[0.999996812334544, 0.5199884756017968, 0.497...      0
1  [[0.7616934811690044, 0.8265418089535981, 0.70...      0
2  [[0.7504225260323647, 0.847607005532843, 0.721...      0
```

```
3  [[0.7646550277745912, 0.8309432398146688, 0.76...      0
4  [[0.7815466225016134, 0.8159679242940718, 0.73...      0
```

```
print(x_test_ID.shape)
print(x_train_ID.shape)
print(y_test_ID.shape)
print(y_train_ID.shape)
```

```
(181, 50, 16)
(569, 50, 16)
(181,)
(569,)
```

Now the data is structured in sequences of 50 datapoints, with uniform label values for each sequence.

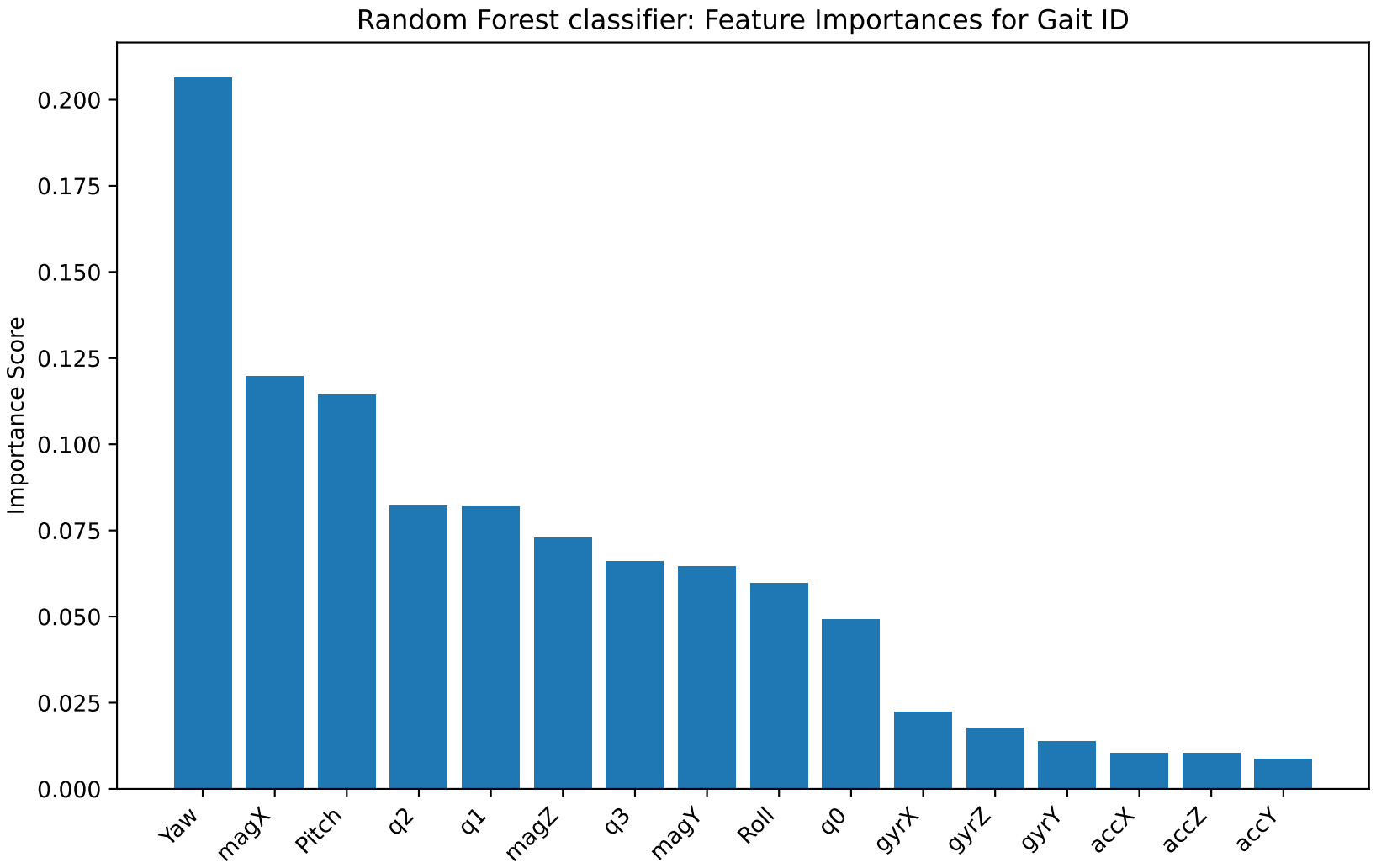
```
unique_labels_train = np.unique(y_train_ID)
unique_labels_test = np.unique(y_test_ID)

print("Unique labels in training set ID:", unique_labels_train)
print("Unique labels in testing set ID:", unique_labels_test)
```

```
Unique labels in training set ID: [ 0  1  2  3  4  5  6  7  8  9 10 11 12]
Unique labels in testing set ID: [ 0  1  2  3  4  5  6  7  8  9 10 11 12]
```

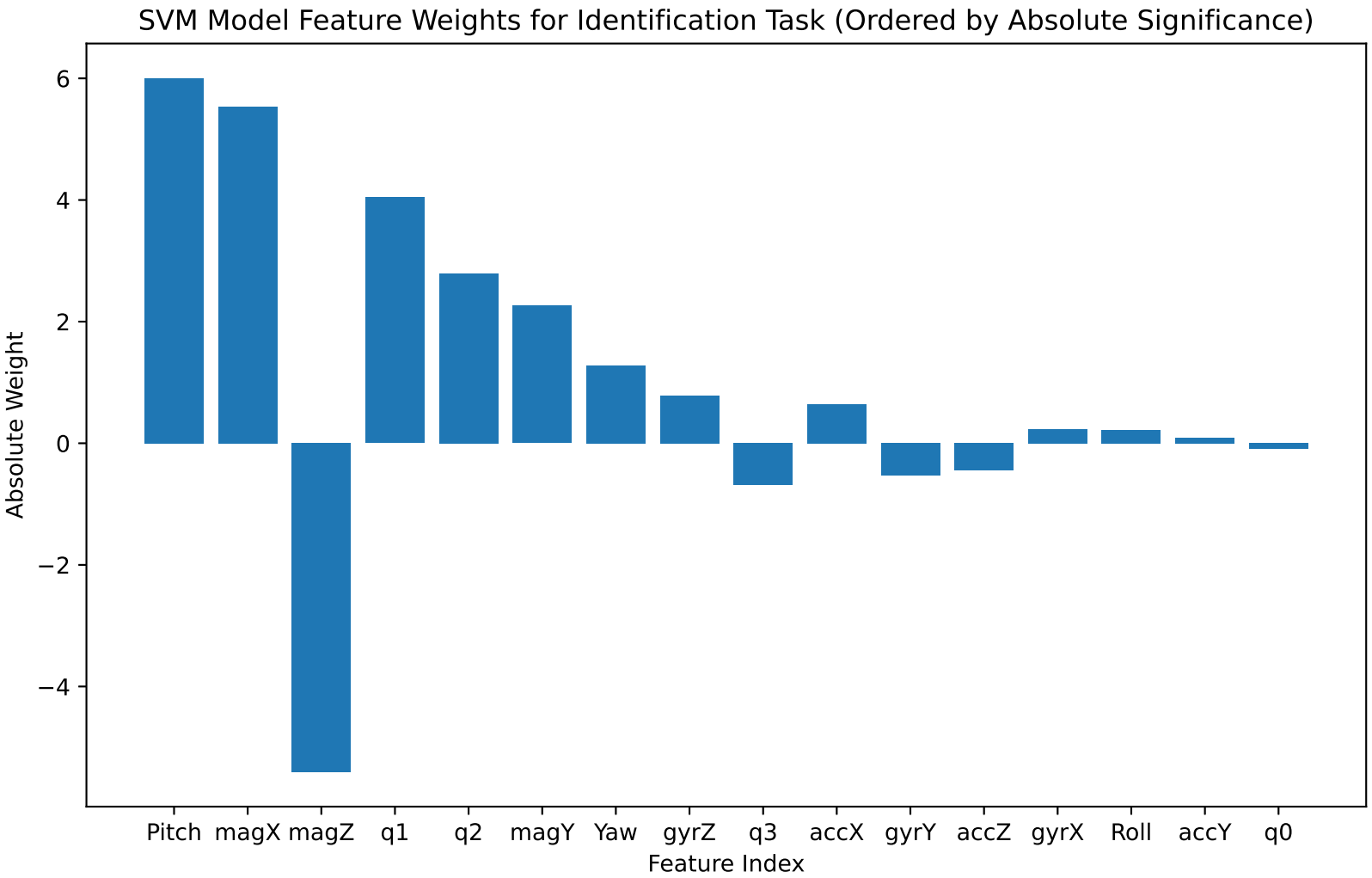
Prep the SVG feature importance analysis, execute random forest and support vector machine analysis for Identification task

```
import matplotlib.pyplot as plt

from sklearn import svm #SVM

from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt

# Feats_ID
Feats_ID = Feats_ID

# Fit the random forest classifier
model = RandomForestClassifier(n_estimators=1000, random_state=42)
```

```
model.fit(x_train_ID_Unseq, y_train_ID_Unseq)

# Get feature importances
feature_importances = model.feature_importances_

# Get the indices of features sorted by importance
indices_rf_identification = sorted(range(len(feature_importances)), key=lambda i: feature_importances[i], reverse=True)

# Plot the feature importances with flipped axis
plt.figure(figsize=(10, 6))
plt.bar(range(len(indices_rf_identification)), feature_importances[indices_rf_identification], align="center")
plt.xticks(range(len(indices_rf_identification)), [Feats_ID[i] for i in indices_rf_identification], rotation=45, ha='right')
plt.ylabel("Importance Score")
plt.title("Random Forest classifier: Feature Importances for Gait ID")

# Save the figure as EPS
plt.savefig('Identification_Random_Forest_feature_importances.eps', format='eps', bbox_inches='tight')

plt.show()
```


Random Forest classifier: Feature Importances for Gait ID
Importance Score
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Yaw
magX
Pitch
q2
q1
magZ
q3
magY
Roll
q0
gyrX
gyrZ
gyrY
accX
accZ
accY

```
    from sklearn import svm
import matplotlib.pyplot as plt

# Feats_ID
Feats_ID = Feats_ID


# Fit the SVM model for identification task
svm_model_Feature_Analysis_Identification = svm.SVC(kernel='linear')
svm_model_Feature_Analysis_Identification.fit(x_train_ID_Unseq, y_train_ID_Unseq)
feature_weights_Identification = svm_model_Feature_Analysis_Identification.coef_

# Get the indices of features sorted by absolute weight
indices_svm_identification = sorted(range(len(feature_weights_Identification[0])), key=lambda i: ...
    abs(feature_weights_Identification[0][i]), reverse=True)

# Plot the feature weights with flipped axis
plt.figure(figsize=(10, 6))
plt.bar(range(len(indices_svm_identification)), [feature_weights_Identification[0][i] for i in ...
    indices_svm_identification], align="center")
plt.xticks(range(len(indices_svm_identification)), [Feats_ID[i] for i in indices_svm_identification])
plt.xlabel('Feature Index')
plt.ylabel('Absolute Weight')
plt.title('SVM Model Feature Weights for Identification Task (Ordered by Absolute Significance)')

plt.savefig('Identification_SVM_feature_importances.eps', format='eps', bbox_inches='tight')

plt.show()
```




```
# Top 5 features from Random Forest for Identification task
top_rf_identification = [Feats_ID[i] for i in indices_rf_identification[:5]]
```

```python
# Top 5 features from SVM for Identification task
top_svm_identification = [Feats_ID[i] for i in indices_svm_identification[:5]]

# Combine top features for Identification task
combined_top_identification = set(top_rf_identification) | set(top_svm_identification)

# Display top features for Identification task
print("RF Top Features for Identification Task:", top_rf_identification)
print("SVM Top Features for Identification Task:", top_svm_identification)
print("Combined Top Features for Identification Task:", combined_top_identification)
print("\n")  # Add a newline for better separation in output
```

```
RF Top Features for Identification Task: ['Yaw', 'magX', 'Pitch', 'q2', 'q1']
SVM Top Features for Identification Task: ['Pitch', 'magX', 'magZ', 'q1', 'q2']
Combined Top Features for Identification Task: {'magX', 'Yaw', 'q1', 'Pitch', 'q2', 'magZ'}
```

D.2 UIA IMU: Temporal Convolutional Network for subject identification

```python
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_ID_Walking_Gait_Dataset_8_1_2024
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_IMU_9ax_WG_Dataset_W_Calibration_Data_U_21_Des_23.csv
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_IMU_9ax_WG_Dataset_U_19_Des_23
```

```python
df = pd.read_csv('/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_ID_Walking_Gait_Dataset_8_1_2024')

df
```

```
       Unnamed: 0   Timestamp        dt  Subject_no        q0        q1  \
0               0    0.034056  0.034056           0  0.999981 -0.005295
1               1    0.068112  0.034056           0  0.999967 -0.007995
2               2    0.102168  0.034056           0  0.999951 -0.009922
3               3    0.142416  0.040248           0  0.999923 -0.012365
4               4    0.182664  0.040248           0  0.999878 -0.015495
...           ...         ...       ...         ...       ...       ...
38754       38754  132.178021  0.039903          12 -0.978575 -0.198041
38755       38755  132.217924  0.039903          12 -0.980173 -0.194980
38756       38756  132.257827  0.039903          12 -0.982184 -0.185914
38757       38757  132.297729  0.039902          12 -0.984022 -0.175340
38758       38758  132.337632  0.039903          12 -0.985002 -0.166676

             q2        q3     Pitch      Roll         Yaw      accX      accY  \
0     -0.003050 -0.000229 -0.349617 -0.024413   -0.606688  0.001269 -0.007415
```

```
1     -0.001483 -0.000205 -0.170067 -0.022105   -0.916112 -0.014983 -0.010159
2     -0.000083 -0.000221 -0.009767 -0.025253   -1.136978 -0.024849 -0.013850
3      0.000831 -0.000712  0.094222 -0.082716   -1.417035 -0.025351 -0.018286
4      0.000539 -0.001899  0.058380 -0.218530   -1.775726 -0.018526 -0.021769
...         ...       ...       ...       ...         ...       ...       ...
38754 -0.020191  0.052557  3.458979 -5.437485 -337.283108  0.181484  0.031365
38755 -0.004102  0.035017  1.243237 -3.843961 -337.540503  0.114871 -0.046202
38756  0.004001  0.027092  0.126920 -3.136616 -338.566529  0.002566 -0.098474
38757  0.004847  0.030541  0.067100 -3.544386 -339.795425 -0.110981 -0.107848
38758  0.002149  0.044560  0.608490 -5.078503 -340.818421 -0.178788 -0.079801

           accZ      gyrX      gyrY      gyrZ  magX  magY  magZ
0      0.007225 -0.015511 -0.128565 -0.013460    72   -42     7
1      0.003744  0.009013 -0.156497 -0.001248    73   -43     7
2     -0.001352  0.032911 -0.179508 -0.004759    72   -43     8
3     -0.007889  0.054869 -0.196698 -0.028749    72   -43     7
4     -0.014033  0.074107 -0.206459 -0.063698    72   -43     5
...         ...       ...       ...       ...   ...   ...   ...
38754  0.091784  0.130059 -0.464335  1.226585    56   -27    41
38755  0.108562  0.023426 -0.404651  0.938618    55   -27    40
38756  0.088519 -0.176852 -0.326740  0.471631    55   -28    41
38757  0.041200 -0.331125 -0.271021 -0.084528    56   -30    39
38758 -0.012090 -0.334927 -0.259580 -0.616271    56   -31    36

[38759 rows x 20 columns]
```

From feature selection analysis using svm and random forest, top features:

#Feats_ID = ['Pitch', 'q1', 'Yaw', 'MotionDeg', 'q2', 'q3']

```
Feats = ['Pitch', 'Yaw', 'q1', 'magX', 'q2', 'magZ']

df['Gait_Identification'] = df['Subject_no']

Label = ['Gait_Identification']

Columns = ['Gait_Identification', 'Pitch', 'Yaw', 'q1', 'magX', 'q2', 'magZ']
```

```
df = df[Columns]

df
```

```
       Gait_Identification     Pitch         Yaw        q1  magX        q2  \
0                        0 -0.349617   -0.606688 -0.005295    72 -0.003050
1                        0 -0.170067   -0.916112 -0.007995    73 -0.001483
2                        0 -0.009767   -1.136978 -0.009922    72 -0.000083
3                        0  0.094222   -1.417035 -0.012365    72  0.000831
4                        0  0.058380   -1.775726 -0.015495    72  0.000539
...                    ...       ...         ...       ...   ...       ...
38754                   12  3.458979 -337.283108 -0.198041    56 -0.020191
38755                   12  1.243237 -337.540503 -0.194980    55 -0.004102
38756                   12  0.126920 -338.566529 -0.185914    55  0.004001
38757                   12  0.067100 -339.795425 -0.175340    56  0.004847
38758                   12  0.608490 -340.818421 -0.166676    56  0.002149

       magZ
0         7
1         7
2         8
3         7
4         5
...     ...
38754    41
38755    40
38756    41
38757    39
38758    36

[38759 rows x 7 columns]
```

```
df_ID = df.copy()
```

```
import pandas as pd
import numpy as np
```

```
from sklearn.preprocessing import StandardScaler


sequence_length = 30
overlap_percentage = 0.75


X_columns = Feats
y_column = Label

# Extrapolate features
X = df[X_columns].values


# Scale
scaler = StandardScaler()
df[X_columns] = scaler.fit_transform(X)


# Calculate the overlap and step size
overlap_size = int(sequence_length * overlap_percentage)
step_size = sequence_length - overlap_size

# Initialize lists to store sequences and labels
sequences = []
labels = []

# Iterate through the dataframe to create sequences
for i in range(0, len(df) - sequence_length + 1, step_size):
    sequence = df[X_columns].values[i:i + sequence_length]
    label_values = tuple(tuple(row) for row in df[y_column].values[i:i + sequence_length])  # Convert nested arrays to tuples

    '''
```

```
    Only include sequences with uniform label value, i.e.
    drop data in transition from one subject to another.
    Several users for same prediction is not a real-world
    scenario.

    '''

    if len(set(label_values)) == 1:
        label = label_values[0]
        sequences.append(sequence)
        labels.append(label)

# Convert lists to numpy arrays
X = np.array(sequences)
y = np.array(labels)

print(X.shape)

print(y.shape)
```

```
(4800, 30, 6)
(4800, 1)
```

```
np.unique(y)
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

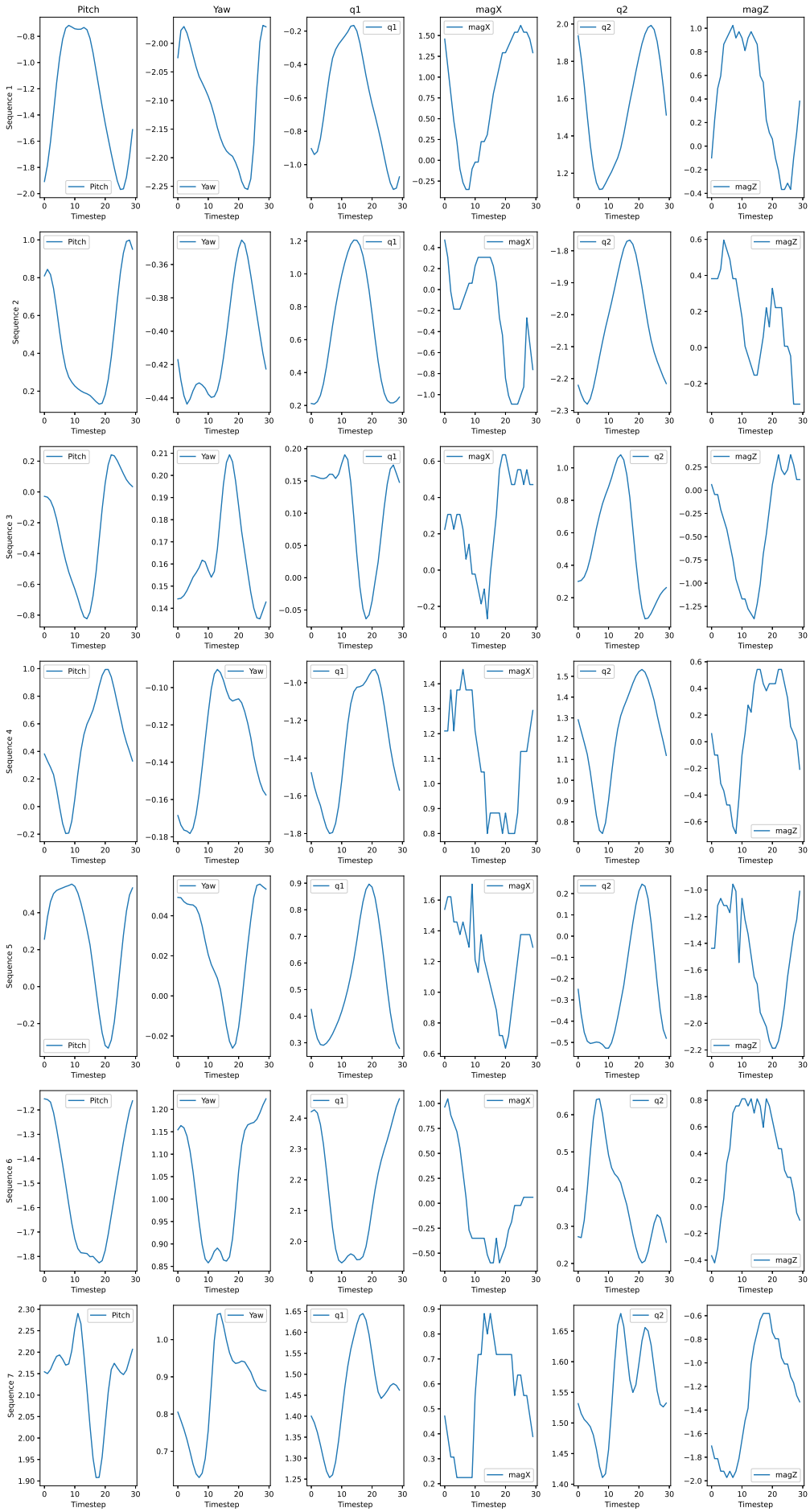
```
from sklearn.model_selection import train_test_split

# Flatten y for stratification
y_flat = y.ravel()

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y_flat, random_state=42)

# Reshape labels so that they are 1 dinemnisonal
```

```
y_train = y_train.ravel()
y_test = y_test.ravel()

# Print the shapes of the resulting sets
print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
print("X_test shape:", X_test.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (3840, 30, 6)
y_train shape: (3840,)
X_test shape: (960, 30, 6)
y_test shape: (960,)
```

```
import matplotlib.pyplot as plt
import numpy as np

def plot_sequences(X, y, num_sequences=10):
    """
    Plot a random selection of sequences with individual spines for each feature.

    Parameters:
    - X: Feature sequences
    - y: Labels
    - num_sequences: Number of sequences to plot
    """
    num_sequences_to_plot = min(num_sequences, len(X))
    num_features = X.shape[2]
    figsize_height = 4 * num_sequences_to_plot  # Adjust the height based on your preference
    plt.figure(figsize=(15, figsize_height))
    indices = np.random.choice(len(X), num_sequences_to_plot, replace=False)

    for i, idx in enumerate(indices, 1):
        for j in range(num_features):
            plt.subplot(num_sequences_to_plot, num_features, (i - 1) * num_features + j + 1)
```

```
            plt.plot(X[idx, :, j], label=f"{Feats[j]}") #Extrapolate name from list of feats
            plt.xlabel("Timestep")

            if i == 1:
                plt.title(f"{Feats[j]}")
                plt.xlabel("Timestep")


            if j == 0:
                plt.ylabel(f"Sequence {i}")
                plt.xlabel("Timestep")

            plt.legend()

    plt.tight_layout()
    plt.savefig('Sequence_Samples_UIA_IMU.eps', format='eps')
    plt.savefig('Sequence_Samples_UIA_IMU.jpeg', format='jpeg')
    plt.show()


# Plot 7 random sequences from the training set
plot_sequences(X_train, y_train)


#Remember: Feats = ['Pitch' (1), 'Yaw'(2), 'q1'(3), 'magX'(4), 'q2'(5), 'magZ'(6)]
```


Pitch
Yaw
q1
magX
q2
magZ
Sequence 1
Sequence 2
Sequence 3
Sequence 4
Sequence 5
Sequence 6
Sequence 7
Timestep

```
from keras.layers import Multiply
from keras.models import Sequential
from keras.layers import Conv1D, Dense, Activation, Input, Reshape, Lambda, concatenate, Layer
from keras.layers import Add, Dropout, BatchNormalization, GlobalAveragePooling1D
from keras.optimizers import Adam

class ResidualBlock(Layer):
    def __init__(self, dilation_rate, nb_filters, kernel_size, padding, dropout_rate, **kwargs):
        super(ResidualBlock, self).__init__(**kwargs)
        self.dilation_rate = dilation_rate
        self.nb_filters = nb_filters
        self.kernel_size = kernel_size
        self.padding = padding
        self.dropout_rate = dropout_rate

    def build(self, input_shape):
        super(ResidualBlock, self).build(input_shape)
        self.tanh_conv = Conv1D(filters=self.nb_filters, kernel_size=self.kernel_size, dilation_rate=self.dilation_rate,
                                padding=self.padding, activation='tanh')
        self.sigm_conv = Conv1D(filters=self.nb_filters, kernel_size=self.kernel_size, dilation_rate=self.dilation_rate,
                                padding=self.padding, activation='sigmoid')
        self.multiply = Multiply()
        self.one_by_one_conv = Conv1D(filters=self.nb_filters, kernel_size=1, padding='same')
        self.add_layer = Add()
        self.activation = Activation('relu')

    def call(self, x):
        prev_x = x
        tanh_out = self.tanh_conv(x)
        sigm_out = self.sigm_conv(x)
        x = self.multiply([tanh_out, sigm_out])
        x = self.one_by_one_conv(x)
        res_x = self.add_layer([prev_x, x])
```

```
        activation = self.activation(res_x)
        return activation

def build_tcn_model(input_shape, nb_filters, kernel_size, dilations, nb_stacks, dropout_rate=0.0, output_dim=30):
    model = Sequential()
    model.add(Conv1D(filters=nb_filters, kernel_size=1, padding='same', input_shape=input_shape))
    for _ in range(nb_stacks):
        for dilation_rate in dilations:
            model.add(ResidualBlock(dilation_rate=dilation_rate,
                                    nb_filters=nb_filters, kernel_size=kernel_size,
                                    padding='causal', dropout_rate=dropout_rate))
    model.add(GlobalAveragePooling1D())
    model.add(Dense(units=output_dim, activation='softmax'))
    model.compile(optimizer=Adam(lr=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model

# Example usage:
input_dim = X.shape[2]
sequence_length = X.shape[1]
input_shape = (sequence_length, input_dim)
output_dim = len(np.unique(y))
nb_filters = 64
kernel_size = 3
dilations = [1, 2, 3, 6]
nb_stacks = 3
dropout_rate = 0.2

model = build_tcn_model(input_shape, nb_filters, kernel_size, dilations, nb_stacks, dropout_rate, output_dim)
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_1 (Conv1D)           (None, 30, 64)            448
```

```
residual_block_12 (Residua  (None, 30, 64)           28864
lBlock)

residual_block_13 (Residua  (None, 30, 64)           28864
lBlock)

residual_block_14 (Residua  (None, 30, 64)           28864
lBlock)

residual_block_15 (Residua  (None, 30, 64)           28864
lBlock)

residual_block_16 (Residua  (None, 30, 64)           28864
lBlock)

residual_block_17 (Residua  (None, 30, 64)           28864
lBlock)

residual_block_18 (Residua  (None, 30, 64)           28864
lBlock)

residual_block_19 (Residua  (None, 30, 64)           28864
lBlock)

residual_block_20 (Residua  (None, 30, 64)           28864
lBlock)

residual_block_21 (Residua  (None, 30, 64)           28864
lBlock)

residual_block_22 (Residua  (None, 30, 64)           28864
lBlock)
```

```
 residual_block_23 (Residua  (None, 30, 64)            28864
 lBlock)

 global_average_pooling1d_1  (None, 64)                0
  (GlobalAveragePooling1D)

 dense_1 (Dense)             (None, 13)                845

=================================================================
Total params: 347661 (1.33 MB)
Trainable params: 347661 (1.33 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```python
# Fit model to data (training)
history = model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test, y_test))
```



```python
import matplotlib.pyplot as plt
import numpy as np


# Plotting the learning curve
plt.figure(figsize=(10, 6))
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Temporal Convolutional Network Learning Curve')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.savefig('UIA_IMU_Temporal_Convolutional_Network_Gait_ID_8_1_2024_trials.eps', format='eps')
plt.savefig('UIA_IMU_Temporal_Convolutional_Network_Gait_ID_8_1_2024_trials.jpeg', format='jpeg')
plt.show()

# Save the model summary to a text file
```

```
with open('summary_UIA_IMU_temporal_convolutional_network_gait_id_model_8_1_2024.txt', 'w') as f:
    model.summary(print_fn=lambda x: f.write(x + '\n'))

# Save the entire model to a file
model.save('temporal_convolutional_network_gait_id_model')
```

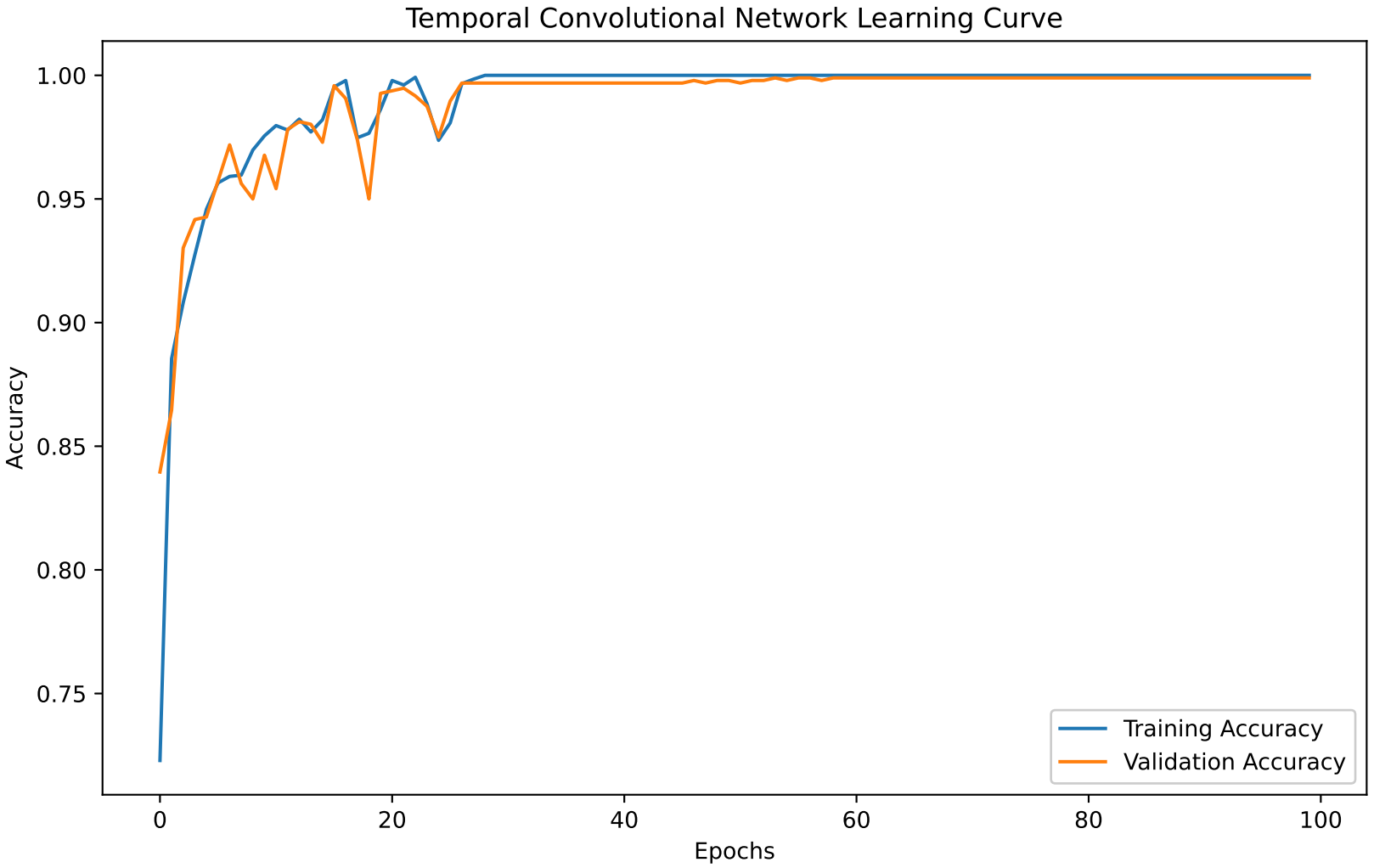
```
Epoch 1/100
120/120 [==============================] - 16s 24ms/step - loss: 0.8250 - accuracy: 0.7091 - val_loss: 0.4943 - ...
   val_accuracy: 0.8271
Epoch 2/100
120/120 [==============================] - 2s 17ms/step - loss: 0.3497 - accuracy: 0.8716 - val_loss: 0.2496 - ...
   val_accuracy: 0.9208
Epoch 3/100
120/120 [==============================] - 2s 17ms/step - loss: 0.2465 - accuracy: 0.9128 - val_loss: 0.2512 - ...
   val_accuracy: 0.8990
Epoch 4/100
120/120 [==============================] - 2s 17ms/step - loss: 0.1752 - accuracy: 0.9409 - val_loss: 0.1759 - ...
   val_accuracy: 0.9271
Epoch 5/100
120/120 [==============================] - 2s 17ms/step - loss: 0.1483 - accuracy: 0.9508 - val_loss: 0.1836 - ...
   val_accuracy: 0.9417
Epoch 6/100
120/120 [==============================] - 2s 17ms/step - loss: 0.1441 - accuracy: 0.9474 - val_loss: 0.0942 - ...
   val_accuracy: 0.9750
Epoch 7/100
120/120 [==============================] - 2s 18ms/step - loss: 0.0800 - accuracy: 0.9742 - val_loss: 0.0526 - ...
   val_accuracy: 0.9844
Epoch 8/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0514 - accuracy: 0.9826 - val_loss: 0.1341 - ...
   val_accuracy: 0.9646
Epoch 9/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0918 - accuracy: 0.9643 - val_loss: 0.0769 - ...
   val_accuracy: 0.9688
Epoch 10/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0710 - accuracy: 0.9758 - val_loss: 0.1697 - ...
   val_accuracy: 0.9615
```

```
Epoch 11/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0469 - accuracy: 0.9849 - val_loss: 0.0576 - ...
    val_accuracy: 0.9802
Epoch 12/100
120/120 [==============================] - 2s 17ms/step - loss: 0.1010 - accuracy: 0.9693 - val_loss: 0.1064 - ...
    val_accuracy: 0.9677
Epoch 13/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0789 - accuracy: 0.9742 - val_loss: 0.0843 - ...
    val_accuracy: 0.9719
Epoch 14/100
120/120 [==============================] - 2s 19ms/step - loss: 0.0234 - accuracy: 0.9943 - val_loss: 0.0244 - ...
    val_accuracy: 0.9927
Epoch 15/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0509 - accuracy: 0.9852 - val_loss: 0.1502 - ...
    val_accuracy: 0.9604
Epoch 16/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0561 - accuracy: 0.9815 - val_loss: 0.0644 - ...
    val_accuracy: 0.9781
Epoch 17/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0422 - accuracy: 0.9865 - val_loss: 0.0923 - ...
    val_accuracy: 0.9677
Epoch 18/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0367 - accuracy: 0.9880 - val_loss: 0.0444 - ...
    val_accuracy: 0.9812
Epoch 19/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0126 - accuracy: 0.9966 - val_loss: 0.0192 - ...
    val_accuracy: 0.9927
Epoch 20/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0111 - accuracy: 0.9969 - val_loss: 0.0295 - ...
    val_accuracy: 0.9896
Epoch 21/100
120/120 [==============================] - 2s 18ms/step - loss: 0.0861 - accuracy: 0.9755 - val_loss: 0.1957 - ...
    val_accuracy: 0.9240
Epoch 22/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0541 - accuracy: 0.9823 - val_loss: 0.0291 - ...
    val_accuracy: 0.9906
Epoch 23/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0159 - accuracy: 0.9953 - val_loss: 0.0416 - ...
    val_accuracy: 0.9854
```

```
Epoch 24/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0262 - accuracy: 0.9924 - val_loss: 0.0686 - ...
    val_accuracy: 0.9771
Epoch 25/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0291 - accuracy: 0.9917 - val_loss: 0.0143 - ...
    val_accuracy: 0.9937
Epoch 26/100
120/120 [==============================] - 2s 18ms/step - loss: 0.0062 - accuracy: 0.9979 - val_loss: 0.0074 - ...
    val_accuracy: 0.9979
Epoch 27/100
120/120 [==============================] - 2s 18ms/step - loss: 0.0056 - accuracy: 0.9990 - val_loss: 0.0148 - ...
    val_accuracy: 0.9958
Epoch 28/100
120/120 [==============================] - 2s 18ms/step - loss: 0.0083 - accuracy: 0.9977 - val_loss: 0.0215 - ...
    val_accuracy: 0.9937
Epoch 29/100
120/120 [==============================] - 2s 19ms/step - loss: 0.0132 - accuracy: 0.9964 - val_loss: 0.0677 - ...
    val_accuracy: 0.9844
Epoch 30/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0739 - accuracy: 0.9784 - val_loss: 0.1358 - ...
    val_accuracy: 0.9521
Epoch 31/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0669 - accuracy: 0.9812 - val_loss: 0.0849 - ...
    val_accuracy: 0.9760
Epoch 32/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0369 - accuracy: 0.9896 - val_loss: 0.0126 - ...
    val_accuracy: 0.9948
Epoch 33/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0126 - accuracy: 0.9964 - val_loss: 0.0384 - ...
    val_accuracy: 0.9948
Epoch 34/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0049 - accuracy: 0.9992 - val_loss: 0.0069 - ...
    val_accuracy: 0.9979
Epoch 35/100
120/120 [==============================] - 2s 17ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0039 - ...
    val_accuracy: 0.9990
Epoch 36/100
120/120 [==============================] - 2s 18ms/step - loss: 7.5755e-04 - accuracy: 1.0000 - val_loss: 0.0040 - ...
    val_accuracy: 0.9990
```

```
Epoch 37/100
120/120 [==============================] - 2s 17ms/step - loss: 5.4096e-04 - accuracy: 1.0000 - val_loss: 0.0034 - ...
   val_accuracy: 0.9990
Epoch 38/100
120/120 [==============================] - 2s 16ms/step - loss: 4.3130e-04 - accuracy: 1.0000 - val_loss: 0.0034 - ...
   val_accuracy: 1.0000
Epoch 39/100
120/120 [==============================] - 2s 16ms/step - loss: 3.5903e-04 - accuracy: 1.0000 - val_loss: 0.0032 - ...
   val_accuracy: 1.0000
Epoch 40/100
120/120 [==============================] - 2s 16ms/step - loss: 2.9934e-04 - accuracy: 1.0000 - val_loss: 0.0030 - ...
   val_accuracy: 1.0000
Epoch 41/100
120/120 [==============================] - 2s 17ms/step - loss: 2.6073e-04 - accuracy: 1.0000 - val_loss: 0.0027 - ...
   val_accuracy: 1.0000
Epoch 42/100
120/120 [==============================] - 2s 17ms/step - loss: 2.2533e-04 - accuracy: 1.0000 - val_loss: 0.0026 - ...
   val_accuracy: 1.0000
Epoch 43/100
120/120 [==============================] - 2s 17ms/step - loss: 1.9842e-04 - accuracy: 1.0000 - val_loss: 0.0024 - ...
   val_accuracy: 1.0000
Epoch 44/100
120/120 [==============================] - 2s 18ms/step - loss: 1.7648e-04 - accuracy: 1.0000 - val_loss: 0.0023 - ...
   val_accuracy: 1.0000
Epoch 45/100
120/120 [==============================] - 2s 17ms/step - loss: 1.5710e-04 - accuracy: 1.0000 - val_loss: 0.0021 - ...
   val_accuracy: 1.0000
Epoch 46/100
120/120 [==============================] - 2s 17ms/step - loss: 1.4095e-04 - accuracy: 1.0000 - val_loss: 0.0023 - ...
   val_accuracy: 1.0000
Epoch 47/100
120/120 [==============================] - 2s 17ms/step - loss: 1.2598e-04 - accuracy: 1.0000 - val_loss: 0.0022 - ...
   val_accuracy: 1.0000
Epoch 48/100
120/120 [==============================] - 2s 18ms/step - loss: 1.1282e-04 - accuracy: 1.0000 - val_loss: 0.0020 - ...
   val_accuracy: 1.0000
Epoch 49/100
120/120 [==============================] - 2s 17ms/step - loss: 1.0234e-04 - accuracy: 1.0000 - val_loss: 0.0019 - ...
   val_accuracy: 1.0000
```

```
Epoch 50/100

.............. Skip to end


120/120 [==============================] - 2s 17ms/step - loss: 3.0636e-06 - accuracy: 1.0000 - val_loss: 0.0012 - ...
   val_accuracy: 1.0000
Epoch 96/100
120/120 [==============================] - 2s 17ms/step - loss: 2.8743e-06 - accuracy: 1.0000 - val_loss: 0.0013 - ...
   val_accuracy: 1.0000
Epoch 97/100
120/120 [==============================] - 2s 17ms/step - loss: 2.6766e-06 - accuracy: 1.0000 - val_loss: 0.0013 - ...
   val_accuracy: 1.0000
Epoch 98/100
120/120 [==============================] - 2s 16ms/step - loss: 2.4995e-06 - accuracy: 1.0000 - val_loss: 0.0013 - ...
   val_accuracy: 1.0000
Epoch 99/100
120/120 [==============================] - 2s 16ms/step - loss: 2.3310e-06 - accuracy: 1.0000 - val_loss: 0.0014 - ...
   val_accuracy: 1.0000
Epoch 100/100
120/120 [==============================] - 2s 17ms/step - loss: 2.1796e-06 - accuracy: 1.0000 - val_loss: 0.0012 - ...
   val_accuracy: 1.0000
```




```
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(X_test, y_test)
```

```
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')
```

```
30/30 [==============================] - 0s 5ms/step - loss: 0.0012 - accuracy: 1.0000
Test Loss: 0.0012283691903576255, Test Accuracy: 1.0
```

UIA IMU: TCN model simplification and analysis. Comparing performance against other ML model architectures

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_ID_Walking_Gait_Dataset_8_1_2024
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_IMU_9ax_WG_Dataset_W_Calibration_Data_U_21_Des_23.csv
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_IMU_9ax_WG_Dataset_U_19_Des_23
```

```
df = pd.read_csv('/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_ID_Walking_Gait_Dataset_8_1_2024')
```

```
df
```

```
       Unnamed: 0   Timestamp        dt  Subject_no        q0        q1  \
0               0    0.034056  0.034056           0  0.999981 -0.005295
1               1    0.068112  0.034056           0  0.999967 -0.007995
2               2    0.102168  0.034056           0  0.999951 -0.009922
3               3    0.142416  0.040248           0  0.999923 -0.012365
4               4    0.182664  0.040248           0  0.999878 -0.015495
...           ...         ...       ...         ...       ...       ...
38754       38754  132.178021  0.039903          12 -0.978575 -0.198041
38755       38755  132.217924  0.039903          12 -0.980173 -0.194980
38756       38756  132.257827  0.039903          12 -0.982184 -0.185914
38757       38757  132.297729  0.039902          12 -0.984022 -0.175340
38758       38758  132.337632  0.039903          12 -0.985002 -0.166676
```

```
             q2        q3     Pitch      Roll         Yaw      accX      accY  \
0     -0.003050 -0.000229 -0.349617 -0.024413   -0.606688  0.001269 -0.007415
1     -0.001483 -0.000205 -0.170067 -0.022105   -0.916112 -0.014983 -0.010159
2     -0.000083 -0.000221 -0.009767 -0.025253   -1.136978 -0.024849 -0.013850
3      0.000831 -0.000712  0.094222 -0.082716   -1.417035 -0.025351 -0.018286
4      0.000539 -0.001899  0.058380 -0.218530   -1.775726 -0.018526 -0.021769
...         ...       ...       ...       ...         ...       ...       ...
38754 -0.020191  0.052557  3.458979 -5.437485 -337.283108  0.181484  0.031365
38755 -0.004102  0.035017  1.243237 -3.843961 -337.540503  0.114871 -0.046202
38756  0.004001  0.027092  0.126920 -3.136616 -338.566529  0.002566 -0.098474
38757  0.004847  0.030541  0.067100 -3.544386 -339.795425 -0.110981 -0.107848
38758  0.002149  0.044560  0.608490 -5.078503 -340.818421 -0.178788 -0.079801

           accZ      gyrX      gyrY      gyrZ  magX  magY  magZ
0      0.007225 -0.015511 -0.128565 -0.013460    72   -42     7
1      0.003744  0.009013 -0.156497 -0.001248    73   -43     7
2     -0.001352  0.032911 -0.179508 -0.004759    72   -43     8
3     -0.007889  0.054869 -0.196698 -0.028749    72   -43     7
4     -0.014033  0.074107 -0.206459 -0.063698    72   -43     5
...         ...       ...       ...       ...   ...   ...   ...
38754  0.091784  0.130059 -0.464335  1.226585    56   -27    41
38755  0.108562  0.023426 -0.404651  0.938618    55   -27    40
38756  0.088519 -0.176852 -0.326740  0.471631    55   -28    41
38757  0.041200 -0.331125 -0.271021 -0.084528    56   -30    39
38758 -0.012090 -0.334927 -0.259580 -0.616271    56   -31    36

[38759 rows x 20 columns]
```

From feature selection analysis using svm and random forest, top features:

#Feats_ID = ['Pitch', 'q1', 'Yaw', 'MotionDeg', 'q2', 'q3']

```
Feats = ['Pitch', 'Yaw', 'q1', 'magX', 'q2', 'magZ']

df['Gait_Identification'] = df['Subject_no']
```

```
Label = ['Gait_Identification']

Columns = ['Gait_Identification', 'Pitch', 'Yaw', 'q1', 'magX', 'q2', 'magZ']

df = df[Columns]

df
```

```
       Gait_Identification     Pitch         Yaw        q1  magX        q2  \
0                        0 -0.349617   -0.606688 -0.005295    72 -0.003050
1                        0 -0.170067   -0.916112 -0.007995    73 -0.001483
2                        0 -0.009767   -1.136978 -0.009922    72 -0.000083
3                        0  0.094222   -1.417035 -0.012365    72  0.000831
4                        0  0.058380   -1.775726 -0.015495    72  0.000539
...                    ...       ...         ...       ...   ...       ...
38754                   12  3.458979 -337.283108 -0.198041    56 -0.020191
38755                   12  1.243237 -337.540503 -0.194980    55 -0.004102
38756                   12  0.126920 -338.566529 -0.185914    55  0.004001
38757                   12  0.067100 -339.795425 -0.175340    56  0.004847
38758                   12  0.608490 -340.818421 -0.166676    56  0.002149

       magZ
0         7
1         7
2         8
3         7
4         5
...     ...
38754    41
38755    40
38756    41
38757    39
38758    36

[38759 rows x 7 columns]
```

```
df_ID = df.copy()
```

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler


sequence_length = 30
overlap_percentage = 0.75


X_columns = Feats
y_column = Label

# Extrapolate features
X = df[X_columns].values


# Scale
scaler = StandardScaler()
df[X_columns] = scaler.fit_transform(X)


# Calculate the overlap and step size
overlap_size = int(sequence_length * overlap_percentage)
step_size = sequence_length - overlap_size

# Initialize lists to store sequences and labels
sequences = []
labels = []

# Iterate through the dataframe to create sequences
for i in range(0, len(df) - sequence_length + 1, step_size):
    sequence = df[X_columns].values[i:i + sequence_length]
    label_values = tuple(tuple(row) for row in df[y_column].values[i:i + sequence_length])  # Convert nested arrays to tuples
```

```
    '''

    Only include sequences with uniform label value, i.e.
    drop data in transition from one subject to another.
    Several users for same prediction is not a real-world
    scenario.

    '''

    if len(set(label_values)) == 1:
        label = label_values[0]
        sequences.append(sequence)
        labels.append(label)

# Convert lists to numpy arrays
X = np.array(sequences)
y = np.array(labels)

print(X.shape)

print(y.shape)
```

```
(4800, 30, 6)
(4800, 1)
```

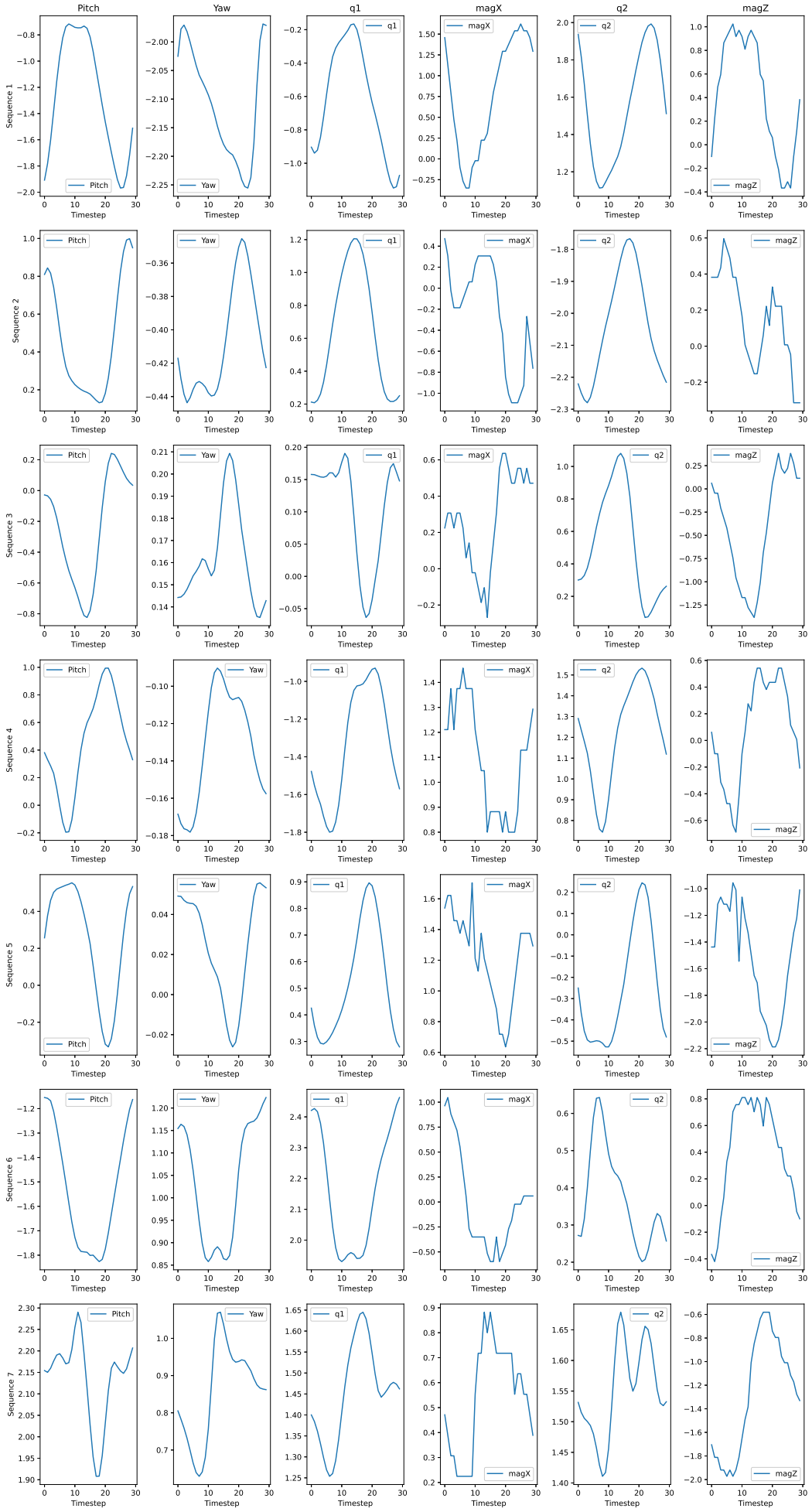
```
np.unique(y)
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

```
from sklearn.model_selection import train_test_split

# Flatten y for stratification
y_flat = y.ravel()

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y_flat, random_state=42)
```

```
# Reshape labels so that they are 1 dinemnisonal

y_train = y_train.ravel()
y_test = y_test.ravel()

# Print the shapes of the resulting sets
print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
print("X_test shape:", X_test.shape)
print("y_test shape:", y_test.shape)

X_train shape: (3840, 30, 6)
y_train shape: (3840,)
X_test shape: (960, 30, 6)
y_test shape: (960,)

import matplotlib.pyplot as plt
import numpy as np

def plot_sequences(X, y, num_sequences=7):
    """
    Plot a random selection of sequences with individual spines for each feature.

    Parameters:
    - X: Feature sequences
    - y: Labels
    - num_sequences: Number of sequences to plot
    """
    num_sequences_to_plot = min(num_sequences, len(X))
    num_features = X.shape[2]
    figsize_height = 4 * num_sequences_to_plot  # Adjust the height based on your preference
    plt.figure(figsize=(15, figsize_height))
    indices = np.random.choice(len(X), num_sequences_to_plot, replace=False)

    for i, idx in enumerate(indices, 1):
```

```
        for j in range(num_features):
            plt.subplot(num_sequences_to_plot, num_features, (i - 1) * num_features + j + 1)
            plt.plot(X[idx, :, j], label=f"{Feats[j]}") #Extrapolate name from list of feats
            plt.xlabel("Timestep")

            if i == 1:
                plt.title(f"{Feats[j]}")
                plt.xlabel("Timestep")


            if j == 0:
                plt.ylabel(f"Sequence {i}")
                plt.xlabel("Timestep")

            plt.legend()

    plt.tight_layout()
    plt.savefig('Sequence_Samples_UIA_IMU.eps', format='eps')
    plt.savefig('Sequence_Samples_UIA_IMU.jpeg', format='jpeg')
    plt.show()


# Plot some random sequences from the training set
plot_sequences(X_train, y_train)


#Remember: Feats = ['Pitch' (1), 'Yaw'(2), 'q1'(3), 'magX'(4), 'q2'(5), 'magZ'(6)]
```


Pitch
Yaw
q1
magX
q2
magZ
Sequence 1
Sequence 2
Sequence 3
Sequence 4
Sequence 5
Sequence 6
Sequence 7
Timestep

Simplified TCN model training and results

```
from keras.layers import Multiply
from keras.models import Sequential
from keras.layers import Conv1D, Dense, Activation, Input, Reshape, Lambda, concatenate, Layer
from keras.layers import Add, Dropout, BatchNormalization, GlobalAveragePooling1D
from keras.optimizers import Adam

class ResidualBlock(Layer):
    def __init__(self, dilation_rate, nb_filters, kernel_size, padding, dropout_rate, **kwargs):
        super(ResidualBlock, self).__init__(**kwargs)
        self.dilation_rate = dilation_rate
        self.nb_filters = nb_filters
        self.kernel_size = kernel_size
        self.padding = padding
        self.dropout_rate = dropout_rate

    def build(self, input_shape):
        super(ResidualBlock, self).build(input_shape)
        self.tanh_conv = Conv1D(filters=self.nb_filters, kernel_size=self.kernel_size, dilation_rate=self.dilation_rate,
                                padding=self.padding, activation='tanh')
        self.sigm_conv = Conv1D(filters=self.nb_filters, kernel_size=self.kernel_size, dilation_rate=self.dilation_rate,
                                padding=self.padding, activation='sigmoid')
        self.multiply = Multiply()
        self.one_by_one_conv = Conv1D(filters=self.nb_filters, kernel_size=1, padding='same')
        self.add_layer = Add()
        self.activation = Activation('relu')

    def call(self, x):
        prev_x = x
        tanh_out = self.tanh_conv(x)
        sigm_out = self.sigm_conv(x)
        x = self.multiply([tanh_out, sigm_out])
        x = self.one_by_one_conv(x)
```

```
        res_x = self.add_layer([prev_x, x])
        activation = self.activation(res_x)
        return activation

def build_tcn_model(input_shape, nb_filters, kernel_size, dilations, nb_stacks, dropout_rate=0.0, output_dim=30):
    model = Sequential()
    model.add(Conv1D(filters=nb_filters, kernel_size=1, padding='same', input_shape=input_shape))
    for _ in range(nb_stacks):
        for dilation_rate in dilations:
            model.add(ResidualBlock(dilation_rate=dilation_rate,
                                    nb_filters=nb_filters, kernel_size=kernel_size,
                                    padding='causal', dropout_rate=dropout_rate))
    model.add(GlobalAveragePooling1D())
    model.add(Dense(units=output_dim, activation='softmax'))
    model.compile(optimizer=Adam(lr=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model

# Example usage:
input_dim = X.shape[2]
sequence_length = X.shape[1]
input_shape = (sequence_length, input_dim)
output_dim = len(np.unique(y))
nb_filters = 36
kernel_size = 3
dilations = [1,2,4,5,6]
nb_stacks = 3
dropout_rate = 0.2

TCN_Model = build_tcn_model(input_shape, nb_filters, kernel_size, dilations, nb_stacks, dropout_rate, output_dim)
TCN_Model.summary()
```

```
Model: "sequential_27"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

```
conv1d_27 (Conv1D)          (None, 30, 36)            252

residual_block_210 (Residu  (None, 30, 36)            9180
alBlock)

residual_block_211 (Residu  (None, 30, 36)            9180
alBlock)

residual_block_212 (Residu  (None, 30, 36)            9180
alBlock)

residual_block_213 (Residu  (None, 30, 36)            9180
alBlock)

residual_block_214 (Residu  (None, 30, 36)            9180
alBlock)

residual_block_215 (Residu  (None, 30, 36)            9180
alBlock)

residual_block_216 (Residu  (None, 30, 36)            9180
alBlock)

residual_block_217 (Residu  (None, 30, 36)            9180
alBlock)

residual_block_218 (Residu  (None, 30, 36)            9180
alBlock)

residual_block_219 (Residu  (None, 30, 36)            9180
alBlock)

residual_block_220 (Residu  (None, 30, 36)            9180
alBlock)
```

```
 residual_block_221 (Residu  (None, 30, 36)            9180
 alBlock)

 residual_block_222 (Residu  (None, 30, 36)            9180
 alBlock)

 residual_block_223 (Residu  (None, 30, 36)            9180
 alBlock)

 residual_block_224 (Residu  (None, 30, 36)            9180
 alBlock)

 global_average_pooling1d_2  (None, 36)                0
 7 (GlobalAveragePooling1D)

 dense_27 (Dense)            (None, 13)                481

=================================================================
Total params: 138433 (540.75 KB)
Trainable params: 138433 (540.75 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```python
'''add a custom callback to only update the model if
a new best validation score is reached'''

from keras.callbacks import ModelCheckpoint

best_validation_callback = ModelCheckpoint(
    'TCN_Model.tf',  # Filename to save the model in the SavedModel format
    monitor='val_accuracy',        # Monitor the validation accuracy
    verbose=1,                     # Logging level
    save_best_only=True,           # Save only when the validation accuracy improves
    mode='max',                    # Mode 'max' because we are monitoring 'val_accuracy'
```

```
    save_format='tf'                # Explicitly state to save in TensorFlow SavedModel format
)


# Fit model to data (training)
history = TCN_Model.fit(X_train, y_train,
                        epochs=100, batch_size=32,
                        validation_data=(X_test, y_test),
                       callbacks=[best_validation_callback])


import matplotlib.pyplot as plt
import numpy as np


# Plotting the learning curve
plt.figure(figsize=(10, 6))
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Temporal Convolutional Network Learning Curve')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.savefig('UIA_IMU_Temporal_Convolutional_Network_Gait_ID_8_1_2024_trials.eps', format='eps')
plt.savefig('UIA_IMU_Temporal_Convolutional_Network_Gait_ID_8_1_2024_trials.jpeg', format='jpeg')
plt.show()

# Save the model summary to a text file
with open('summary_UIA_IMU_temporal_convolutional_network_gait_id_model_8_1_2024.txt', 'w') as f:
    TCN_Model.summary(print_fn=lambda x: f.write(x + '\n'))
```

```
Epoch 1/100
120/120 [==============================] - ETA: 0s - loss: 1.0938 - accuracy: 0.6289
Epoch 1: val_accuracy improved from -inf to 0.80625, saving model to TCN_Model.tf
```

```
120/120 [==============================] - 29s 99ms/step - loss: 1.0938 - accuracy: 0.6289 - val_loss: 0.5574 - ...
    val_accuracy: 0.8062
Epoch 2/100
118/120 [============================>.] - ETA: 0s - loss: 0.4219 - accuracy: 0.8498
Epoch 2: val_accuracy improved from 0.80625 to 0.87292, saving model to TCN_Model.tf
120/120 [==============================] - 10s 83ms/step - loss: 0.4223 - accuracy: 0.8503 - val_loss: 0.3298 - ...
    val_accuracy: 0.8729
Epoch 3/100
118/120 [============================>.] - ETA: 0s - loss: 0.2940 - accuracy: 0.8988
Epoch 3: val_accuracy improved from 0.87292 to 0.90729, saving model to TCN_Model.tf
120/120 [==============================] - 10s 87ms/step - loss: 0.2924 - accuracy: 0.8995 - val_loss: 0.2410 - ...
    val_accuracy: 0.9073
Epoch 4/100
118/120 [============================>.] - ETA: 0s - loss: 0.2078 - accuracy: 0.9258
Epoch 4: val_accuracy improved from 0.90729 to 0.92708, saving model to TCN_Model.tf
120/120 [==============================] - 10s 87ms/step - loss: 0.2086 - accuracy: 0.9260 - val_loss: 0.2030 - ...
    val_accuracy: 0.9271
Epoch 5/100
118/120 [============================>.] - ETA: 0s - loss: 0.1765 - accuracy: 0.9356
Epoch 5: val_accuracy did not improve from 0.92708
120/120 [==============================] - 2s 20ms/step - loss: 0.1781 - accuracy: 0.9349 - val_loss: 0.2188 - ...
    val_accuracy: 0.9125
Epoch 6/100
118/120 [============================>.] - ETA: 0s - loss: 0.1631 - accuracy: 0.9465
Epoch 6: val_accuracy improved from 0.92708 to 0.95208, saving model to TCN_Model.tf
120/120 [==============================] - 10s 88ms/step - loss: 0.1627 - accuracy: 0.9469 - val_loss: 0.1296 - ...
    val_accuracy: 0.9521
Epoch 7/100
118/120 [============================>.] - ETA: 0s - loss: 0.1266 - accuracy: 0.9600
Epoch 7: val_accuracy improved from 0.95208 to 0.96458, saving model to TCN_Model.tf
120/120 [==============================] - 10s 87ms/step - loss: 0.1275 - accuracy: 0.9594 - val_loss: 0.1137 - ...
    val_accuracy: 0.9646
Epoch 8/100
119/120 [============================>.] - ETA: 0s - loss: 0.1025 - accuracy: 0.9651
Epoch 8: val_accuracy did not improve from 0.96458
120/120 [==============================] - 2s 20ms/step - loss: 0.1023 - accuracy: 0.9651 - val_loss: 0.1233 - ...
    val_accuracy: 0.9594
Epoch 9/100
118/120 [============================>.] - ETA: 0s - loss: 0.1449 - accuracy: 0.9473
```

```
Epoch 9: val_accuracy did not improve from 0.96458
120/120 [==============================] - 2s 20ms/step - loss: 0.1451 - accuracy: 0.9471 - val_loss: 0.1635 - ...
    val_accuracy: 0.9406
Epoch 10/100
118/120 [============================>.] - ETA: 0s - loss: 0.0982 - accuracy: 0.9650
Epoch 10: val_accuracy improved from 0.96458 to 0.96875, saving model to TCN_Model.tf
120/120 [==============================] - 10s 83ms/step - loss: 0.0981 - accuracy: 0.9651 - val_loss: 0.0819 - ...
    val_accuracy: 0.9688
Epoch 11/100
118/120 [============================>.] - ETA: 0s - loss: 0.0540 - accuracy: 0.9828
Epoch 11: val_accuracy did not improve from 0.96875
120/120 [==============================] - 2s 20ms/step - loss: 0.0535 - accuracy: 0.9831 - val_loss: 0.1301 - ...
    val_accuracy: 0.9510
Epoch 12/100
118/120 [============================>.] - ETA: 0s - loss: 0.0696 - accuracy: 0.9762
Epoch 12: val_accuracy improved from 0.96875 to 0.97292, saving model to TCN_Model.tf
120/120 [==============================] - 10s 88ms/step - loss: 0.0688 - accuracy: 0.9766 - val_loss: 0.0687 - ...
    val_accuracy: 0.9729
Epoch 13/100
120/120 [==============================] - ETA: 0s - loss: 0.0549 - accuracy: 0.9823
Epoch 13: val_accuracy improved from 0.97292 to 0.98438, saving model to TCN_Model.tf
120/120 [==============================] - 11s 88ms/step - loss: 0.0549 - accuracy: 0.9823 - val_loss: 0.0515 - ...
    val_accuracy: 0.9844
Epoch 14/100
118/120 [============================>.] - ETA: 0s - loss: 0.0380 - accuracy: 0.9865
Epoch 14: val_accuracy improved from 0.98438 to 0.99271, saving model to TCN_Model.tf
120/120 [==============================] - 10s 87ms/step - loss: 0.0377 - accuracy: 0.9867 - val_loss: 0.0291 - ...
    val_accuracy: 0.9927
Epoch 15/100
118/120 [============================>.] - ETA: 0s - loss: 0.0518 - accuracy: 0.9825
Epoch 15: val_accuracy did not improve from 0.99271
120/120 [==============================] - 2s 20ms/step - loss: 0.0516 - accuracy: 0.9826 - val_loss: 0.1880 - ...
    val_accuracy: 0.9427
Epoch 16/100
120/120 [==============================] - ETA: 0s - loss: 0.0739 - accuracy: 0.9737
Epoch 16: val_accuracy did not improve from 0.99271
120/120 [==============================] - 2s 20ms/step - loss: 0.0739 - accuracy: 0.9737 - val_loss: 0.0777 - ...
    val_accuracy: 0.9698
Epoch 17/100
```

```
118/120 [============================>.] - ETA: 0s - loss: 0.0272 - accuracy: 0.9923
Epoch 17: val_accuracy did not improve from 0.99271
120/120 [==============================] - 2s 20ms/step - loss: 0.0276 - accuracy: 0.9919 - val_loss: 0.0458 - ...
   val_accuracy: 0.9844
Epoch 18/100
118/120 [============================>.] - ETA: 0s - loss: 0.0532 - accuracy: 0.9841
Epoch 18: val_accuracy did not improve from 0.99271
120/120 [==============================] - 2s 20ms/step - loss: 0.0530 - accuracy: 0.9841 - val_loss: 0.0323 - ...
   val_accuracy: 0.9917
Epoch 19/100
118/120 [============================>.] - ETA: 0s - loss: 0.0388 - accuracy: 0.9868
Epoch 19: val_accuracy did not improve from 0.99271
120/120 [==============================] - 2s 20ms/step - loss: 0.0392 - accuracy: 0.9865 - val_loss: 0.1032 - ...
   val_accuracy: 0.9656
Epoch 20/100
119/120 [============================>.] - ETA: 0s - loss: 0.0922 - accuracy: 0.9751
Epoch 20: val_accuracy did not improve from 0.99271
120/120 [==============================] - 3s 21ms/step - loss: 0.0918 - accuracy: 0.9750 - val_loss: 0.0292 - ...
   val_accuracy: 0.9927
Epoch 21/100
118/120 [============================>.] - ETA: 0s - loss: 0.0317 - accuracy: 0.9923
Epoch 21: val_accuracy improved from 0.99271 to 0.99479, saving model to TCN_Model.tf
120/120 [==============================] - 11s 91ms/step - loss: 0.0313 - accuracy: 0.9924 - val_loss: 0.0227 - ...
   val_accuracy: 0.9948
Epoch 22/100
118/120 [============================>.] - ETA: 0s - loss: 0.0626 - accuracy: 0.9836
Epoch 22: val_accuracy did not improve from 0.99479
120/120 [==============================] - 3s 22ms/step - loss: 0.0625 - accuracy: 0.9836 - val_loss: 0.1955 - ...
   val_accuracy: 0.9458
Epoch 23/100
118/120 [============================>.] - ETA: 0s - loss: 0.0675 - accuracy: 0.9785
Epoch 23: val_accuracy did not improve from 0.99479
120/120 [==============================] - 3s 22ms/step - loss: 0.0669 - accuracy: 0.9786 - val_loss: 0.0432 - ...
   val_accuracy: 0.9844
Epoch 24/100
118/120 [============================>.] - ETA: 0s - loss: 0.0203 - accuracy: 0.9947
Epoch 24: val_accuracy did not improve from 0.99479
120/120 [==============================] - 3s 21ms/step - loss: 0.0202 - accuracy: 0.9948 - val_loss: 0.0836 - ...
   val_accuracy: 0.9802
```

```
Epoch 25/100
120/120 [==============================] - ETA: 0s - loss: 0.0125 - accuracy: 0.9966
Epoch 25: val_accuracy did not improve from 0.99479
120/120 [==============================] - 3s 22ms/step - loss: 0.0125 - accuracy: 0.9966 - val_loss: 0.0305 - ...
   val_accuracy: 0.9844
Epoch 26/100
118/120 [============================>.] - ETA: 0s - loss: 0.0109 - accuracy: 0.9976
Epoch 26: val_accuracy improved from 0.99479 to 0.99792, saving model to TCN_Model.tf
120/120 [==============================] - 11s 90ms/step - loss: 0.0108 - accuracy: 0.9977 - val_loss: 0.0119 - ...
   val_accuracy: 0.9979
Epoch 27/100
118/120 [============================>.] - ETA: 0s - loss: 0.0032 - accuracy: 0.9997
Epoch 27: val_accuracy did not improve from 0.99792
120/120 [==============================] - 3s 22ms/step - loss: 0.0034 - accuracy: 0.9995 - val_loss: 0.0224 - ...
   val_accuracy: 0.9927
Epoch 28/100
118/120 [============================>.] - ETA: 0s - loss: 0.0466 - accuracy: 0.9854
Epoch 28: val_accuracy did not improve from 0.99792
120/120 [==============================] - 3s 22ms/step - loss: 0.0465 - accuracy: 0.9852 - val_loss: 0.0322 - ...
   val_accuracy: 0.9906
Epoch 29/100
120/120 [==============================] - ETA: 0s - loss: 0.0398 - accuracy: 0.9875
Epoch 29: val_accuracy did not improve from 0.99792
120/120 [==============================] - 3s 22ms/step - loss: 0.0398 - accuracy: 0.9875 - val_loss: 0.0450 - ...
   val_accuracy: 0.9833
Epoch 30/100
118/120 [============================>.] - ETA: 0s - loss: 0.0492 - accuracy: 0.9860
Epoch 30: val_accuracy did not improve from 0.99792
120/120 [==============================] - 3s 22ms/step - loss: 0.0501 - accuracy: 0.9857 - val_loss: 0.0429 - ...
   val_accuracy: 0.9844
Epoch 31/100
118/120 [============================>.] - ETA: 0s - loss: 0.0981 - accuracy: 0.9727
Epoch 31: val_accuracy did not improve from 0.99792
120/120 [==============================] - 3s 22ms/step - loss: 0.0975 - accuracy: 0.9729 - val_loss: 0.1017 - ...
   val_accuracy: 0.9708
Epoch 32/100
118/120 [============================>.] - ETA: 0s - loss: 0.0246 - accuracy: 0.9944
Epoch 32: val_accuracy did not improve from 0.99792
120/120 [==============================] - 3s 22ms/step - loss: 0.0243 - accuracy: 0.9945 - val_loss: 0.0179 - ...
```

```
   val_accuracy: 0.9937
Epoch 33/100
119/120 [============================>.] - ETA: 0s - loss: 0.0036 - accuracy: 1.0000
Epoch 33: val_accuracy did not improve from 0.99792
120/120 [==============================] - 3s 22ms/step - loss: 0.0037 - accuracy: 1.0000 - val_loss: 0.0154 - ...
   val_accuracy: 0.9937
Epoch 34/100
118/120 [============================>.] - ETA: 0s - loss: 0.0021 - accuracy: 1.0000
Epoch 34: val_accuracy improved from 0.99792 to 0.99896, saving model to TCN_Model.tf
120/120 [==============================] - 11s 95ms/step - loss: 0.0020 - accuracy: 1.0000 - val_loss: 0.0085 - ...
   val_accuracy: 0.9990
Epoch 35/100
118/120 [============================>.] - ETA: 0s - loss: 0.0014 - accuracy: 1.0000
Epoch 35: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 22ms/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.0085 - ...
   val_accuracy: 0.9969
Epoch 36/100
118/120 [============================>.] - ETA: 0s - loss: 9.9586e-04 - accuracy: 1.0000
Epoch 36: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 22ms/step - loss: 9.9117e-04 - accuracy: 1.0000 - val_loss: 0.0076 - ...
   val_accuracy: 0.9990
Epoch 37/100
120/120 [==============================] - ETA: 0s - loss: 8.2781e-04 - accuracy: 1.0000
Epoch 37: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 23ms/step - loss: 8.2781e-04 - accuracy: 1.0000 - val_loss: 0.0072 - ...
   val_accuracy: 0.9979
Epoch 38/100
120/120 [==============================] - ETA: 0s - loss: 6.5603e-04 - accuracy: 1.0000
Epoch 38: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 24ms/step - loss: 6.5603e-04 - accuracy: 1.0000 - val_loss: 0.0070 - ...
   val_accuracy: 0.9969
Epoch 39/100
119/120 [============================>.] - ETA: 0s - loss: 5.3753e-04 - accuracy: 1.0000
Epoch 39: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 24ms/step - loss: 5.3427e-04 - accuracy: 1.0000 - val_loss: 0.0062 - ...
   val_accuracy: 0.9990
Epoch 40/100
118/120 [============================>.] - ETA: 0s - loss: 4.4465e-04 - accuracy: 1.0000
Epoch 40: val_accuracy did not improve from 0.99896
```

```
120/120 [==============================] - 3s 21ms/step - loss: 4.4848e-04 - accuracy: 1.0000 - val_loss: 0.0067 - ...
    val_accuracy: 0.9979
Epoch 41/100
119/120 [============================>.] - ETA: 0s - loss: 3.8334e-04 - accuracy: 1.0000
Epoch 41: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 22ms/step - loss: 3.9552e-04 - accuracy: 1.0000 - val_loss: 0.0063 - ...
    val_accuracy: 0.9969
Epoch 42/100
118/120 [============================>.] - ETA: 0s - loss: 3.5031e-04 - accuracy: 1.0000
Epoch 42: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 22ms/step - loss: 3.4807e-04 - accuracy: 1.0000 - val_loss: 0.0062 - ...
    val_accuracy: 0.9969
Epoch 43/100
118/120 [============================>.] - ETA: 0s - loss: 3.0506e-04 - accuracy: 1.0000
Epoch 43: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 22ms/step - loss: 3.0589e-04 - accuracy: 1.0000 - val_loss: 0.0057 - ...
    val_accuracy: 0.9990
Epoch 44/100
118/120 [============================>.] - ETA: 0s - loss: 2.7709e-04 - accuracy: 1.0000
Epoch 44: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 21ms/step - loss: 2.7525e-04 - accuracy: 1.0000 - val_loss: 0.0061 - ...
    val_accuracy: 0.9969
Epoch 45/100
119/120 [============================>.] - ETA: 0s - loss: 2.5041e-04 - accuracy: 1.0000
Epoch 45: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 23ms/step - loss: 2.5006e-04 - accuracy: 1.0000 - val_loss: 0.0057 - ...
    val_accuracy: 0.9990
Epoch 46/100
118/120 [============================>.] - ETA: 0s - loss: 2.2503e-04 - accuracy: 1.0000
Epoch 46: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 23ms/step - loss: 2.2355e-04 - accuracy: 1.0000 - val_loss: 0.0059 - ...
    val_accuracy: 0.9979
Epoch 47/100
118/120 [============================>.] - ETA: 0s - loss: 2.0204e-04 - accuracy: 1.0000
Epoch 47: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 22ms/step - loss: 2.0095e-04 - accuracy: 1.0000 - val_loss: 0.0054 - ...
    val_accuracy: 0.9979
Epoch 48/100
119/120 [============================>.] - ETA: 0s - loss: 1.8253e-04 - accuracy: 1.0000
```

```
Epoch 48: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 22ms/step - loss: 1.8218e-04 - accuracy: 1.0000 - val_loss: 0.0053 - ...
   val_accuracy: 0.9990
Epoch 49/100
118/120 [============================>.] - ETA: 0s - loss: 1.6711e-04 - accuracy: 1.0000
Epoch 49: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 22ms/step - loss: 1.6599e-04 - accuracy: 1.0000 - val_loss: 0.0049 - ...
   val_accuracy: 0.9990
Epoch 50/100
118/120 [============================>.] - ETA: 0s - loss: 1.5275e-04 - accuracy: 1.0000
Epoch 50: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 22ms/step - loss: 1.5138e-04 - accuracy: 1.0000 - val_loss: 0.0051 - ...
   val_accuracy: 0.9990
Epoch 51/100
118/120 [============================>.] - ETA: 0s - loss: 1.3836e-04 - accuracy: 1.0000
Epoch 51: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 22ms/step - loss: 1.3831e-04 - accuracy: 1.0000 - val_loss: 0.0054 - ...
   val_accuracy: 0.9979
Epoch 52/100
120/120 [==============================] - ETA: 0s - loss: 1.2818e-04 - accuracy: 1.0000
Epoch 52: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 22ms/step - loss: 1.2818e-04 - accuracy: 1.0000 - val_loss: 0.0051 - ...
   val_accuracy: 0.9990
Epoch 53/100
118/120 [============================>.] - ETA: 0s - loss: 1.1501e-04 - accuracy: 1.0000
Epoch 53: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 21ms/step - loss: 1.1517e-04 - accuracy: 1.0000 - val_loss: 0.0046 - ...
   val_accuracy: 0.9990
Epoch 54/100
118/120 [============================>.] - ETA: 0s - loss: 1.0339e-04 - accuracy: 1.0000
Epoch 54: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 22ms/step - loss: 1.0575e-04 - accuracy: 1.0000 - val_loss: 0.0050 - ...
   val_accuracy: 0.9979
Epoch 55/100
118/120 [============================>.] - ETA: 0s - loss: 9.7722e-05 - accuracy: 1.0000
Epoch 55: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 22ms/step - loss: 9.6890e-05 - accuracy: 1.0000 - val_loss: 0.0047 - ...
   val_accuracy: 0.9990
Epoch 56/100
```

```
118/120 [============================>.] - ETA: 0s - loss: 9.0075e-05 - accuracy: 1.0000
Epoch 56: val_accuracy did not improve from 0.99896
120/120 [==============================] - 3s 22ms/step - loss: 8.9409e-05 - accuracy: 1.0000 - val_loss: 0.0048 - ...
   val_accuracy: 0.9990
Epoch 57/100
119/120 [============================>.] - ETA: 0s - loss: 8.1762e-05 - accuracy: 1.0000
Epoch 57: val_accuracy improved from 0.99896 to 1.00000, saving model to TCN_Model.tf
120/120 [==============================] - 11s 94ms/step - loss: 8.2506e-05 - accuracy: 1.0000 - val_loss: 0.0045 - ...
   val_accuracy: 1.0000
Epoch 58/100
118/120 [============================>.] - ETA: 0s - loss: 7.6214e-05 - accuracy: 1.0000
Epoch 58: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 20ms/step - loss: 7.5877e-05 - accuracy: 1.0000 - val_loss: 0.0047 - ...
   val_accuracy: 0.9990
Epoch 59/100
118/120 [============================>.] - ETA: 0s - loss: 7.1116e-05 - accuracy: 1.0000
Epoch 59: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 21ms/step - loss: 7.0521e-05 - accuracy: 1.0000 - val_loss: 0.0045 - ...
   val_accuracy: 0.9990
Epoch 60/100
118/120 [============================>.] - ETA: 0s - loss: 6.5348e-05 - accuracy: 1.0000
Epoch 60: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 20ms/step - loss: 6.5364e-05 - accuracy: 1.0000 - val_loss: 0.0046 - ...
   val_accuracy: 0.9990
Epoch 61/100
118/120 [============================>.] - ETA: 0s - loss: 6.1008e-05 - accuracy: 1.0000
Epoch 61: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 20ms/step - loss: 6.0603e-05 - accuracy: 1.0000 - val_loss: 0.0046 - ...
   val_accuracy: 0.9990
Epoch 62/100
118/120 [============================>.] - ETA: 0s - loss: 5.5861e-05 - accuracy: 1.0000
Epoch 62: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 20ms/step - loss: 5.5882e-05 - accuracy: 1.0000 - val_loss: 0.0046 - ...
   val_accuracy: 0.9979
Epoch 63/100
118/120 [============================>.] - ETA: 0s - loss: 5.2038e-05 - accuracy: 1.0000
Epoch 63: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 20ms/step - loss: 5.1533e-05 - accuracy: 1.0000 - val_loss: 0.0043 - ...
   val_accuracy: 0.9990
```

```
Epoch 64/100
118/120 [============================>.] - ETA: 0s - loss: 4.7948e-05 - accuracy: 1.0000
Epoch 64: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 20ms/step - loss: 4.7505e-05 - accuracy: 1.0000 - val_loss: 0.0045 - ...
   val_accuracy: 0.9990
Epoch 65/100
120/120 [==============================] - ETA: 0s - loss: 4.4471e-05 - accuracy: 1.0000
Epoch 65: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 21ms/step - loss: 4.4471e-05 - accuracy: 1.0000 - val_loss: 0.0044 - ...
   val_accuracy: 0.9990
Epoch 66/100
118/120 [============================>.] - ETA: 0s - loss: 4.1237e-05 - accuracy: 1.0000
Epoch 66: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 20ms/step - loss: 4.1085e-05 - accuracy: 1.0000 - val_loss: 0.0042 - ...
   val_accuracy: 0.9990
Epoch 67/100
119/120 [============================>.] - ETA: 0s - loss: 3.8653e-05 - accuracy: 1.0000
Epoch 67: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 22ms/step - loss: 3.8504e-05 - accuracy: 1.0000 - val_loss: 0.0042 - ...
   val_accuracy: 1.0000
Epoch 68/100
118/120 [============================>.] - ETA: 0s - loss: 3.6111e-05 - accuracy: 1.0000
Epoch 68: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 20ms/step - loss: 3.5723e-05 - accuracy: 1.0000 - val_loss: 0.0044 - ...
   val_accuracy: 0.9979
Epoch 69/100
120/120 [==============================] - ETA: 0s - loss: 3.2913e-05 - accuracy: 1.0000
Epoch 69: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 20ms/step - loss: 3.2913e-05 - accuracy: 1.0000 - val_loss: 0.0044 - ...
   val_accuracy: 0.9979
Epoch 70/100
118/120 [============================>.] - ETA: 0s - loss: 3.0719e-05 - accuracy: 1.0000
Epoch 70: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 20ms/step - loss: 3.0582e-05 - accuracy: 1.0000 - val_loss: 0.0043 - ...
   val_accuracy: 0.9990
Epoch 71/100
118/120 [============================>.] - ETA: 0s - loss: 2.8519e-05 - accuracy: 1.0000
Epoch 71: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 20ms/step - loss: 2.8626e-05 - accuracy: 1.0000 - val_loss: 0.0043 - ...
```

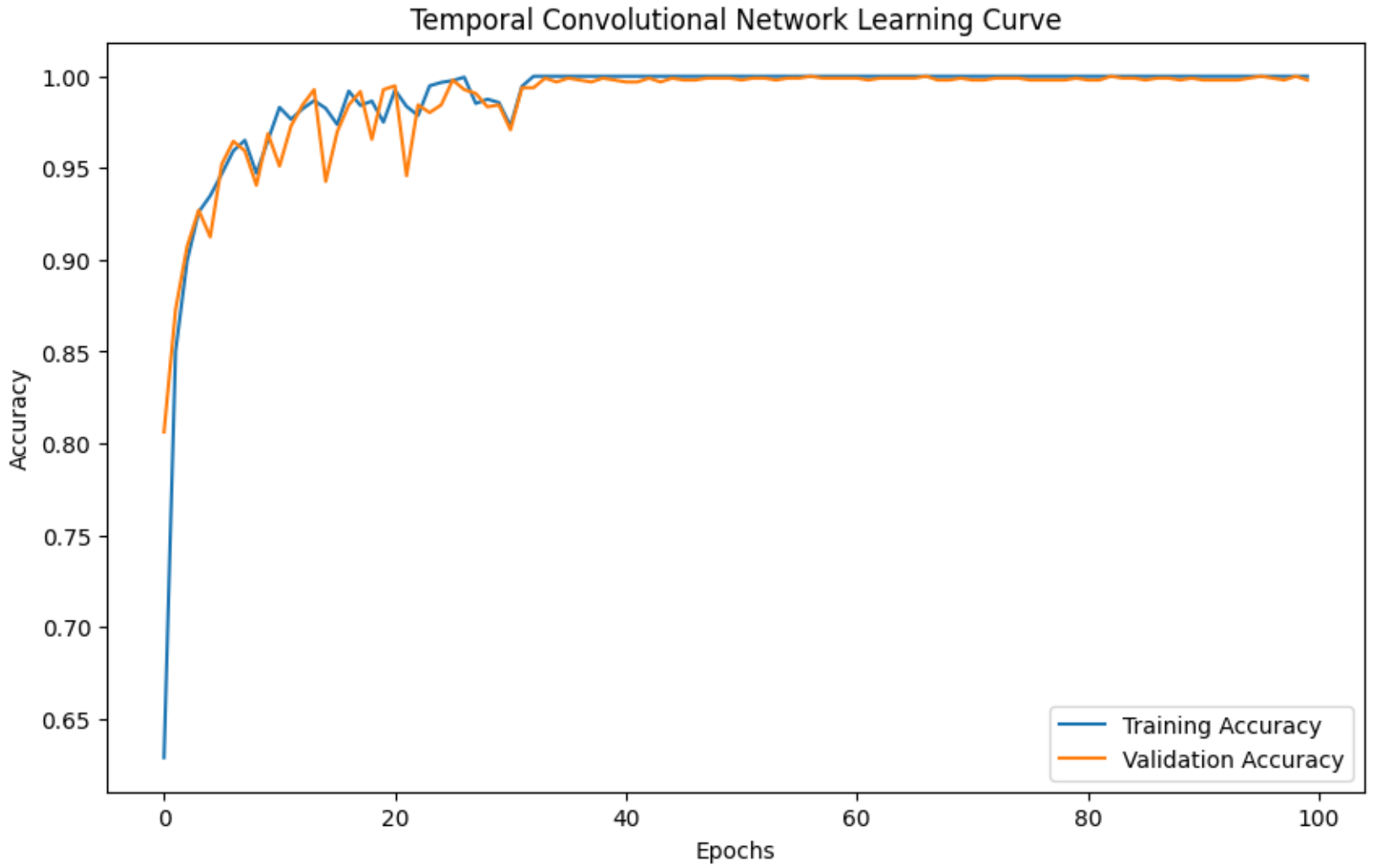
```
   val_accuracy: 0.9979
Epoch 72/100
118/120 [============================>.] - ETA: 0s - loss: 2.6499e-05 - accuracy: 1.0000
Epoch 72: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 20ms/step - loss: 2.6497e-05 - accuracy: 1.0000 - val_loss: 0.0043 - ...
   val_accuracy: 0.9979
Epoch 73/100
118/120 [============================>.] - ETA: 0s - loss: 2.5145e-05 - accuracy: 1.0000
Epoch 73: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 20ms/step - loss: 2.4901e-05 - accuracy: 1.0000 - val_loss: 0.0041 - ...
   val_accuracy: 0.9990
Epoch 74/100
118/120 [============================>.] - ETA: 0s - loss: 2.3236e-05 - accuracy: 1.0000
Epoch 74: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 20ms/step - loss: 2.3041e-05 - accuracy: 1.0000 - val_loss: 0.0042 - ...
   val_accuracy: 0.9990
Epoch 75/100
118/120 [============================>.] - ETA: 0s - loss: 2.1490e-05 - accuracy: 1.0000
Epoch 75: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 20ms/step - loss: 2.1532e-05 - accuracy: 1.0000 - val_loss: 0.0042 - ...
   val_accuracy: 0.9990
Epoch 76/100
118/120 [============================>.] - ETA: 0s - loss: 2.0094e-05 - accuracy: 1.0000
Epoch 76: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 20ms/step - loss: 2.0074e-05 - accuracy: 1.0000 - val_loss: 0.0044 - ...
   val_accuracy: 0.9979
Epoch 77/100
119/120 [============================>.] - ETA: 0s - loss: 1.8479e-05 - accuracy: 1.0000
Epoch 77: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 21ms/step - loss: 1.8493e-05 - accuracy: 1.0000 - val_loss: 0.0041 - ...
   val_accuracy: 0.9979
Epoch 78/100
118/120 [============================>.] - ETA: 0s - loss: 1.7332e-05 - accuracy: 1.0000
Epoch 78: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 20ms/step - loss: 1.7348e-05 - accuracy: 1.0000 - val_loss: 0.0042 - ...
   val_accuracy: 0.9979
Epoch 79/100
118/120 [============================>.] - ETA: 0s - loss: 1.6224e-05 - accuracy: 1.0000
Epoch 79: val_accuracy did not improve from 1.00000
```

```
120/120 [==============================] - 2s 20ms/step - loss: 1.6127e-05 - accuracy: 1.0000 - val_loss: 0.0043 - ...
   val_accuracy: 0.9979
Epoch 80/100
120/120 [==============================] - ETA: 0s - loss: 1.4928e-05 - accuracy: 1.0000
Epoch 80: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 21ms/step - loss: 1.4928e-05 - accuracy: 1.0000 - val_loss: 0.0039 - ...
   val_accuracy: 0.9990
Epoch 81/100
120/120 [==============================] - ETA: 0s - loss: 1.4120e-05 - accuracy: 1.0000
Epoch 81: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 21ms/step - loss: 1.4120e-05 - accuracy: 1.0000 - val_loss: 0.0041 - ...
   val_accuracy: 0.9979
Epoch 82/100
118/120 [============================>.] - ETA: 0s - loss: 1.3157e-05 - accuracy: 1.0000
Epoch 82: val_accuracy did not improve from 1.00000
120/120 [==============================] - 2s 21ms/step - loss: 1.3061e-05 - accuracy: 1.0000 - val_loss: 0.0042 - ...
   val_accuracy: 0.9979
Epoch 83/100
118/120 [============================>.] - ETA: 0s - loss: 1.2087e-05 - accuracy: 1.0000
Epoch 83: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 22ms/step - loss: 1.2246e-05 - accuracy: 1.0000 - val_loss: 0.0038 - ...
   val_accuracy: 1.0000
Epoch 84/100
118/120 [============================>.] - ETA: 0s - loss: 1.1480e-05 - accuracy: 1.0000
Epoch 84: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 22ms/step - loss: 1.1373e-05 - accuracy: 1.0000 - val_loss: 0.0041 - ...
   val_accuracy: 0.9990
Epoch 85/100
119/120 [============================>.] - ETA: 0s - loss: 1.0707e-05 - accuracy: 1.0000
Epoch 85: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 22ms/step - loss: 1.0651e-05 - accuracy: 1.0000 - val_loss: 0.0038 - ...
   val_accuracy: 0.9990
Epoch 86/100
118/120 [============================>.] - ETA: 0s - loss: 9.8509e-06 - accuracy: 1.0000
Epoch 86: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 22ms/step - loss: 9.9544e-06 - accuracy: 1.0000 - val_loss: 0.0042 - ...
   val_accuracy: 0.9979
Epoch 87/100
118/120 [============================>.] - ETA: 0s - loss: 9.2316e-06 - accuracy: 1.0000
```

```
Epoch 87: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 22ms/step - loss: 9.1618e-06 - accuracy: 1.0000 - val_loss: 0.0037 - ...
   val_accuracy: 0.9990
Epoch 88/100
118/120 [============================>.] - ETA: 0s - loss: 8.5453e-06 - accuracy: 1.0000
Epoch 88: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 22ms/step - loss: 8.6367e-06 - accuracy: 1.0000 - val_loss: 0.0039 - ...
   val_accuracy: 0.9990
Epoch 89/100
120/120 [==============================] - ETA: 0s - loss: 8.1294e-06 - accuracy: 1.0000
Epoch 89: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 22ms/step - loss: 8.1294e-06 - accuracy: 1.0000 - val_loss: 0.0041 - ...
   val_accuracy: 0.9979
Epoch 90/100
118/120 [============================>.] - ETA: 0s - loss: 7.3485e-06 - accuracy: 1.0000
Epoch 90: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 22ms/step - loss: 7.5102e-06 - accuracy: 1.0000 - val_loss: 0.0038 - ...
   val_accuracy: 0.9990
Epoch 91/100
118/120 [============================>.] - ETA: 0s - loss: 6.9839e-06 - accuracy: 1.0000
Epoch 91: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 22ms/step - loss: 7.0601e-06 - accuracy: 1.0000 - val_loss: 0.0044 - ...
   val_accuracy: 0.9979
Epoch 92/100
118/120 [============================>.] - ETA: 0s - loss: 6.6032e-06 - accuracy: 1.0000
Epoch 92: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 24ms/step - loss: 6.6065e-06 - accuracy: 1.0000 - val_loss: 0.0042 - ...
   val_accuracy: 0.9979
Epoch 93/100
118/120 [============================>.] - ETA: 0s - loss: 6.1081e-06 - accuracy: 1.0000
Epoch 93: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 22ms/step - loss: 6.1232e-06 - accuracy: 1.0000 - val_loss: 0.0041 - ...
   val_accuracy: 0.9979
Epoch 94/100
118/120 [============================>.] - ETA: 0s - loss: 5.8260e-06 - accuracy: 1.0000
Epoch 94: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 22ms/step - loss: 5.7447e-06 - accuracy: 1.0000 - val_loss: 0.0039 - ...
   val_accuracy: 0.9979
Epoch 95/100
```

```
118/120 [============================>.] - ETA: 0s - loss: 5.3492e-06 - accuracy: 1.0000
Epoch 95: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 22ms/step - loss: 5.3227e-06 - accuracy: 1.0000 - val_loss: 0.0037 - ...
    val_accuracy: 0.9990
Epoch 96/100
119/120 [============================>.] - ETA: 0s - loss: 5.0127e-06 - accuracy: 1.0000
Epoch 96: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 22ms/step - loss: 4.9991e-06 - accuracy: 1.0000 - val_loss: 0.0037 - ...
    val_accuracy: 1.0000
Epoch 97/100
118/120 [============================>.] - ETA: 0s - loss: 4.7154e-06 - accuracy: 1.0000
Epoch 97: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 22ms/step - loss: 4.6723e-06 - accuracy: 1.0000 - val_loss: 0.0042 - ...
    val_accuracy: 0.9990
Epoch 98/100
118/120 [============================>.] - ETA: 0s - loss: 4.3383e-06 - accuracy: 1.0000
Epoch 98: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 22ms/step - loss: 4.3302e-06 - accuracy: 1.0000 - val_loss: 0.0039 - ...
    val_accuracy: 0.9979
Epoch 99/100
118/120 [============================>.] - ETA: 0s - loss: 4.1375e-06 - accuracy: 1.0000
Epoch 99: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 22ms/step - loss: 4.0916e-06 - accuracy: 1.0000 - val_loss: 0.0036 - ...
    val_accuracy: 1.0000
Epoch 100/100
118/120 [============================>.] - ETA: 0s - loss: 3.8558e-06 - accuracy: 1.0000
Epoch 100: val_accuracy did not improve from 1.00000
120/120 [==============================] - 3s 22ms/step - loss: 3.8182e-06 - accuracy: 1.0000 - val_loss: 0.0040 - ...
    val_accuracy: 0.9979
```


Model performance does in fact still reach 100%, even though model complexity (and size) is now reduced by around 50%. Additional simplification has been attempted, but this resulted in the model making one (or more) erroneous prediction(s) on the validation dataset.




```
TCN_Model = load_model('/kaggle/input/uia_id_tcn_simplified_100_val/keras/.h5/1/TCN_Model.h5')

# Load best version of model
from tensorflow.keras.models import load_model
```

```
TCN_Model = load_model('TCN_Model.tf')

TCN_Model.save('TCN_Model.h5')


# Evaluate the model on the test set
test_loss, test_accuracy = TCN_Model.evaluate(X_test, y_test)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')

import tensorflow as tf
classes_X = TCN_Model.predict(X_test)

classes_X.shape

#classes_X[0]
'''The predict function outputs a probability for each label.
Need to extrapolate the max value per prediction.'''

classes_X = np.argmax(classes_X, axis=1)


cm = tf.math.confusion_matrix(labels=y_test, predictions=classes_X, num_classes=output_dim)

cm
```

```
/opt/conda/lib/python3.10/site-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an HDF5 file via `mod
  saving_api.save_model(
```

```
30/30 [==============================] - 1s 6ms/step - loss: 0.0045 - accuracy: 1.0000
Test Loss: 0.004527099896222353, Test Accuracy: 1.0
30/30 [==============================] - 1s 6ms/step
```

```
<tf.Tensor: shape=(13, 13), dtype=int32, numpy=
array([[100,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,  78,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
```

```
 [  0,   0,   0,  66,   0,   0,   0,   0,   0,   0,   0,   0,   0],
 [  0,   0,   0,   0,  67,   0,   0,   0,   0,   0,   0,   0,   0],
 [  0,   0,   0,   0,   0,  68,   0,   0,   0,   0,   0,   0,   0],
 [  0,   0,   0,   0,   0,   0,  75,   0,   0,   0,   0,   0,   0],
 [  0,   0,   0,   0,   0,   0,   0,  68,   0,   0,   0,   0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,  70,   0,   0,   0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,  80,   0,   0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  62,   0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  74,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  82]],
dtype=int32)>
```

As expected from the 100% accuracy of the model: No confusion.

Reading the matrix: Rows are actual classes, columns are predictions. Diagonal is correct predictions for the given class.

Comparative analysis: Model performance against other ML architectures.

Overarching goal of this body of work is to opitmize for real-time systems. Computational efficiency, accuracy, inference time, FLOPs (Floating Point Operations Per Second) == Computational effort per prediction, Memory usage, Energy consumption and Batch processing efficiency are some of the possible measurands for this task.

CNN model

```
from keras.callbacks import ModelCheckpoint

best_validation_callback = ModelCheckpoint(
    'Strict_CNN_model.tf',  # Filename to save the model in the SavedModel format
    monitor='val_accuracy',        # Monitor the validation accuracy
    verbose=1,                     # Logging level
    save_best_only=True,           # Save only when the validation accuracy improves
    mode='max',                    # Mode 'max' because we are monitoring 'val_accuracy'
    save_format='tf'               # Explicitly state to save in TensorFlow SavedModel forma
)


# Fit CNN to data (training)
Strict_CNN_model_history = Strict_CNN_model.fit(X_train, y_train,
                         epochs=600, batch_size=32,
                         validation_data=(X_test, y_test),
                        callbacks=[best_validation_callback])


import matplotlib.pyplot as plt
import numpy as np


# Plotting the learning curve
plt.figure(figsize=(10, 6))
plt.plot(Strict_CNN_model_history.history['accuracy'], label='Training Accuracy')
plt.plot(Strict_CNN_model_history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Convolutional Network Learning Curve for UIA IMU Identification')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.savefig('UIA_IMU_Strict_CNN_Gait_ID_14_1_2024_trials.eps', format='eps')
plt.savefig('UIA_IMU_Strict_CNN_Gait_ID_14_1_2024_trials.jpeg', format='jpeg')
plt.show()

# Save the Strict_CNN_model summary to a text file
with open('summary_UIA_IMU_UIA_IMU_Strict_CNN_Gait_ID_14_1_2024_trials.txt', 'w') as f:
    Strict_CNN_model.summary(print_fn=lambda x: f.write(x + '\n'))


# Load the best version of model

from tensorflow.keras.models import load_model

Strict_CNN_model = load_model('Strict_CNN_model.tf')



# Evaluate best version of model on the test set

test_loss, test_accuracy = Strict_CNN_model.evaluate(X_test, y_test)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')
```

```
import tensorflow as tf
classes_X = Strict_CNN_model.predict(X_test)

classes_X.shape

#classes_X[0]
'''The predict function outputs a probability for each label.
Need to extrapolate the max value per prediction.'''

classes_X = np.argmax(classes_X, axis=1)


cm_Strict_CNN_model = tf.math.confusion_matrix(labels=y_test, predictions=classes_X, num_cla

cm_Strict_CNN_model
```

```
    Epoch 1/600
120/120 [==============================] - ETA: 0s - loss: 2.3912 - ...
    accuracy: 0.4716
Epoch 1: val_accuracy improved from -inf to 0.51979, saving model to ...
    Strict_CNN_model.tf
120/120 [==============================] - 8s 36ms/step - loss: 2.3912 - ...
    accuracy: 0.4716 - val_loss: 2.7486 - val_accuracy: 0.5198
Epoch 2/600
116/120 [============================>.] - ETA: 0s - loss: 1.7731 - ...
    accuracy: 0.6123
Epoch 2: val_accuracy improved from 0.51979 to 0.68646, saving model to ...
    Strict_CNN_model.tf
120/120 [==============================] - 5s 43ms/step - loss: 1.7683 - ...
    accuracy: 0.6125 - val_loss: 1.5267 - val_accuracy: 0.6865
Epoch 3/600
120/120 [==============================] - ETA: 0s - loss: 1.5970 - ...
    accuracy: 0.6695
Epoch 3: val_accuracy improved from 0.68646 to 0.73021, saving model to ...
    Strict_CNN_model.tf
120/120 [==============================] - 4s 32ms/step - loss: 1.5970 - ...
    accuracy: 0.6695 - val_loss: 1.3600 - val_accuracy: 0.7302
Epoch 4/600
113/120 [===========================>..] - ETA: 0s - loss: 1.4431 - ...
    accuracy: 0.7240
Epoch 4: val_accuracy improved from 0.73021 to 0.76979, saving model to ...
    Strict_CNN_model.tf
120/120 [==============================] - 4s 32ms/step - loss: 1.4484 - ...
    accuracy: 0.7250 - val_loss: 1.2356 - val_accuracy: 0.7698
Epoch 5/600
117/120 [============================>.] - ETA: 0s - loss: 1.3980 - ...
    accuracy: 0.7444
Epoch 5: val_accuracy improved from 0.76979 to 0.82812, saving model to ...
    Strict_CNN_model.tf
120/120 [==============================] - 4s 33ms/step - loss: 1.3946 - ...
    accuracy: 0.7453 - val_loss: 1.0849 - val_accuracy: 0.8281
Epoch 6/600
114/120 [===========================>..] - ETA: 0s - loss: 1.3276 - ...
    accuracy: 0.7552
Epoch 6: val_accuracy improved from 0.82812 to 0.83438, saving model to ...
    Strict_CNN_model.tf
120/120 [==============================] - 4s 32ms/step - loss: 1.3318 - ...
    accuracy: 0.7544 - val_loss: 1.0967 - val_accuracy: 0.8344
Epoch 7/600
```

```
120/120 [==============================] - ETA: 0s - loss: 1.2905 - ...
    accuracy: 0.7857
Epoch 7: val_accuracy improved from 0.83438 to 0.86146, saving model to ...
    Strict_CNN_model.tf
120/120 [==============================] - 4s 32ms/step - loss: 1.2905 - ...
    accuracy: 0.7857 - val_loss: 1.0078 - val_accuracy: 0.8615
Epoch 8/600
117/120 [============================>.] - ETA: 0s - loss: 1.2627 - ...
    accuracy: 0.7804
Epoch 8: val_accuracy did not improve from 0.86146
120/120 [==============================] - 1s 8ms/step - loss: 1.2655 - ...
    accuracy: 0.7794 - val_loss: 1.1400 - val_accuracy: 0.8146
Epoch 9/600
113/120 [===========================>..] - ETA: 0s - loss: 1.2363 - ...
    accuracy: 0.7848
Epoch 9: val_accuracy improved from 0.86146 to 0.88021, saving model to ...
    Strict_CNN_model.tf
120/120 [==============================] - 4s 32ms/step - loss: 1.2327 - ...
    accuracy: 0.7867 - val_loss: 0.8881 - val_accuracy: 0.8802
Epoch 10/600
113/120 [===========================>..] - ETA: 0s - loss: 1.1843 - ...
    accuracy: 0.7965
Epoch 10: val_accuracy did not improve from 0.88021
120/120 [==============================] - 1s 7ms/step - loss: 1.1814 - ...
    accuracy: 0.7969 - val_loss: 0.9767 - val_accuracy: 0.8573
Epoch 11/600
113/120 [===========================>..] - ETA: 0s - loss: 1.1608 - ...
    accuracy: 0.8106
Epoch 11: val_accuracy improved from 0.88021 to 0.88646, saving model to ...
    Strict_CNN_model.tf
120/120 [==============================] - 5s 41ms/step - loss: 1.1546 - ...
    accuracy: 0.8115 - val_loss: 0.8388 - val_accuracy: 0.8865
Epoch 12/600
113/120 [===========================>..] - ETA: 0s - loss: 1.1545 - ...
    accuracy: 0.7987
Epoch 12: val_accuracy did not improve from 0.88646
120/120 [==============================] - 1s 8ms/step - loss: 1.1548 - ...
    accuracy: 0.7995 - val_loss: 1.0081 - val_accuracy: 0.8521
Epoch 13/600
113/120 [===========================>..] - ETA: 0s - loss: 1.1227 - ...
    accuracy: 0.8103
Epoch 13: val_accuracy did not improve from 0.88646
120/120 [==============================] - 1s 8ms/step - loss: 1.1173 - ...
    accuracy: 0.8107 - val_loss: 1.1161 - val_accuracy: 0.7771
Epoch 14/600
113/120 [===========================>..] - ETA: 0s - loss: 1.1074 - ...
    accuracy: 0.8067
Epoch 14: val_accuracy did not improve from 0.88646
120/120 [==============================] - 1s 8ms/step - loss: 1.1121 - ...
    accuracy: 0.8049 - val_loss: 0.9641 - val_accuracy: 0.8438
Epoch 15/600
113/120 [===========================>..] - ETA: 0s - loss: 1.0859 - ...
    accuracy: 0.8200
Epoch 15: val_accuracy did not improve from 0.88646
120/120 [==============================] - 1s 8ms/step - loss: 1.0888 - ...
    accuracy: 0.8190 - val_loss: 1.0513 - val_accuracy: 0.8490
Epoch 16/600
113/120 [===========================>..] - ETA: 0s - loss: 1.0710 - ...
    accuracy: 0.8241
Epoch 16: val_accuracy did not improve from 0.88646
120/120 [==============================] - 1s 8ms/step - loss: 1.0675 - ...
```

```
    accuracy: 0.8242 - val_loss: 0.9441 - val_accuracy: 0.8500
Epoch 17/600
113/120 [===========================>..] - ETA: 0s - loss: 1.0449 - ...
    accuracy: 0.8296
Epoch 17: val_accuracy improved from 0.88646 to 0.91250, saving model to ...
    Strict_CNN_model.tf
120/120 [==============================] - 4s 32ms/step - loss: 1.0471 - ...
    accuracy: 0.8289 - val_loss: 0.7906 - val_accuracy: 0.9125
Epoch 18/600
117/120 [============================>.] - ETA: 0s - loss: 1.0590 - ...
    accuracy: 0.8275
Epoch 18: val_accuracy did not improve from 0.91250
120/120 [==============================] - 1s 8ms/step - loss: 1.0527 - ...
    accuracy: 0.8297 - val_loss: 0.7673 - val_accuracy: 0.9104
Epoch 19/600
115/120 [===========================>..] - ETA: 0s - loss: 1.0097 - ...
    accuracy: 0.8389
Epoch 19: val_accuracy did not improve from 0.91250
120/120 [==============================] - 1s 8ms/step - loss: 1.0042 - ...
    accuracy: 0.8401 - val_loss: 0.8113 - val_accuracy: 0.8844
Epoch 20/600
113/120 [===========================>..] - ETA: 0s - loss: 1.0173 - ...
    accuracy: 0.8321
Epoch 20: val_accuracy did not improve from 0.91250
120/120 [==============================] - 1s 8ms/step - loss: 1.0241 - ...
    accuracy: 0.8292 - val_loss: 0.8611 - val_accuracy: 0.8677
Epoch 21/600
113/120 [===========================>..] - ETA: 0s - loss: 1.0269 - ...
    accuracy: 0.8338
Epoch 21: val_accuracy did not improve from 0.91250
120/120 [==============================] - 1s 7ms/step - loss: 1.0316 - ...
    accuracy: 0.8318 - val_loss: 0.8457 - val_accuracy: 0.8854
Epoch 22/600
120/120 [==============================] - ETA: 0s - loss: 1.0444 - ...
    accuracy: 0.8339
Epoch 22: val_accuracy did not improve from 0.91250
120/120 [==============================] - 1s 8ms/step - loss: 1.0444 - ...
    accuracy: 0.8339 - val_loss: 0.7607 - val_accuracy: 0.9104
Epoch 23/600
113/120 [===========================>..] - ETA: 0s - loss: 0.9819 - ...
    accuracy: 0.8507
Epoch 23: val_accuracy improved from 0.91250 to 0.91771, saving model to ...
    Strict_CNN_model.tf
120/120 [==============================] - 4s 32ms/step - loss: 0.9914 - ...
    accuracy: 0.8490 - val_loss: 0.7104 - val_accuracy: 0.9177
Epoch 24/600
113/120 [===========================>..] - ETA: 0s - loss: 1.0093 - ...
    accuracy: 0.8382
Epoch 24: val_accuracy did not improve from 0.91771
120/120 [==============================] - 1s 7ms/step - loss: 1.0081 - ...
    accuracy: 0.8380 - val_loss: 0.7937 - val_accuracy: 0.8781
Epoch 25/600
120/120 [==============================] - ETA: 0s - loss: 1.0150 - ...
    accuracy: 0.8378
Epoch 25: val_accuracy did not improve from 0.91771
120/120 [==============================] - 1s 8ms/step - loss: 1.0150 - ...
    accuracy: 0.8378 - val_loss: 0.7813 - val_accuracy: 0.9042
Epoch 26/600
113/120 [===========================>..] - ETA: 0s - loss: 0.9792 - ...
    accuracy: 0.8496
Epoch 26: val_accuracy did not improve from 0.91771
```

```
120/120 [==============================] - 1s 7ms/step - loss: 0.9786 - ...
    accuracy: 0.8505 - val_loss: 0.7198 - val_accuracy: 0.9062
Epoch 27/600
117/120 [============================>.] - ETA: 0s - loss: 1.0145 - ...
    accuracy: 0.8381
Epoch 27: val_accuracy did not improve from 0.91771
120/120 [==============================] - 1s 8ms/step - loss: 1.0141 - ...
    accuracy: 0.8380 - val_loss: 1.4370 - val_accuracy: 0.7792
Epoch 28/600
120/120 [==============================] - ETA: 0s - loss: 1.0058 - ...
    accuracy: 0.8336
Epoch 28: val_accuracy did not improve from 0.91771
120/120 [==============================] - 1s 8ms/step - loss: 1.0058 - ...
    accuracy: 0.8336 - val_loss: 0.7777 - val_accuracy: 0.9073
Epoch 29/600
120/120 [==============================] - ETA: 0s - loss: 0.9821 - ...
    accuracy: 0.8445
Epoch 29: val_accuracy improved from 0.91771 to 0.92396, saving model to ...
    Strict_CNN_model.tf
120/120 [==============================] - 4s 32ms/step - loss: 0.9821 - ...
    accuracy: 0.8445 - val_loss: 0.7140 - val_accuracy: 0.9240
Epoch 30/600
120/120 [==============================] - ETA: 0s - loss: 0.9989 - ...
    accuracy: 0.8378
Epoch 30: val_accuracy did not improve from 0.92396
120/120 [==============================] - 1s 8ms/step - loss: 0.9989 - ...
    accuracy: 0.8378 - val_loss: 0.8505 - val_accuracy: 0.8698
Epoch 31/600
116/120 [============================>.] - ETA: 0s - loss: 1.0182 - ...
    accuracy: 0.8354
Epoch 31: val_accuracy did not improve from 0.92396
120/120 [==============================] - 1s 8ms/step - loss: 1.0249 - ...
    accuracy: 0.8336 - val_loss: 0.7199 - val_accuracy: 0.9219
Epoch 32/600
117/120 [============================>.] - ETA: 0s - loss: 0.9689 - ...
    accuracy: 0.8480
Epoch 32: val_accuracy did not improve from 0.92396
120/120 [==============================] - 1s 8ms/step - loss: 0.9690 - ...
    accuracy: 0.8484 - val_loss: 0.7353 - val_accuracy: 0.9031
Epoch 33/600
115/120 [===========================>..] - ETA: 0s - loss: 0.9473 - ...
    accuracy: 0.8565
Epoch 33: val_accuracy did not improve from 0.92396
120/120 [==============================] - 1s 9ms/step - loss: 0.9564 - ...
    accuracy: 0.8555 - val_loss: 0.7679 - val_accuracy: 0.9083
Epoch 34/600
118/120 [============================>.] - ETA: 0s - loss: 0.9488 - ...
    accuracy: 0.8578
Epoch 34: val_accuracy did not improve from 0.92396
120/120 [==============================] - 1s 8ms/step - loss: 0.9514 - ...
    accuracy: 0.8576 - val_loss: 0.7158 - val_accuracy: 0.9187
Epoch 35/600
113/120 [===========================>..] - ETA: 0s - loss: 0.9648 - ...
    accuracy: 0.8532
Epoch 35: val_accuracy improved from 0.92396 to 0.94167, saving model to ...
    Strict_CNN_model.tf
120/120 [==============================] - 4s 32ms/step - loss: 0.9632 - ...
    accuracy: 0.8536 - val_loss: 0.6739 - val_accuracy: 0.9417
Epoch 36/600
120/120 [==============================] - ETA: 0s - loss: 0.9228 - ...
    accuracy: 0.8570
```

```
Epoch 36: val_accuracy did not improve from 0.94167
120/120 [==============================] - 1s 7ms/step - loss: 0.9228 - ...
    accuracy: 0.8570 - val_loss: 0.7823 - val_accuracy: 0.8875

.............

Epoch 43/600
113/120 [===========================>..] - ETA: 0s - loss: 0.9459 - ...
    accuracy: 0.8529
Epoch 43: val_accuracy did not improve from 0.94167
120/120 [==============================] - 1s 7ms/step - loss: 0.9370 - ...
    accuracy: 0.8552 - val_loss: 0.7793 - val_accuracy: 0.8990
Epoch 44/600
120/120 [==============================] - ETA: 0s - loss: 0.9373 - ...
    accuracy: 0.8581
Epoch 44: val_accuracy did not improve from 0.94167
120/120 [==============================] - 1s 8ms/step - loss: 0.9373 - ...
    accuracy: 0.8581 - val_loss: 0.7320 - val_accuracy: 0.9146
Epoch 45/600
113/120 [===========================>..] - ETA: 0s - loss: 0.9213 - ...
    accuracy: 0.8551
Epoch 45: val_accuracy did not improve from 0.94167
120/120 [==============================] - 1s 7ms/step - loss: 0.9208 - ...
    accuracy: 0.8560 - val_loss: 0.7823 - val_accuracy: 0.8927
Epoch 46/600
113/120 [===========================>..] - ETA: 0s - loss: 0.9428 - ...
    accuracy: 0.8556
Epoch 46: val_accuracy did not improve from 0.94167
120/120 [==============================] - 1s 7ms/step - loss: 0.9336 - ...
    accuracy: 0.8576 - val_loss: 0.7048 - val_accuracy: 0.9167
Epoch 47/600
120/120 [==============================] - ETA: 0s - loss: 0.9339 - ...
    accuracy: 0.8622
Epoch 47: val_accuracy did not improve from 0.94167
120/120 [==============================] - 1s 8ms/step - loss: 0.9339 - ...
    accuracy: 0.8622 - val_loss: 0.6862 - val_accuracy: 0.9187
Epoch 48/600
113/120 [===========================>..] - ETA: 0s - loss: 0.9434 - ...
    accuracy: 0.8595
Epoch 48: val_accuracy did not improve from 0.94167
120/120 [==============================] - 1s 7ms/step - loss: 0.9534 - ...
    accuracy: 0.8562 - val_loss: 0.6735 - val_accuracy: 0.9281
Epoch 49/600
116/120 [============================>.] - ETA: 0s - loss: 0.9103 - ...
    accuracy: 0.8607
Epoch 49: val_accuracy did not improve from 0.94167
120/120 [==============================] - 1s 8ms/step - loss: 0.9072 - ...
    accuracy: 0.8609 - val_loss: 0.7838 - val_accuracy: 0.8781
Epoch 50/600
113/120 [===========================>..] - ETA: 0s - loss: 0.9330 - ...
    accuracy: 0.8626
Epoch 50: val_accuracy did not improve from 0.94167
120/120 [==============================] - 1s 8ms/step - loss: 0.9235 - ...
    accuracy: 0.8643 - val_loss: 0.6483 - val_accuracy: 0.9302
Epoch 51/600
113/120 [===========================>..] - ETA: 0s - loss: 0.9249 - ...
    accuracy: 0.8598
Epoch 51: val_accuracy did not improve from 0.94167
120/120 [==============================] - 1s 7ms/step - loss: 0.9202 - ...
    accuracy: 0.8612 - val_loss: 0.6912 - val_accuracy: 0.9250
Epoch 52/600
```

```
113/120 [===========================>..] - ETA: 0s - loss: 0.9469 - ...
    accuracy: 0.8534
Epoch 52: val_accuracy did not improve from 0.94167
120/120 [==============================] - 1s 7ms/step - loss: 0.9591 - ...
    accuracy: 0.8531 - val_loss: 0.7579 - val_accuracy: 0.9156

.............


119/120 [===========================>.] - ETA: 0s - loss: 0.7838 - ...
    accuracy: 0.8929
Epoch 244: val_accuracy did not improve from 0.96042
120/120 [==============================] - 1s 8ms/step - loss: 0.7816 - ...
    accuracy: 0.8935 - val_loss: 0.5611 - val_accuracy: 0.9594
Epoch 245/600
113/120 [===========================>..] - ETA: 0s - loss: 0.8120 - ...
    accuracy: 0.8894
Epoch 245: val_accuracy did not improve from 0.96042
120/120 [==============================] - 1s 7ms/step - loss: 0.8127 - ...
    accuracy: 0.8888 - val_loss: 0.5714 - val_accuracy: 0.9500

...............

Epoch 265/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7639 - ...
    accuracy: 0.8941
Epoch 265: val_accuracy improved from 0.96042 to 0.96458, saving model to ...
    Strict_CNN_model.tf
120/120 [==============================] - 4s 32ms/step - loss: 0.7630 - ...
    accuracy: 0.8940 - val_loss: 0.5252 - val_accuracy: 0.9646
Epoch 266/600
118/120 [===========================>.] - ETA: 0s - loss: 0.8337 - ...
    accuracy: 0.8745
Epoch 266: val_accuracy did not improve from 0.96458
120/120 [==============================] - 1s 8ms/step - loss: 0.8312 - ...
    accuracy: 0.8755 - val_loss: 0.5653 - val_accuracy: 0.9604
Epoch 267/600
113/120 [===========================>..] - ETA: 0s - loss: 0.8296 - ...
    accuracy: 0.8816
Epoch 267: val_accuracy did not improve from 0.96458
120/120 [==============================] - 1s 8ms/step - loss: 0.8310 - ...
    accuracy: 0.8810 - val_loss: 0.5939 - val_accuracy: 0.9573
Epoch 268/600
115/120 [===========================>..] - ETA: 0s - loss: 0.8091 - ...
    accuracy: 0.8853
Epoch 268: val_accuracy did not improve from 0.96458
120/120 [==============================] - 1s 8ms/step - loss: 0.8067 - ...
    accuracy: 0.8852 - val_loss: 0.6359 - val_accuracy: 0.9458
Epoch 269/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7968 - ...
    accuracy: 0.8924
Epoch 269: val_accuracy did not improve from 0.96458
120/120 [==============================] - 1s 8ms/step - loss: 0.7920 - ...
    accuracy: 0.8938 - val_loss: 0.5680 - val_accuracy: 0.9510
Epoch 270/600
113/120 [===========================>..] - ETA: 0s - loss: 0.8016 - ...
    accuracy: 0.8874
Epoch 270: val_accuracy did not improve from 0.96458
120/120 [==============================] - 1s 8ms/step - loss: 0.8113 - ...
    accuracy: 0.8849 - val_loss: 0.6183 - val_accuracy: 0.9406
Epoch 271/600
```

```
113/120 [==========================>..] - ETA: 0s - loss: 0.7953 - ...
    accuracy: 0.8874
Epoch 271: val_accuracy did not improve from 0.96458
120/120 [==============================] - 1s 8ms/step - loss: 0.7993 - ...
    accuracy: 0.8865 - val_loss: 0.5723 - val_accuracy: 0.9500
Epoch 272/600
113/120 [==========================>..] - ETA: 0s - loss: 0.7851 - ...
    accuracy: 0.8863
Epoch 272: val_accuracy did not improve from 0.96458
120/120 [==============================] - 1s 7ms/step - loss: 0.7841 - ...
    accuracy: 0.8859 - val_loss: 0.6220 - val_accuracy: 0.9469
Epoch 273/600
120/120 [==============================] - ETA: 0s - loss: 0.7869 - ...
    accuracy: 0.8924
Epoch 273: val_accuracy did not improve from 0.96458
120/120 [==============================] - 1s 7ms/step - loss: 0.7869 - ...
    accuracy: 0.8924 - val_loss: 0.5708 - val_accuracy: 0.9479
Epoch 274/600
113/120 [==========================>..] - ETA: 0s - loss: 0.7574 - ...
    accuracy: 0.8952
Epoch 274: val_accuracy did not improve from 0.96458
120/120 [==============================] - 1s 8ms/step - loss: 0.7625 - ...
    accuracy: 0.8940 - val_loss: 0.5801 - val_accuracy: 0.9406
Epoch 275/600
113/120 [==========================>..] - ETA: 0s - loss: 0.7695 - ...
    accuracy: 0.8985
Epoch 275: val_accuracy did not improve from 0.96458
120/120 [==============================] - 1s 8ms/step - loss: 0.7798 - ...
    accuracy: 0.8964 - val_loss: 0.5660 - val_accuracy: 0.9531
Epoch 276/600
120/120 [==============================] - ETA: 0s - loss: 0.8037 - ...
    accuracy: 0.8841
Epoch 276: val_accuracy did not improve from 0.96458
120/120 [==============================] - 1s 8ms/step - loss: 0.8037 - ...
    accuracy: 0.8841 - val_loss: 0.6078 - val_accuracy: 0.9500
Epoch 277/600
113/120 [==========================>..] - ETA: 0s - loss: 0.8082 - ...
    accuracy: 0.8880
Epoch 277: val_accuracy did not improve from 0.96458
120/120 [==============================] - 1s 7ms/step - loss: 0.8017 - ...
    accuracy: 0.8896 - val_loss: 0.5662 - val_accuracy: 0.9563
Epoch 278/600
113/120 [==========================>..] - ETA: 0s - loss: 0.7729 - ...
    accuracy: 0.8899
Epoch 278: val_accuracy did not improve from 0.96458
120/120 [==============================] - 1s 8ms/step - loss: 0.7780 - ...
    accuracy: 0.8883 - val_loss: 0.6120 - val_accuracy: 0.9323
Epoch 279/600
118/120 [==========================>.] - ETA: 0s - loss: 0.7703 - ...
    accuracy: 0.8954
Epoch 279: val_accuracy did not improve from 0.96458
120/120 [==============================] - 1s 9ms/step - loss: 0.7689 - ...
    accuracy: 0.8958 - val_loss: 0.5697 - val_accuracy: 0.9490
Epoch 280/600
117/120 [==========================>.] - ETA: 0s - loss: 0.7699 - ...
    accuracy: 0.8950
Epoch 280: val_accuracy did not improve from 0.96458
120/120 [==============================] - 1s 8ms/step - loss: 0.7659 - ...
    accuracy: 0.8961 - val_loss: 0.5929 - val_accuracy: 0.9500
Epoch 281/600
113/120 [==========================>..] - ETA: 0s - loss: 0.8535 - ...
```

```
    accuracy: 0.8814
Epoch 281: val_accuracy improved from 0.96458 to 0.96667, saving model to ...
    Strict_CNN_model.tf
120/120 [==============================] - 4s 31ms/step - loss: 0.8500 - ...
    accuracy: 0.8831 - val_loss: 0.5514 - val_accuracy: 0.9667
Epoch 282/600
120/120 [==============================] - ETA: 0s - loss: 0.8053 - ...
    accuracy: 0.8917
Epoch 282: val_accuracy did not improve from 0.96667
120/120 [==============================] - 1s 7ms/step - loss: 0.8053 - ...
    accuracy: 0.8917 - val_loss: 0.5832 - val_accuracy: 0.9531
Epoch 283/600
120/120 [==============================] - ETA: 0s - loss: 0.8232 - ...
    accuracy: 0.8909
Epoch 283: val_accuracy did not improve from 0.96667
120/120 [==============================] - 1s 8ms/step - loss: 0.8232 - ...
    accuracy: 0.8909 - val_loss: 0.5739 - val_accuracy: 0.9406
Epoch 284/600
113/120 [===========================>..] - ETA: 0s - loss: 0.8056 - ...
    accuracy: 0.8866
Epoch 284: val_accuracy did not improve from 0.96667
120/120 [==============================] - 1s 7ms/step - loss: 0.8008 - ...
    accuracy: 0.8875 - val_loss: 0.5455 - val_accuracy: 0.9573
Epoch 285/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7835 - ...
    accuracy: 0.8905
Epoch 285: val_accuracy did not improve from 0.96667
120/120 [==============================] - 1s 7ms/step - loss: 0.7812 - ...
    accuracy: 0.8904 - val_loss: 0.6452 - val_accuracy: 0.9365
Epoch 286/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7726 - ...
    accuracy: 0.8977
Epoch 286: val_accuracy did not improve from 0.96667
120/120 [==============================] - 1s 8ms/step - loss: 0.7732 - ...
    accuracy: 0.8974 - val_loss: 0.5703 - val_accuracy: 0.9490
Epoch 287/600
116/120 [============================>.] - ETA: 0s - loss: 0.7662 - ...
    accuracy: 0.8947
Epoch 287: val_accuracy did not improve from 0.96667
120/120 [==============================] - 1s 8ms/step - loss: 0.7645 - ...
    accuracy: 0.8945 - val_loss: 0.6031 - val_accuracy: 0.9333

.................

Epoch 473/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7349 - ...
    accuracy: 0.9029
Epoch 473: val_accuracy did not improve from 0.96667
120/120 [==============================] - 1s 7ms/step - loss: 0.7299 - ...
    accuracy: 0.9039 - val_loss: 0.5400 - val_accuracy: 0.9521
Epoch 474/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7797 - ...
    accuracy: 0.8888
Epoch 474: val_accuracy did not improve from 0.96667
120/120 [==============================] - 1s 7ms/step - loss: 0.7784 - ...
    accuracy: 0.8888 - val_loss: 0.6487 - val_accuracy: 0.9292
Epoch 475/600
117/120 [============================>.] - ETA: 0s - loss: 0.7519 - ...
    accuracy: 0.8993
Epoch 475: val_accuracy did not improve from 0.96667
120/120 [==============================] - 1s 8ms/step - loss: 0.7549 - ...
```

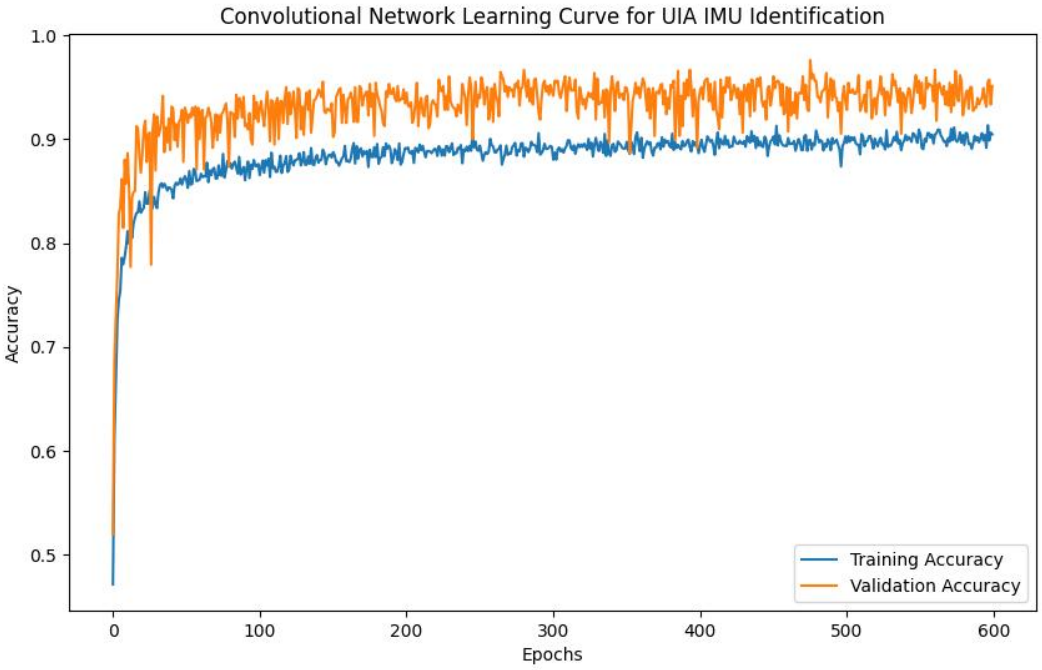
```
   accuracy: 0.8987 - val_loss: 0.6404 - val_accuracy: 0.9271
Epoch 476/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7372 - ...
   accuracy: 0.8966
Epoch 476: val_accuracy improved from 0.96667 to 0.97604, saving model to ...
   Strict_CNN_model.tf
120/120 [==============================] - 4s 32ms/step - loss: 0.7287 - ...
   accuracy: 0.8992 - val_loss: 0.4898 - val_accuracy: 0.9760

.................


115/120 [===========================>..] - ETA: 0s - loss: 0.6998 - ...
   accuracy: 0.9046
Epoch 561: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7019 - ...
   accuracy: 0.9044 - val_loss: 0.5220 - val_accuracy: 0.9667
Epoch 562/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7229 - ...
   accuracy: 0.9051
Epoch 562: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7246 - ...
   accuracy: 0.9055 - val_loss: 0.6684 - val_accuracy: 0.9177
Epoch 563/600
120/120 [==============================] - ETA: 0s - loss: 0.7227 - ...
   accuracy: 0.9091
Epoch 563: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7227 - ...
   accuracy: 0.9091 - val_loss: 0.5312 - val_accuracy: 0.9521
Epoch 564/600
119/120 [============================>.] - ETA: 0s - loss: 0.7011 - ...
   accuracy: 0.9089
Epoch 564: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.6999 - ...
   accuracy: 0.9091 - val_loss: 0.5368 - val_accuracy: 0.9542
Epoch 565/600
120/120 [==============================] - ETA: 0s - loss: 0.7241 - ...
   accuracy: 0.9034
Epoch 565: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7241 - ...
   accuracy: 0.9034 - val_loss: 0.5254 - val_accuracy: 0.9479
Epoch 566/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7051 - ...
   accuracy: 0.9029
Epoch 566: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7149 - ...
   accuracy: 0.9013 - val_loss: 0.5554 - val_accuracy: 0.9427
Epoch 567/600
114/120 [===========================>..] - ETA: 0s - loss: 0.7278 - ...
   accuracy: 0.8961
Epoch 567: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7224 - ...
   accuracy: 0.8977 - val_loss: 0.5265 - val_accuracy: 0.9510
Epoch 568/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7318 - ...
   accuracy: 0.8960
Epoch 568: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7268 - ...
   accuracy: 0.8982 - val_loss: 0.6153 - val_accuracy: 0.9333
Epoch 569/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7452 - ...
```

```
    accuracy: 0.8949
Epoch 569: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7491 - ...
    accuracy: 0.8932 - val_loss: 0.5802 - val_accuracy: 0.9458
Epoch 570/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7148 - ...
    accuracy: 0.9021
Epoch 570: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 7ms/step - loss: 0.7216 - ...
    accuracy: 0.9010 - val_loss: 0.6125 - val_accuracy: 0.9323
Epoch 571/600
118/120 [============================>.] - ETA: 0s - loss: 0.7396 - ...
    accuracy: 0.8975
Epoch 571: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7428 - ...
    accuracy: 0.8974 - val_loss: 0.5248 - val_accuracy: 0.9542
Epoch 572/600
120/120 [==============================] - ETA: 0s - loss: 0.6809 - ...
    accuracy: 0.9099
Epoch 572: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 7ms/step - loss: 0.6809 - ...
    accuracy: 0.9099 - val_loss: 0.6046 - val_accuracy: 0.9260
Epoch 573/600
120/120 [==============================] - ETA: 0s - loss: 0.7368 - ...
    accuracy: 0.8990
Epoch 573: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7368 - ...
    accuracy: 0.8990 - val_loss: 0.5433 - val_accuracy: 0.9521
Epoch 574/600
113/120 [===========================>..] - ETA: 0s - loss: 0.6616 - ...
    accuracy: 0.9126
Epoch 574: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 7ms/step - loss: 0.6680 - ...
    accuracy: 0.9112 - val_loss: 0.6115 - val_accuracy: 0.9323
Epoch 575/600
120/120 [==============================] - ETA: 0s - loss: 0.7145 - ...
    accuracy: 0.8995
Epoch 575: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7145 - ...
    accuracy: 0.8995 - val_loss: 0.5086 - val_accuracy: 0.9656
Epoch 576/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7204 - ...
    accuracy: 0.8957
Epoch 576: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7139 - ...
    accuracy: 0.8974 - val_loss: 0.5524 - val_accuracy: 0.9417
Epoch 577/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7210 - ...
    accuracy: 0.8996
Epoch 577: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 7ms/step - loss: 0.7231 - ...
    accuracy: 0.8995 - val_loss: 0.5468 - val_accuracy: 0.9365
Epoch 578/600
116/120 [============================>.] - ETA: 0s - loss: 0.7001 - ...
    accuracy: 0.9071
Epoch 578: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.6964 - ...
    accuracy: 0.9065 - val_loss: 0.5358 - val_accuracy: 0.9615
Epoch 579/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7221 - ...
    accuracy: 0.8963
```

```
Epoch 579: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 7ms/step - loss: 0.7216 - ...
    accuracy: 0.8966 - val_loss: 0.5260 - val_accuracy: 0.9563
Epoch 580/600
117/120 [============================>.] - ETA: 0s - loss: 0.6971 - ...
    accuracy: 0.9041
Epoch 580: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 9ms/step - loss: 0.6958 - ...
    accuracy: 0.9039 - val_loss: 0.5691 - val_accuracy: 0.9229
Epoch 581/600
115/120 [===========================>..] - ETA: 0s - loss: 0.7206 - ...
    accuracy: 0.8943
Epoch 581: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7163 - ...
    accuracy: 0.8945 - val_loss: 0.5797 - val_accuracy: 0.9323
Epoch 582/600
113/120 [===========================>..] - ETA: 0s - loss: 0.6993 - ...
    accuracy: 0.9046
Epoch 582: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 7ms/step - loss: 0.7008 - ...
    accuracy: 0.9060 - val_loss: 0.5721 - val_accuracy: 0.9438
Epoch 583/600
120/120 [==============================] - ETA: 0s - loss: 0.6775 - ...
    accuracy: 0.9052
Epoch 583: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.6775 - ...
    accuracy: 0.9052 - val_loss: 0.5249 - val_accuracy: 0.9469
Epoch 584/600
118/120 [============================>.] - ETA: 0s - loss: 0.7071 - ...
    accuracy: 0.8951
Epoch 584: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7056 - ...
    accuracy: 0.8953 - val_loss: 0.5958 - val_accuracy: 0.9292
Epoch 585/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7519 - ...
    accuracy: 0.8963
Epoch 585: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7474 - ...
    accuracy: 0.8958 - val_loss: 0.5321 - val_accuracy: 0.9500
Epoch 586/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7491 - ...
    accuracy: 0.8919
Epoch 586: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7505 - ...
    accuracy: 0.8906 - val_loss: 0.5445 - val_accuracy: 0.9500
Epoch 587/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7381 - ...
    accuracy: 0.8952
Epoch 587: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7369 - ...
    accuracy: 0.8956 - val_loss: 0.5996 - val_accuracy: 0.9271
Epoch 588/600
119/120 [============================>.] - ETA: 0s - loss: 0.7176 - ...
    accuracy: 0.9031
Epoch 588: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7167 - ...
    accuracy: 0.9031 - val_loss: 0.5788 - val_accuracy: 0.9302
Epoch 589/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7257 - ...
    accuracy: 0.9027
Epoch 589: val_accuracy did not improve from 0.97604
```

```
120/120 [==============================] - 1s 8ms/step - loss: 0.7407 - ...
    accuracy: 0.8984 - val_loss: 0.6256 - val_accuracy: 0.9302
Epoch 590/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7231 - ...
    accuracy: 0.8999
Epoch 590: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7177 - ...
    accuracy: 0.9018 - val_loss: 0.5612 - val_accuracy: 0.9396
Epoch 591/600
116/120 [============================>.] - ETA: 0s - loss: 0.7334 - ...
    accuracy: 0.8971
Epoch 591: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7331 - ...
    accuracy: 0.8969 - val_loss: 0.5834 - val_accuracy: 0.9333
Epoch 592/600
119/120 [============================>.] - ETA: 0s - loss: 0.7251 - ...
    accuracy: 0.9005
Epoch 592: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7234 - ...
    accuracy: 0.9010 - val_loss: 0.5363 - val_accuracy: 0.9375
Epoch 593/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7051 - ...
    accuracy: 0.9079
Epoch 593: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7128 - ...
    accuracy: 0.9057 - val_loss: 0.6091 - val_accuracy: 0.9396
Epoch 594/600
119/120 [============================>.] - ETA: 0s - loss: 0.7229 - ...
    accuracy: 0.8989
Epoch 594: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7236 - ...
    accuracy: 0.8987 - val_loss: 0.5220 - val_accuracy: 0.9490
Epoch 595/600
120/120 [==============================] - ETA: 0s - loss: 0.7068 - ...
    accuracy: 0.9039
Epoch 595: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7068 - ...
    accuracy: 0.9039 - val_loss: 0.5836 - val_accuracy: 0.9365
Epoch 596/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7582 - ...
    accuracy: 0.8933
Epoch 596: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7596 - ...
    accuracy: 0.8917 - val_loss: 0.5723 - val_accuracy: 0.9312
Epoch 597/600
113/120 [===========================>..] - ETA: 0s - loss: 0.6629 - ...
    accuracy: 0.9140
Epoch 597: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.6646 - ...
    accuracy: 0.9133 - val_loss: 0.5116 - val_accuracy: 0.9552
Epoch 598/600
113/120 [===========================>..] - ETA: 0s - loss: 0.7157 - ...
    accuracy: 0.8996
Epoch 598: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7217 - ...
    accuracy: 0.8987 - val_loss: 0.5119 - val_accuracy: 0.9573
Epoch 599/600
115/120 [===========================>..] - ETA: 0s - loss: 0.6935 - ...
    accuracy: 0.9087
Epoch 599: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.7019 - ...
```

```
   accuracy: 0.9062 - val_loss: 0.5398 - val_accuracy: 0.9333
Epoch 600/600
118/120 [============================>.] - ETA: 0s - loss: 0.6988 - ...
   accuracy: 0.9041
Epoch 600: val_accuracy did not improve from 0.97604
120/120 [==============================] - 1s 8ms/step - loss: 0.6965 - ...
   accuracy: 0.9047 - val_loss: 0.5043 - val_accuracy: 0.9510
```



```
30/30 [==============================] - 0s 3ms/step - loss: 0.4898 - accuracy: 0.9760
Test Loss: 0.4897750914096832, Test Accuracy: 0.9760416746139526
30/30 [==============================] - 0s 2ms/step
<tf.Tensor: shape=(13, 13), dtype=int32, numpy=
array([[100,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,  74,   0,   0,   0,   1,   0,   2,   1,   0,   0,   0,   0],
       [  0,   0,  66,   0,   0,   0,   0,   4,   0,   0,   0,   0,   0],
       [  0,   0,   0,  64,   1,   1,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,  66,   0,   0,   0,   0,   1,   0,   0,   0],
       [  0,   0,   0,   0,   0,  67,   0,   1,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   2,   0,  73,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  68,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  70,   0,   0,   0,   0],
       [  0,   0,   0,   0,   1,   0,   0,   0,   0,  78,   1,   0,   0],
       [  0,   0,   0,   0,   3,   0,   0,   0,   0,   0,  59,   0,   0],
       [  0,   0,   0,   0,   0,   0,   1,   0,   0,   0,   3,  70,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  82]],
      dtype=int32)>
```

Attention-based Convolutional LSTM

```
#Variables:

input_dim = X.shape[2]
sequence_length = X.shape[1]
input_shape = (sequence_length, input_dim)
output_dim = len(np.unique(y))
nb_filters = 64
kernel_size = 3
dilations = [1, 2, 3, 6]
nb_stacks = 3
dropout_rate = 0.2

#model-specific variables:

#Dense layer, to exand num combinations
Units_Dens_Expand_Dim = 80

Units_Unity_Connected_Layer = 64

#Unity_Conected_Layer = Conv

#Conv layer, create new abstractions from features
Filters = 100

Kernel_Size = 2


# Pooling
Pool_Size = 2


#LSTM, learn patterns in data
LSTM_Units = 64

Recurrent_Dropout = 0.2




# dropout, prevent overfitting
DropOut_Rate = 0.4
from keras.layers import Input, Conv1D, LSTM, Concatenate, Flatten
from tensorflow.keras.models import Model
import tensorflow as tf

class MyAttention(Layer):
    def __init__(self, **kwargs):
        super(MyAttention, self).__init__(**kwargs)

    def build(self, input_shape):
        sequence_dim = input_shape[1] if input_shape[1] is not None else 1
```

```
        feature_dim = input_shape[-1] if input_shape[-1] is not None else 1
        initializer = tf.keras.initializers.GlorotUniform(seed=None)  # You can choose a di
        self.kernel = self.add_weight(
            name='kernel',
            shape=(feature_dim, 1),
            initializer=initializer,
            trainable=True
        )
        super(MyAttention, self).build(input_shape)

    def call(self, x):
        scores = tf.matmul(x, self.kernel)
        attention_weights = tf.nn.softmax(scores, axis=1)
        attended_output = x * attention_weights
        return attended_output


# Input layer
In_Layer = Input(shape=input_shape)

# Convolutional layer

C_Layer = Conv1D(data_format= 'channels_last', padding= 'same', filters = Filters, kernel_si

C_Layer_2 = Conv1D(data_format= 'channels_last', padding= 'same', filters = Filters*2, kerne

C_Layer_3 = Conv1D(data_format= 'channels_last', padding= 'same', filters = Filters*2, kerne

Max_Pool = MaxPooling1D(data_format='channels_first',
                                padding= 'same',
                                pool_size=4)(C_Layer_3)

dropout_layer = Dropout(rate=DropOut_Rate)(Max_Pool)

Unity_Connected_Layer = Conv1D(filters=Units_Unity_Connected_Layer,
                                kernel_size=1,
                                use_bias=False,
                               data_format='channels_last')(dropout_layer)

# LSTM layer
lstm_layer_ = LSTM(units=LSTM_Units, return_sequences=True, activation='tanh',
                   recurrent_activation='sigmoid', recurrent_dropout=Recurrent_Dropout,
                   unroll=False, use_bias=True, return_state=False)(Unity_Connected_Layer)
'''
lstm_layer_1 = LSTM(units=LSTM_Units, return_sequences=True, activation='tanh',
                   recurrent_activation='sigmoid', recurrent_dropout=Recurrent_Dropout,
                   unroll=False, use_bias=True, return_state=False)(lstm_layer_)
'''

attention_layer = MyAttention()

attended_output = attention_layer(lstm_layer_)
```

```
# Dropout layer
dropout_layer_2 = Dropout(rate=DropOut_Rate)(attended_output)


# LSTM layer
lstm_layer_1 = LSTM(units=LSTM_Units, return_sequences=True, activation='tanh',
                    recurrent_activation='sigmoid', recurrent_dropout=Recurrent_Dropout,
                    unroll=False, use_bias=True, return_state=False)(dropout_layer_2)
'''
lstm_layer_1 = LSTM(units=LSTM_Units, return_sequences=True, activation='tanh',
                    recurrent_activation='sigmoid', recurrent_dropout=Recurrent_Dropout,
                    unroll=False, use_bias=True, return_state=False)(lstm_layer_)
'''

attention_layer_1 = MyAttention()

attended_output_1 = attention_layer_1(lstm_layer_1)


# Dropout layer
dropout_layer_2 = Dropout(rate=DropOut_Rate)(attended_output_1)


# Flatten layer
flat_layer = Flatten()(dropout_layer_2)

# Output layer
OUT = Dense(output_dim, activation='softmax')(flat_layer)




# Build the Attentive_Conv_LSTM
Attentive_Conv_LSTM_model = Model(inputs=In_Layer, outputs=OUT)



# Compile the Attentive_Conv_LSTM
Attentive_Conv_LSTM_model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])




Attentive_Conv_LSTM_model.summary()

from keras.callbacks import ModelCheckpoint

best_validation_callback = ModelCheckpoint(
    'Attentive_Conv_LSTM_model.tf',  # Filename to save the model in the SavedModel format
    monitor='val_accuracy',         # Monitor the validation accuracy
    verbose=1,                      # Logging level
    save_best_only=True,            # Save only when the validation accuracy improves
```

```
    mode='max',                    # Mode 'max' because we are monitoring 'val_accuracy'
    save_format='tf'               # Explicitly state to save in TensorFlow SavedModel form
)



# Fit CNN to data (training)
Strict_CNN_model_history = Attentive_Conv_LSTM_model.fit(X_train, y_train,
                         epochs=600, batch_size=32,
                         validation_data=(X_test, y_test),
                        callbacks=[best_validation_callback])


import matplotlib.pyplot as plt
import numpy as np

Attentive_Conv_LSTM_model = Strict_CNN_model_history


# Plotting the learning curve
plt.figure(figsize=(10, 6))
plt.plot(Attentive_Conv_LSTM_model_history.history['accuracy'], label='Training Accuracy')
plt.plot(Attentive_Conv_LSTM_model_history.history['val_accuracy'], label='Validation Accura
plt.title('Attention-based Convolutional LSTM Learning Curve for UIA IMU Identification')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.savefig('UIA_IMU_Attentive_Conv_LSTM_Gait_ID_14_1_2024_trials.eps', format='eps')
plt.savefig('UIA_IMU_Attentive_Conv_LSTM_Gait_ID_14_1_2024_trials.jpeg', format='jpeg')
plt.show()

# Save the Attentive_Conv_LSTM_model summary to a text file
with open('summary_Attentive_Conv_LSTM_Gait_ID_14_1_2024_trials.txt', 'w') as f:
    Attentive_Conv_LSTM_model.summary(print_fn=lambda x: f.write(x + '\n'))


# Evaluate the model on the test set
test_loss, test_accuracy = Attentive_Conv_LSTM_model.evaluate(X_test, y_test)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')

import tensorflow as tf
classes_X = Attentive_Conv_LSTM_model.predict(X_test)
j
classes_X.shape

#classes_X[0]
'''The predict function outputs a probability for each label.
Need to extrapolate the max value per prediction.'''

classes_X = np.argmax(classes_X, axis=1)


Attentive_Conv_LSTM_model_cm = tf.math.confusion_matrix(labels=y_test, predictions=classes_X
```

```
Attentive_Conv_LSTM_model_cm

Training output here.

Attentive_Conv_LSTM_model_history = Strict_CNN_model_history

Attentive_Conv_LSTM_model = load_model('Attentive_Conv_LSTM_model.tf')

'''

forgot to swap out name in code before training. Re-load best version of model,
and swap the name for the 'history' variable from fit call (training)


This fixes the issue.

'''


# Plotting the learning curve
plt.figure(figsize=(10, 6))
plt.plot(Attentive_Conv_LSTM_model_history.history['accuracy'], label='Training Accuracy')
plt.plot(Attentive_Conv_LSTM_model_history.history['val_accuracy'], label='Validation Accura
plt.title('Attention-based Convolutional LSTM Learning Curve for UIA IMU Identification')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.savefig('UIA_IMU_Attentive_Conv_LSTM_Gait_ID_14_1_2024_trials.eps', format='eps')
plt.savefig('UIA_IMU_Attentive_Conv_LSTM_Gait_ID_14_1_2024_trials.jpeg', format='jpeg')
plt.show()




# Evaluate the model on the test set
test_loss, test_accuracy = Attentive_Conv_LSTM_model.evaluate(X_test, y_test)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')

import tensorflow as tf
classes_X = Attentive_Conv_LSTM_model.predict(X_test)

classes_X.shape

#classes_X[0]
'''The predict function outputs a probability for each label.
Need to extrapolate the max value per prediction.'''

classes_X = np.argmax(classes_X, axis=1)


Attentive_Conv_LSTM_model_cm = tf.math.confusion_matrix(labels=y_test, predictions=classes_X

Attentive_Conv_LSTM_model_cm
```

```
    Epoch 1/600
```

```
120/120 [==============================] - ETA: 0s - loss: 2.2392 - ...
    accuracy: 0.1917
Epoch 1: val_accuracy improved from -inf to 0.25625, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 22s 140ms/step - loss: 2.2392 - ...
    accuracy: 0.1917 - val_loss: 2.0078 - val_accuracy: 0.2562
Epoch 2/600
120/120 [==============================] - ETA: 0s - loss: 1.8749 - ...
    accuracy: 0.2859
Epoch 2: val_accuracy improved from 0.25625 to 0.32708, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 132ms/step - loss: 1.8749 - ...
    accuracy: 0.2859 - val_loss: 1.7634 - val_accuracy: 0.3271
Epoch 3/600
120/120 [==============================] - ETA: 0s - loss: 1.7140 - ...
    accuracy: 0.3464
Epoch 3: val_accuracy improved from 0.32708 to 0.39687, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 134ms/step - loss: 1.7140 - ...
    accuracy: 0.3464 - val_loss: 1.6496 - val_accuracy: 0.3969
Epoch 4/600
120/120 [==============================] - ETA: 0s - loss: 1.6917 - ...
    accuracy: 0.3667
Epoch 4: val_accuracy did not improve from 0.39687
120/120 [==============================] - 11s 90ms/step - loss: 1.6917 - ...
    accuracy: 0.3667 - val_loss: 1.7683 - val_accuracy: 0.3292
Epoch 5/600
120/120 [==============================] - ETA: 0s - loss: 1.5165 - ...
    accuracy: 0.4456
Epoch 5: val_accuracy improved from 0.39687 to 0.44688, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 134ms/step - loss: 1.5165 - ...
    accuracy: 0.4456 - val_loss: 1.4481 - val_accuracy: 0.4469
Epoch 6/600
120/120 [==============================] - ETA: 0s - loss: 1.3454 - ...
    accuracy: 0.4969
Epoch 6: val_accuracy improved from 0.44688 to 0.50521, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 131ms/step - loss: 1.3454 - ...
    accuracy: 0.4969 - val_loss: 1.3692 - val_accuracy: 0.5052
Epoch 7/600
120/120 [==============================] - ETA: 0s - loss: 1.7532 - ...
    accuracy: 0.3826
Epoch 7: val_accuracy did not improve from 0.50521
120/120 [==============================] - 11s 90ms/step - loss: 1.7532 - ...
    accuracy: 0.3826 - val_loss: 1.8160 - val_accuracy: 0.3187
Epoch 8/600
120/120 [==============================] - ETA: 0s - loss: 1.6445 - ...
    accuracy: 0.3974
Epoch 8: val_accuracy did not improve from 0.50521
120/120 [==============================] - 11s 90ms/step - loss: 1.6445 - ...
    accuracy: 0.3974 - val_loss: 1.6988 - val_accuracy: 0.3167
Epoch 9/600
120/120 [==============================] - ETA: 0s - loss: 1.4347 - ...
    accuracy: 0.4573
Epoch 9: val_accuracy did not improve from 0.50521
120/120 [==============================] - 11s 89ms/step - loss: 1.4347 - ...
    accuracy: 0.4573 - val_loss: 1.5954 - val_accuracy: 0.3833
Epoch 10/600
120/120 [==============================] - ETA: 0s - loss: 1.2836 - ...
    accuracy: 0.5216
```

```
Epoch 10: val_accuracy did not improve from 0.50521
120/120 [==============================] - 11s 90ms/step - loss: 1.2836 - ...
    accuracy: 0.5216 - val_loss: 1.3519 - val_accuracy: 0.4781
Epoch 11/600
120/120 [==============================] - ETA: 0s - loss: 1.1879 - ...
    accuracy: 0.5604
Epoch 11: val_accuracy did not improve from 0.50521
120/120 [==============================] - 11s 89ms/step - loss: 1.1879 - ...
    accuracy: 0.5604 - val_loss: 1.3798 - val_accuracy: 0.4708
Epoch 12/600
120/120 [==============================] - ETA: 0s - loss: 1.1725 - ...
    accuracy: 0.5607
Epoch 12: val_accuracy improved from 0.50521 to 0.52083, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 17s 146ms/step - loss: 1.1725 - ...
    accuracy: 0.5607 - val_loss: 1.2051 - val_accuracy: 0.5208
Epoch 13/600
120/120 [==============================] - ETA: 0s - loss: 1.1138 - ...
    accuracy: 0.5760
Epoch 13: val_accuracy improved from 0.52083 to 0.56667, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 132ms/step - loss: 1.1138 - ...
    accuracy: 0.5760 - val_loss: 1.1043 - val_accuracy: 0.5667
Epoch 14/600
120/120 [==============================] - ETA: 0s - loss: 1.0277 - ...
    accuracy: 0.6112
Epoch 14: val_accuracy did not improve from 0.56667
120/120 [==============================] - 11s 89ms/step - loss: 1.0277 - ...
    accuracy: 0.6112 - val_loss: 1.2350 - val_accuracy: 0.5479
Epoch 15/600
120/120 [==============================] - ETA: 0s - loss: 1.0072 - ...
    accuracy: 0.6107
Epoch 15: val_accuracy improved from 0.56667 to 0.58021, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 135ms/step - loss: 1.0072 - ...
    accuracy: 0.6107 - val_loss: 1.1223 - val_accuracy: 0.5802
Epoch 16/600
120/120 [==============================] - ETA: 0s - loss: 0.9585 - ...
    accuracy: 0.6383
Epoch 16: val_accuracy improved from 0.58021 to 0.58438, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 131ms/step - loss: 0.9585 - ...
    accuracy: 0.6383 - val_loss: 1.0353 - val_accuracy: 0.5844
Epoch 17/600
120/120 [==============================] - ETA: 0s - loss: 0.9058 - ...
    accuracy: 0.6578
Epoch 17: val_accuracy improved from 0.58438 to 0.59167, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 133ms/step - loss: 0.9058 - ...
    accuracy: 0.6578 - val_loss: 1.0484 - val_accuracy: 0.5917
Epoch 18/600
120/120 [==============================] - ETA: 0s - loss: 0.9502 - ...
    accuracy: 0.6453
Epoch 18: val_accuracy did not improve from 0.59167
120/120 [==============================] - 11s 89ms/step - loss: 0.9502 - ...
    accuracy: 0.6453 - val_loss: 1.0691 - val_accuracy: 0.5896
Epoch 19/600
120/120 [==============================] - ETA: 0s - loss: 0.8642 - ...
    accuracy: 0.6807
Epoch 19: val_accuracy improved from 0.59167 to 0.68125, saving model to ...
    Attentive_Conv_LSTM_model.tf
```

```
120/120 [==============================] - 16s 134ms/step - loss: 0.8642 - ...
    accuracy: 0.6807 - val_loss: 0.8231 - val_accuracy: 0.6812
Epoch 20/600
120/120 [==============================] - ETA: 0s - loss: 0.8882 - ...
    accuracy: 0.6742
Epoch 20: val_accuracy did not improve from 0.68125
120/120 [==============================] - 11s 89ms/step - loss: 0.8882 - ...
    accuracy: 0.6742 - val_loss: 0.9268 - val_accuracy: 0.6271
Epoch 21/600
120/120 [==============================] - ETA: 0s - loss: 0.7997 - ...
    accuracy: 0.6984
Epoch 21: val_accuracy did not improve from 0.68125
120/120 [==============================] - 11s 89ms/step - loss: 0.7997 - ...
    accuracy: 0.6984 - val_loss: 0.9724 - val_accuracy: 0.6229
Epoch 22/600
120/120 [==============================] - ETA: 0s - loss: 0.8205 - ...
    accuracy: 0.7073
Epoch 22: val_accuracy did not improve from 0.68125
120/120 [==============================] - 11s 91ms/step - loss: 0.8205 - ...
    accuracy: 0.7073 - val_loss: 0.9220 - val_accuracy: 0.6396
Epoch 23/600
120/120 [==============================] - ETA: 0s - loss: 0.7579 - ...
    accuracy: 0.7177
Epoch 23: val_accuracy did not improve from 0.68125
120/120 [==============================] - 11s 90ms/step - loss: 0.7579 - ...
    accuracy: 0.7177 - val_loss: 0.9164 - val_accuracy: 0.6302
Epoch 24/600
120/120 [==============================] - ETA: 0s - loss: 0.7446 - ...
    accuracy: 0.7279
Epoch 24: val_accuracy did not improve from 0.68125
120/120 [==============================] - 11s 90ms/step - loss: 0.7446 - ...
    accuracy: 0.7279 - val_loss: 0.9416 - val_accuracy: 0.6438
Epoch 25/600
120/120 [==============================] - ETA: 0s - loss: 0.7328 - ...
    accuracy: 0.7336
Epoch 25: val_accuracy improved from 0.68125 to 0.71667, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 132ms/step - loss: 0.7328 - ...
    accuracy: 0.7336 - val_loss: 0.7666 - val_accuracy: 0.7167
Epoch 26/600
120/120 [==============================] - ETA: 0s - loss: 0.7272 - ...
    accuracy: 0.7328
Epoch 26: val_accuracy did not improve from 0.71667
120/120 [==============================] - 11s 89ms/step - loss: 0.7272 - ...
    accuracy: 0.7328 - val_loss: 0.8258 - val_accuracy: 0.6917
Epoch 27/600
120/120 [==============================] - ETA: 0s - loss: 0.7175 - ...
    accuracy: 0.7417
Epoch 27: val_accuracy did not improve from 0.71667
120/120 [==============================] - 11s 90ms/step - loss: 0.7175 - ...
    accuracy: 0.7417 - val_loss: 0.7915 - val_accuracy: 0.7052
Epoch 28/600
120/120 [==============================] - ETA: 0s - loss: 0.6965 - ...
    accuracy: 0.7526
Epoch 28: val_accuracy did not improve from 0.71667
120/120 [==============================] - 11s 90ms/step - loss: 0.6965 - ...
    accuracy: 0.7526 - val_loss: 0.8679 - val_accuracy: 0.7052
Epoch 29/600
120/120 [==============================] - ETA: 0s - loss: 0.6940 - ...
    accuracy: 0.7513
Epoch 29: val_accuracy did not improve from 0.71667
```

```
120/120 [==============================] - 11s 89ms/step - loss: 0.6940 - ...
    accuracy: 0.7513 - val_loss: 0.9044 - val_accuracy: 0.6562
Epoch 30/600
120/120 [==============================] - ETA: 0s - loss: 0.6850 - ...
    accuracy: 0.7542
Epoch 30: val_accuracy improved from 0.71667 to 0.72604, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 18s 151ms/step - loss: 0.6850 - ...
    accuracy: 0.7542 - val_loss: 0.7911 - val_accuracy: 0.7260
Epoch 31/600
120/120 [==============================] - ETA: 0s - loss: 0.6706 - ...
    accuracy: 0.7646
Epoch 31: val_accuracy improved from 0.72604 to 0.72917, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 131ms/step - loss: 0.6706 - ...
    accuracy: 0.7646 - val_loss: 0.7142 - val_accuracy: 0.7292
Epoch 32/600
120/120 [==============================] - ETA: 0s - loss: 0.6125 - ...
    accuracy: 0.7896
Epoch 32: val_accuracy improved from 0.72917 to 0.74479, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 133ms/step - loss: 0.6125 - ...
    accuracy: 0.7896 - val_loss: 0.7434 - val_accuracy: 0.7448
Epoch 33/600
120/120 [==============================] - ETA: 0s - loss: 0.6099 - ...
    accuracy: 0.7862
Epoch 33: val_accuracy did not improve from 0.74479
120/120 [==============================] - 11s 89ms/step - loss: 0.6099 - ...
    accuracy: 0.7862 - val_loss: 0.7089 - val_accuracy: 0.7417
Epoch 34/600
120/120 [==============================] - ETA: 0s - loss: 0.6228 - ...
    accuracy: 0.7844
Epoch 34: val_accuracy improved from 0.74479 to 0.76875, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 133ms/step - loss: 0.6228 - ...
    accuracy: 0.7844 - val_loss: 0.6115 - val_accuracy: 0.7688

.............................


Epoch 38/600
120/120 [==============================] - ETA: 0s - loss: 0.5708 - ...
    accuracy: 0.8008
Epoch 38: val_accuracy did not improve from 0.76875
120/120 [==============================] - 11s 89ms/step - loss: 0.5708 - ...
    accuracy: 0.8008 - val_loss: 0.6611 - val_accuracy: 0.7656
Epoch 39/600
120/120 [==============================] - ETA: 0s - loss: 0.5398 - ...
    accuracy: 0.8188
Epoch 39: val_accuracy did not improve from 0.76875
120/120 [==============================] - 11s 90ms/step - loss: 0.5398 - ...
    accuracy: 0.8188 - val_loss: 0.8181 - val_accuracy: 0.7333
Epoch 40/600
120/120 [==============================] - ETA: 0s - loss: 0.5377 - ...
    accuracy: 0.8185
Epoch 40: val_accuracy improved from 0.76875 to 0.79167, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 132ms/step - loss: 0.5377 - ...
    accuracy: 0.8185 - val_loss: 0.5926 - val_accuracy: 0.7917
Epoch 41/600
120/120 [==============================] - ETA: 0s - loss: 0.5994 - ...
```

```
    accuracy: 0.7898
Epoch 41: val_accuracy improved from 0.79167 to 0.80000, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 132ms/step - loss: 0.5994 - ...
    accuracy: 0.7898 - val_loss: 0.5689 - val_accuracy: 0.8000
Epoch 42/600
120/120 [==============================] - ETA: 0s - loss: 0.5218 - ...
    accuracy: 0.8234
Epoch 42: val_accuracy improved from 0.80000 to 0.81563, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 132ms/step - loss: 0.5218 - ...
    accuracy: 0.8234 - val_loss: 0.5329 - val_accuracy: 0.8156
Epoch 43/600
120/120 [==============================] - ETA: 0s - loss: 0.5109 - ...
    accuracy: 0.8198
Epoch 43: val_accuracy did not improve from 0.81563
120/120 [==============================] - 11s 89ms/step - loss: 0.5109 - ...
    accuracy: 0.8198 - val_loss: 0.5707 - val_accuracy: 0.7937
Epoch 44/600
120/120 [==============================] - ETA: 0s - loss: 0.4969 - ...
    accuracy: 0.8266
Epoch 44: val_accuracy improved from 0.81563 to 0.83333, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 132ms/step - loss: 0.4969 - ...
    accuracy: 0.8266 - val_loss: 0.5017 - val_accuracy: 0.8333
Epoch 45/600
120/120 [==============================] - ETA: 0s - loss: 0.4991 - ...
    accuracy: 0.8266
Epoch 45: val_accuracy did not improve from 0.83333
120/120 [==============================] - 11s 89ms/step - loss: 0.4991 - ...
    accuracy: 0.8266 - val_loss: 0.5463 - val_accuracy: 0.8156
Epoch 46/600
120/120 [==============================] - ETA: 0s - loss: 0.4904 - ...
    accuracy: 0.8294
Epoch 46: val_accuracy improved from 0.83333 to 0.84062, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 18s 152ms/step - loss: 0.4904 - ...
    accuracy: 0.8294 - val_loss: 0.4702 - val_accuracy: 0.8406
Epoch 47/600
120/120 [==============================] - ETA: 0s - loss: 0.4508 - ...
    accuracy: 0.8440
Epoch 47: val_accuracy did not improve from 0.84062
120/120 [==============================] - 11s 89ms/step - loss: 0.4508 - ...
    accuracy: 0.8440 - val_loss: 0.5346 - val_accuracy: 0.8219
Epoch 48/600
120/120 [==============================] - ETA: 0s - loss: 0.4554 - ...
    accuracy: 0.8451
Epoch 48: val_accuracy improved from 0.84062 to 0.85833, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 132ms/step - loss: 0.4554 - ...
    accuracy: 0.8451 - val_loss: 0.4237 - val_accuracy: 0.8583
Epoch 49/600
120/120 [==============================] - ETA: 0s - loss: 0.4990 - ...
    accuracy: 0.8331
Epoch 49: val_accuracy did not improve from 0.85833
120/120 [==============================] - 11s 89ms/step - loss: 0.4990 - ...
    accuracy: 0.8331 - val_loss: 0.4978 - val_accuracy: 0.8073
Epoch 50/600
120/120 [==============================] - ETA: 0s - loss: 0.4315 - ...
    accuracy: 0.8521
Epoch 50: val_accuracy did not improve from 0.85833
```

```
120/120 [==============================] - 11s 90ms/step - loss: 0.4315 - ...
    accuracy: 0.8521 - val_loss: 0.4750 - val_accuracy: 0.8385
Epoch 51/600
120/120 [==============================] - ETA: 0s - loss: 0.4327 - ...
    accuracy: 0.8513
Epoch 51: val_accuracy did not improve from 0.85833
120/120 [==============================] - 11s 89ms/step - loss: 0.4327 - ...
    accuracy: 0.8513 - val_loss: 0.4797 - val_accuracy: 0.8323
Epoch 52/600
120/120 [==============================] - ETA: 0s - loss: 0.4045 - ...
    accuracy: 0.8633
Epoch 52: val_accuracy did not improve from 0.85833
120/120 [==============================] - 11s 89ms/step - loss: 0.4045 - ...
    accuracy: 0.8633 - val_loss: 0.4389 - val_accuracy: 0.8365
Epoch 53/600
120/120 [==============================] - ETA: 0s - loss: 0.4354 - ...
    accuracy: 0.8518
Epoch 53: val_accuracy did not improve from 0.85833
120/120 [==============================] - 11s 89ms/step - loss: 0.4354 - ...
    accuracy: 0.8518 - val_loss: 0.4075 - val_accuracy: 0.8562
Epoch 54/600
120/120 [==============================] - ETA: 0s - loss: 0.4224 - ...
    accuracy: 0.8521
Epoch 54: val_accuracy improved from 0.85833 to 0.87083, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 132ms/step - loss: 0.4224 - ...
    accuracy: 0.8521 - val_loss: 0.3763 - val_accuracy: 0.8708
Epoch 55/600
120/120 [==============================] - ETA: 0s - loss: 0.3864 - ...
    accuracy: 0.8633
Epoch 55: val_accuracy did not improve from 0.87083
120/120 [==============================] - 11s 90ms/step - loss: 0.3864 - ...
    accuracy: 0.8633 - val_loss: 0.4499 - val_accuracy: 0.8260

..............

Epoch 65/600
120/120 [==============================] - ETA: 0s - loss: 0.3493 - ...
    accuracy: 0.8766
Epoch 65: val_accuracy improved from 0.87604 to 0.87813, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 133ms/step - loss: 0.3493 - ...
    accuracy: 0.8766 - val_loss: 0.3613 - val_accuracy: 0.8781
Epoch 66/600
120/120 [==============================] - ETA: 0s - loss: 0.3214 - ...
    accuracy: 0.8901
Epoch 66: val_accuracy improved from 0.87813 to 0.90938, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 131ms/step - loss: 0.3214 - ...
    accuracy: 0.8901 - val_loss: 0.2872 - val_accuracy: 0.9094

............


Epoch 78/600
120/120 [==============================] - ETA: 0s - loss: 0.3133 - ...
    accuracy: 0.8938
Epoch 78: val_accuracy did not improve from 0.90938
120/120 [==============================] - 11s 89ms/step - loss: 0.3133 - ...
    accuracy: 0.8938 - val_loss: 0.3126 - val_accuracy: 0.8823
Epoch 79/600
```

```
120/120 [==============================] - ETA: 0s - loss: 0.3130 - ...
    accuracy: 0.8919
Epoch 79: val_accuracy did not improve from 0.90938
120/120 [==============================] - 11s 89ms/step - loss: 0.3130 - ...
    accuracy: 0.8919 - val_loss: 0.3091 - val_accuracy: 0.8927
Epoch 80/600
120/120 [==============================] - ETA: 0s - loss: 0.3414 - ...
    accuracy: 0.8828
Epoch 80: val_accuracy did not improve from 0.90938
120/120 [==============================] - 11s 90ms/step - loss: 0.3414 - ...
    accuracy: 0.8828 - val_loss: 0.3246 - val_accuracy: 0.8802
Epoch 81/600
120/120 [==============================] - ETA: 0s - loss: 0.3285 - ...
    accuracy: 0.8854
Epoch 81: val_accuracy did not improve from 0.90938
120/120 [==============================] - 11s 90ms/step - loss: 0.3285 - ...
    accuracy: 0.8854 - val_loss: 0.3205 - val_accuracy: 0.9042
Epoch 82/600
120/120 [==============================] - ETA: 0s - loss: 0.3465 - ...
    accuracy: 0.8818
Epoch 82: val_accuracy did not improve from 0.90938
120/120 [==============================] - 11s 89ms/step - loss: 0.3465 - ...
    accuracy: 0.8818 - val_loss: 0.3354 - val_accuracy: 0.8969
Epoch 83/600
120/120 [==============================] - ETA: 0s - loss: 0.3218 - ...
    accuracy: 0.8878
Epoch 83: val_accuracy did not improve from 0.90938
120/120 [==============================] - 11s 89ms/step - loss: 0.3218 - ...
    accuracy: 0.8878 - val_loss: 0.3553 - val_accuracy: 0.8844
Epoch 84/600
120/120 [==============================] - ETA: 0s - loss: 0.2907 - ...
    accuracy: 0.8979
Epoch 84: val_accuracy did not improve from 0.90938
120/120 [==============================] - 11s 90ms/step - loss: 0.2907 - ...
    accuracy: 0.8979 - val_loss: 0.3613 - val_accuracy: 0.8760
Epoch 85/600
120/120 [==============================] - ETA: 0s - loss: 0.2727 - ...
    accuracy: 0.9073
Epoch 85: val_accuracy did not improve from 0.90938
120/120 [==============================] - 11s 88ms/step - loss: 0.2727 - ...
    accuracy: 0.9073 - val_loss: 0.3151 - val_accuracy: 0.8885
Epoch 86/600
120/120 [==============================] - ETA: 0s - loss: 0.2872 - ...
    accuracy: 0.9018
Epoch 86: val_accuracy did not improve from 0.90938
120/120 [==============================] - 11s 89ms/step - loss: 0.2872 - ...
    accuracy: 0.9018 - val_loss: 0.3199 - val_accuracy: 0.8760
Epoch 87/600
120/120 [==============================] - ETA: 0s - loss: 0.2911 - ...
    accuracy: 0.9013
Epoch 87: val_accuracy did not improve from 0.90938
120/120 [==============================] - 11s 90ms/step - loss: 0.2911 - ...
    accuracy: 0.9013 - val_loss: 0.2726 - val_accuracy: 0.8990
Epoch 88/600
120/120 [==============================] - ETA: 0s - loss: 0.2651 - ...
    accuracy: 0.9089
Epoch 88: val_accuracy did not improve from 0.90938
120/120 [==============================] - 11s 89ms/step - loss: 0.2651 - ...
    accuracy: 0.9089 - val_loss: 0.2833 - val_accuracy: 0.9062
Epoch 89/600
120/120 [==============================] - ETA: 0s - loss: 0.2803 - ...
```

```
    accuracy: 0.9047
Epoch 89: val_accuracy did not improve from 0.90938
120/120 [==============================] - 11s 89ms/step - loss: 0.2803 - ...
    accuracy: 0.9047 - val_loss: 0.3053 - val_accuracy: 0.8969
Epoch 90/600
120/120 [==============================] - ETA: 0s - loss: 0.2542 - ...
    accuracy: 0.9146
Epoch 90: val_accuracy did not improve from 0.90938
120/120 [==============================] - 11s 90ms/step - loss: 0.2542 - ...
    accuracy: 0.9146 - val_loss: 0.3752 - val_accuracy: 0.8813
Epoch 91/600
120/120 [==============================] - ETA: 0s - loss: 0.2971 - ...
    accuracy: 0.8945
Epoch 91: val_accuracy improved from 0.90938 to 0.91146, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 132ms/step - loss: 0.2971 - ...
    accuracy: 0.8945 - val_loss: 0.2685 - val_accuracy: 0.9115
Epoch 92/600
120/120 [==============================] - ETA: 0s - loss: 0.3627 - ...
    accuracy: 0.8781
Epoch 92: val_accuracy did not improve from 0.91146
120/120 [==============================] - 11s 88ms/step - loss: 0.3627 - ...
    accuracy: 0.8781 - val_loss: 0.4150 - val_accuracy: 0.8573
Epoch 93/600
120/120 [==============================] - ETA: 0s - loss: 0.3256 - ...
    accuracy: 0.8852
Epoch 93: val_accuracy did not improve from 0.91146
120/120 [==============================] - 11s 90ms/step - loss: 0.3256 - ...
    accuracy: 0.8852 - val_loss: 0.3008 - val_accuracy: 0.9042
Epoch 94/600
120/120 [==============================] - ETA: 0s - loss: 0.2619 - ...
    accuracy: 0.9102
Epoch 94: val_accuracy improved from 0.91146 to 0.91250, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 131ms/step - loss: 0.2619 - ...
    accuracy: 0.9102 - val_loss: 0.2682 - val_accuracy: 0.9125
Epoch 95/600
120/120 [==============================] - ETA: 0s - loss: 0.2636 - ...
    accuracy: 0.9120
Epoch 95: val_accuracy did not improve from 0.91250
120/120 [==============================] - 11s 91ms/step - loss: 0.2636 - ...
    accuracy: 0.9120 - val_loss: 0.3011 - val_accuracy: 0.8938
Epoch 96/600
120/120 [==============================] - ETA: 0s - loss: 0.2570 - ...
    accuracy: 0.9122
Epoch 96: val_accuracy did not improve from 0.91250
120/120 [==============================] - 11s 89ms/step - loss: 0.2570 - ...
    accuracy: 0.9122 - val_loss: 0.3045 - val_accuracy: 0.9052
Epoch 97/600
120/120 [==============================] - ETA: 0s - loss: 0.2608 - ...
    accuracy: 0.9104
Epoch 97: val_accuracy did not improve from 0.91250
120/120 [==============================] - 11s 89ms/step - loss: 0.2608 - ...
    accuracy: 0.9104 - val_loss: 0.3020 - val_accuracy: 0.8958
Epoch 98/600
120/120 [==============================] - ETA: 0s - loss: 0.2754 - ...
    accuracy: 0.9039
Epoch 98: val_accuracy did not improve from 0.91250
120/120 [==============================] - 11s 91ms/step - loss: 0.2754 - ...
    accuracy: 0.9039 - val_loss: 0.4661 - val_accuracy: 0.8562
Epoch 99/600
```

```
120/120 [==============================] - ETA: 0s - loss: 0.2779 - ...
    accuracy: 0.9049
Epoch 99: val_accuracy did not improve from 0.91250
120/120 [==============================] - 11s 88ms/step - loss: 0.2779 - ...
    accuracy: 0.9049 - val_loss: 0.2850 - val_accuracy: 0.9042
Epoch 100/600
120/120 [==============================] - ETA: 0s - loss: 0.2639 - ...
    accuracy: 0.9148
Epoch 100: val_accuracy did not improve from 0.91250
120/120 [==============================] - 11s 89ms/step - loss: 0.2639 - ...
    accuracy: 0.9148 - val_loss: 0.2505 - val_accuracy: 0.9125
Epoch 101/600
120/120 [==============================] - ETA: 0s - loss: 0.2629 - ...
    accuracy: 0.9091
Epoch 101: val_accuracy did not improve from 0.91250
120/120 [==============================] - 11s 91ms/step - loss: 0.2629 - ...
    accuracy: 0.9091 - val_loss: 0.3078 - val_accuracy: 0.8948
Epoch 102/600
120/120 [==============================] - ETA: 0s - loss: 0.2750 - ...
    accuracy: 0.9065
Epoch 102: val_accuracy did not improve from 0.91250
120/120 [==============================] - 11s 90ms/step - loss: 0.2750 - ...
    accuracy: 0.9065 - val_loss: 0.2590 - val_accuracy: 0.9062
Epoch 103/600
120/120 [==============================] - ETA: 0s - loss: 0.2611 - ...
    accuracy: 0.9169
Epoch 103: val_accuracy did not improve from 0.91250
120/120 [==============================] - 11s 89ms/step - loss: 0.2611 - ...
    accuracy: 0.9169 - val_loss: 0.3057 - val_accuracy: 0.8948
Epoch 104/600
120/120 [==============================] - ETA: 0s - loss: 0.2959 - ...
    accuracy: 0.9049
Epoch 104: val_accuracy did not improve from 0.91250
120/120 [==============================] - 11s 91ms/step - loss: 0.2959 - ...
    accuracy: 0.9049 - val_loss: 0.2715 - val_accuracy: 0.9083
Epoch 105/600
120/120 [==============================] - ETA: 0s - loss: 0.2290 - ...
    accuracy: 0.9208
Epoch 105: val_accuracy did not improve from 0.91250
120/120 [==============================] - 11s 89ms/step - loss: 0.2290 - ...
    accuracy: 0.9208 - val_loss: 0.3056 - val_accuracy: 0.8927
Epoch 106/600
120/120 [==============================] - ETA: 0s - loss: 0.2387 - ...
    accuracy: 0.9206
Epoch 106: val_accuracy did not improve from 0.91250
120/120 [==============================] - 11s 88ms/step - loss: 0.2387 - ...
    accuracy: 0.9206 - val_loss: 0.2699 - val_accuracy: 0.9010
Epoch 107/600
120/120 [==============================] - ETA: 0s - loss: 0.3010 - ...
    accuracy: 0.8951
Epoch 107: val_accuracy did not improve from 0.91250
120/120 [==============================] - 11s 89ms/step - loss: 0.3010 - ...
    accuracy: 0.8951 - val_loss: 0.3998 - val_accuracy: 0.8677
Epoch 108/600
120/120 [==============================] - ETA: 0s - loss: 0.3218 - ...
    accuracy: 0.8922
Epoch 108: val_accuracy did not improve from 0.91250
120/120 [==============================] - 11s 89ms/step - loss: 0.3218 - ...
    accuracy: 0.8922 - val_loss: 0.2739 - val_accuracy: 0.8990
Epoch 109/600
120/120 [==============================] - ETA: 0s - loss: 0.3399 - ...
```

```
    accuracy: 0.8820
Epoch 109: val_accuracy improved from 0.91250 to 0.92917, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 133ms/step - loss: 0.3399 - ...
    accuracy: 0.8820 - val_loss: 0.2504 - val_accuracy: 0.9292
Epoch 110/600
120/120 [==============================] - ETA: 0s - loss: 0.2495 - ...
    accuracy: 0.9104
Epoch 110: val_accuracy did not improve from 0.92917
120/120 [==============================] - 11s 89ms/step - loss: 0.2495 - ...
    accuracy: 0.9104 - val_loss: 0.2672 - val_accuracy: 0.9083
Epoch 111/600
120/120 [==============================] - ETA: 0s - loss: 0.2688 - ...
    accuracy: 0.9034
Epoch 111: val_accuracy did not improve from 0.92917
120/120 [==============================] - 11s 89ms/step - loss: 0.2688 - ...
    accuracy: 0.9034 - val_loss: 0.2520 - val_accuracy: 0.9208
Epoch 112/600
120/120 [==============================] - ETA: 0s - loss: 0.2390 - ...
    accuracy: 0.9161
Epoch 112: val_accuracy did not improve from 0.92917
120/120 [==============================] - 11s 91ms/step - loss: 0.2390 - ...
    accuracy: 0.9161 - val_loss: 0.2266 - val_accuracy: 0.9135
Epoch 113/600
120/120 [==============================] - ETA: 0s - loss: 0.2451 - ...
    accuracy: 0.9180
Epoch 113: val_accuracy did not improve from 0.92917
120/120 [==============================] - 11s 88ms/step - loss: 0.2451 - ...
    accuracy: 0.9180 - val_loss: 0.2947 - val_accuracy: 0.8833
Epoch 114/600
120/120 [==============================] - ETA: 0s - loss: 0.2131 - ...
    accuracy: 0.9284
Epoch 114: val_accuracy did not improve from 0.92917
120/120 [==============================] - 11s 89ms/step - loss: 0.2131 - ...
    accuracy: 0.9284 - val_loss: 0.2967 - val_accuracy: 0.8990
Epoch 115/600
120/120 [==============================] - ETA: 0s - loss: 0.2119 - ...
    accuracy: 0.9260
Epoch 115: val_accuracy did not improve from 0.92917
120/120 [==============================] - 11s 90ms/step - loss: 0.2119 - ...
    accuracy: 0.9260 - val_loss: 0.2437 - val_accuracy: 0.9146
Epoch 116/600
120/120 [==============================] - ETA: 0s - loss: 0.1984 - ...
    accuracy: 0.9297
Epoch 116: val_accuracy did not improve from 0.92917
120/120 [==============================] - 11s 89ms/step - loss: 0.1984 - ...
    accuracy: 0.9297 - val_loss: 0.2457 - val_accuracy: 0.9177
Epoch 117/600
120/120 [==============================] - ETA: 0s - loss: 0.2678 - ...
    accuracy: 0.9073
Epoch 117: val_accuracy did not improve from 0.92917
120/120 [==============================] - 11s 89ms/step - loss: 0.2678 - ...
    accuracy: 0.9073 - val_loss: 0.2683 - val_accuracy: 0.9073
Epoch 118/600
120/120 [==============================] - ETA: 0s - loss: 0.2614 - ...
    accuracy: 0.9086
Epoch 118: val_accuracy did not improve from 0.92917
120/120 [==============================] - 11s 91ms/step - loss: 0.2614 - ...
    accuracy: 0.9086 - val_loss: 0.2826 - val_accuracy: 0.9083
Epoch 119/600
120/120 [==============================] - ETA: 0s - loss: 0.2321 - ...
```

```
    accuracy: 0.9242
Epoch 119: val_accuracy did not improve from 0.92917
120/120 [==============================] - 11s 90ms/step - loss: 0.2321 - ...
    accuracy: 0.9242 - val_loss: 0.2361 - val_accuracy: 0.9125
Epoch 120/600
120/120 [==============================] - ETA: 0s - loss: 0.2247 - ...
    accuracy: 0.9263
Epoch 120: val_accuracy did not improve from 0.92917
120/120 [==============================] - 11s 89ms/step - loss: 0.2247 - ...
    accuracy: 0.9263 - val_loss: 0.2993 - val_accuracy: 0.8906
Epoch 121/600
120/120 [==============================] - ETA: 0s - loss: 0.2359 - ...
    accuracy: 0.9151
Epoch 121: val_accuracy did not improve from 0.92917
120/120 [==============================] - 11s 90ms/step - loss: 0.2359 - ...
    accuracy: 0.9151 - val_loss: 0.2798 - val_accuracy: 0.9052
Epoch 122/600
120/120 [==============================] - ETA: 0s - loss: 0.2237 - ...
    accuracy: 0.9242
Epoch 122: val_accuracy did not improve from 0.92917
120/120 [==============================] - 11s 89ms/step - loss: 0.2237 - ...
    accuracy: 0.9242 - val_loss: 0.3166 - val_accuracy: 0.9021
Epoch 123/600
120/120 [==============================] - ETA: 0s - loss: 0.2593 - ...
    accuracy: 0.9141
Epoch 123: val_accuracy did not improve from 0.92917
120/120 [==============================] - 11s 90ms/step - loss: 0.2593 - ...
    accuracy: 0.9141 - val_loss: 0.2841 - val_accuracy: 0.8990
Epoch 124/600
120/120 [==============================] - ETA: 0s - loss: 0.2966 - ...
    accuracy: 0.8932
Epoch 124: val_accuracy did not improve from 0.92917
120/120 [==============================] - 11s 91ms/step - loss: 0.2966 - ...
    accuracy: 0.8932 - val_loss: 0.2267 - val_accuracy: 0.9115
Epoch 125/600
120/120 [==============================] - ETA: 0s - loss: 0.2480 - ...
    accuracy: 0.9104
Epoch 125: val_accuracy did not improve from 0.92917
120/120 [==============================] - 11s 89ms/step - loss: 0.2480 - ...
    accuracy: 0.9104 - val_loss: 0.2332 - val_accuracy: 0.9156
Epoch 126/600
120/120 [==============================] - ETA: 0s - loss: 0.2738 - ...
    accuracy: 0.9060
Epoch 126: val_accuracy did not improve from 0.92917
120/120 [==============================] - 11s 90ms/step - loss: 0.2738 - ...
    accuracy: 0.9060 - val_loss: 0.2135 - val_accuracy: 0.9240
Epoch 127/600
120/120 [==============================] - ETA: 0s - loss: 0.2838 - ...
    accuracy: 0.9016
Epoch 127: val_accuracy did not improve from 0.92917
120/120 [==============================] - 11s 90ms/step - loss: 0.2838 - ...
    accuracy: 0.9016 - val_loss: 0.3049 - val_accuracy: 0.9083
Epoch 128/600
120/120 [==============================] - ETA: 0s - loss: 0.2361 - ...
    accuracy: 0.9201
Epoch 128: val_accuracy did not improve from 0.92917
120/120 [==============================] - 11s 89ms/step - loss: 0.2361 - ...
    accuracy: 0.9201 - val_loss: 0.2268 - val_accuracy: 0.9187
Epoch 129/600
120/120 [==============================] - ETA: 0s - loss: 0.2467 - ...
    accuracy: 0.9125
```

```
Epoch 129: val_accuracy improved from 0.92917 to 0.93125, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 132ms/step - loss: 0.2467 - ...
    accuracy: 0.9125 - val_loss: 0.2030 - val_accuracy: 0.9312
Epoch 130/600
120/120 [==============================] - ETA: 0s - loss: 0.2108 - ...
    accuracy: 0.9307
Epoch 130: val_accuracy did not improve from 0.93125
120/120 [==============================] - 11s 90ms/step - loss: 0.2108 - ...
    accuracy: 0.9307 - val_loss: 0.2579 - val_accuracy: 0.9115
Epoch 131/600
120/120 [==============================] - ETA: 0s - loss: 0.2053 - ...
    accuracy: 0.9294
Epoch 131: val_accuracy did not improve from 0.93125
120/120 [==============================] - 11s 89ms/step - loss: 0.2053 - ...
    accuracy: 0.9294 - val_loss: 0.2176 - val_accuracy: 0.9260
Epoch 132/600
120/120 [==============================] - ETA: 0s - loss: 0.2187 - ...
    accuracy: 0.9245
Epoch 132: val_accuracy did not improve from 0.93125
120/120 [==============================] - 11s 90ms/step - loss: 0.2187 - ...
    accuracy: 0.9245 - val_loss: 0.2051 - val_accuracy: 0.9240
Epoch 133/600
120/120 [==============================] - ETA: 0s - loss: 0.2095 - ...
    accuracy: 0.9255
Epoch 133: val_accuracy improved from 0.93125 to 0.93333, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 18s 153ms/step - loss: 0.2095 - ...
    accuracy: 0.9255 - val_loss: 0.1951 - val_accuracy: 0.9333
Epoch 134/600
120/120 [==============================] - ETA: 0s - loss: 0.2440 - ...
    accuracy: 0.9172
Epoch 134: val_accuracy did not improve from 0.93333
120/120 [==============================] - 11s 91ms/step - loss: 0.2440 - ...
    accuracy: 0.9172 - val_loss: 0.3499 - val_accuracy: 0.8802
Epoch 135/600
120/120 [==============================] - ETA: 0s - loss: 0.2423 - ...
    accuracy: 0.9109
Epoch 135: val_accuracy did not improve from 0.93333
120/120 [==============================] - 11s 88ms/step - loss: 0.2423 - ...
    accuracy: 0.9109 - val_loss: 0.2295 - val_accuracy: 0.9187
Epoch 136/600
120/120 [==============================] - ETA: 0s - loss: 0.2329 - ...
    accuracy: 0.9177
Epoch 136: val_accuracy did not improve from 0.93333
120/120 [==============================] - 11s 88ms/step - loss: 0.2329 - ...
    accuracy: 0.9177 - val_loss: 0.2427 - val_accuracy: 0.9125
Epoch 137/600
120/120 [==============================] - ETA: 0s - loss: 0.2383 - ...
    accuracy: 0.9182
Epoch 137: val_accuracy improved from 0.93333 to 0.93437, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 132ms/step - loss: 0.2383 - ...
    accuracy: 0.9182 - val_loss: 0.2403 - val_accuracy: 0.9344
Epoch 138/600
120/120 [==============================] - ETA: 0s - loss: 0.2219 - ...
    accuracy: 0.9237
Epoch 138: val_accuracy did not improve from 0.93437
120/120 [==============================] - 11s 89ms/step - loss: 0.2219 - ...
    accuracy: 0.9237 - val_loss: 0.2204 - val_accuracy: 0.9208
Epoch 139/600
```

```
120/120 [==============================] - ETA: 0s - loss: 0.2292 - ...
    accuracy: 0.9214
Epoch 139: val_accuracy did not improve from 0.93437
120/120 [==============================] - 11s 90ms/step - loss: 0.2292 - ...
    accuracy: 0.9214 - val_loss: 0.2556 - val_accuracy: 0.9042
Epoch 140/600
120/120 [==============================] - ETA: 0s - loss: 0.2458 - ...
    accuracy: 0.9174
Epoch 140: val_accuracy did not improve from 0.93437
120/120 [==============================] - 11s 90ms/step - loss: 0.2458 - ...
    accuracy: 0.9174 - val_loss: 0.2310 - val_accuracy: 0.9229
Epoch 141/600
120/120 [==============================] - ETA: 0s - loss: 0.2378 - ...
    accuracy: 0.9187
Epoch 141: val_accuracy did not improve from 0.93437
120/120 [==============================] - 11s 88ms/step - loss: 0.2378 - ...
    accuracy: 0.9187 - val_loss: 0.2161 - val_accuracy: 0.9302
Epoch 142/600
120/120 [==============================] - ETA: 0s - loss: 0.2300 - ...
    accuracy: 0.9177
Epoch 142: val_accuracy did not improve from 0.93437
120/120 [==============================] - 11s 89ms/step - loss: 0.2300 - ...
    accuracy: 0.9177 - val_loss: 0.2239 - val_accuracy: 0.9260

..............

Epoch 166/600
120/120 [==============================] - ETA: 0s - loss: 0.2621 - ...
    accuracy: 0.9070
Epoch 166: val_accuracy did not improve from 0.93437
120/120 [==============================] - 11s 90ms/step - loss: 0.2621 - ...
    accuracy: 0.9070 - val_loss: 0.2013 - val_accuracy: 0.9281
Epoch 167/600
120/120 [==============================] - ETA: 0s - loss: 0.2070 - ...
    accuracy: 0.9255
Epoch 167: val_accuracy did not improve from 0.93437
120/120 [==============================] - 11s 89ms/step - loss: 0.2070 - ...
    accuracy: 0.9255 - val_loss: 0.2256 - val_accuracy: 0.9146
Epoch 168/600
120/120 [==============================] - ETA: 0s - loss: 0.2003 - ...
    accuracy: 0.9289
Epoch 168: val_accuracy did not improve from 0.93437
120/120 [==============================] - 11s 89ms/step - loss: 0.2003 - ...
    accuracy: 0.9289 - val_loss: 0.2276 - val_accuracy: 0.9281
Epoch 169/600
120/120 [==============================] - ETA: 0s - loss: 0.2089 - ...
    accuracy: 0.9276
Epoch 169: val_accuracy did not improve from 0.93437
120/120 [==============================] - 11s 89ms/step - loss: 0.2089 - ...
    accuracy: 0.9276 - val_loss: 0.2464 - val_accuracy: 0.9010
Epoch 170/600
120/120 [==============================] - ETA: 0s - loss: 0.1913 - ...
    accuracy: 0.9328
Epoch 170: val_accuracy did not improve from 0.93437
120/120 [==============================] - 11s 89ms/step - loss: 0.1913 - ...
    accuracy: 0.9328 - val_loss: 0.3242 - val_accuracy: 0.8917
Epoch 171/600
120/120 [==============================] - ETA: 0s - loss: 0.2894 - ...
    accuracy: 0.8969
Epoch 171: val_accuracy did not improve from 0.93437
120/120 [==============================] - 11s 89ms/step - loss: 0.2894 - ...
```

```
    accuracy: 0.8969 - val_loss: 0.2556 - val_accuracy: 0.9031
Epoch 172/600
120/120 [==============================] - ETA: 0s - loss: 0.2342 - ...
    accuracy: 0.9174
Epoch 172: val_accuracy did not improve from 0.93437
120/120 [==============================] - 11s 89ms/step - loss: 0.2342 - ...
    accuracy: 0.9174 - val_loss: 0.1988 - val_accuracy: 0.9240
Epoch 173/600
120/120 [==============================] - ETA: 0s - loss: 0.2088 - ...
    accuracy: 0.9240
Epoch 173: val_accuracy did not improve from 0.93437
120/120 [==============================] - 11s 90ms/step - loss: 0.2088 - ...
    accuracy: 0.9240 - val_loss: 0.2075 - val_accuracy: 0.9302
Epoch 174/600
120/120 [==============================] - ETA: 0s - loss: 0.1860 - ...
    accuracy: 0.9331
Epoch 174: val_accuracy improved from 0.93437 to 0.94063, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 131ms/step - loss: 0.1860 - ...
    accuracy: 0.9331 - val_loss: 0.1763 - val_accuracy: 0.9406
Epoch 175/600
120/120 [==============================] - ETA: 0s - loss: 0.1997 - ...
    accuracy: 0.9297
Epoch 175: val_accuracy improved from 0.94063 to 0.95208, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 133ms/step - loss: 0.1997 - ...
    accuracy: 0.9297 - val_loss: 0.1557 - val_accuracy: 0.9521
Epoch 176/600
120/120 [==============================] - ETA: 0s - loss: 0.1957 - ...
    accuracy: 0.9276
Epoch 176: val_accuracy did not improve from 0.95208
120/120 [==============================] - 11s 88ms/step - loss: 0.1957 - ...
    accuracy: 0.9276 - val_loss: 0.2370 - val_accuracy: 0.9260
Epoch 177/600
120/120 [==============================] - ETA: 0s - loss: 0.1880 - ...
    accuracy: 0.9310
Epoch 177: val_accuracy did not improve from 0.95208
120/120 [==============================] - 11s 90ms/step - loss: 0.1880 - ...
    accuracy: 0.9310 - val_loss: 0.1907 - val_accuracy: 0.9292
Epoch 178/600
120/120 [==============================] - ETA: 0s - loss: 0.1775 - ...
    accuracy: 0.9398
Epoch 178: val_accuracy did not improve from 0.95208
120/120 [==============================] - 11s 89ms/step - loss: 0.1775 - ...
    accuracy: 0.9398 - val_loss: 0.2136 - val_accuracy: 0.9281
Epoch 179/600
120/120 [==============================] - ETA: 0s - loss: 0.1662 - ...
    accuracy: 0.9438
Epoch 179: val_accuracy did not improve from 0.95208
120/120 [==============================] - 11s 88ms/step - loss: 0.1662 - ...
    accuracy: 0.9438 - val_loss: 0.1920 - val_accuracy: 0.9344
Epoch 180/600
120/120 [==============================] - ETA: 0s - loss: 0.1628 - ...
    accuracy: 0.9432
Epoch 180: val_accuracy did not improve from 0.95208
120/120 [==============================] - 11s 90ms/step - loss: 0.1628 - ...
    accuracy: 0.9432 - val_loss: 0.1928 - val_accuracy: 0.9292
Epoch 181/600
120/120 [==============================] - ETA: 0s - loss: 0.1827 - ...
    accuracy: 0.9365
Epoch 181: val_accuracy did not improve from 0.95208
```

```
120/120 [==============================] - 11s 89ms/step - loss: 0.1827 - ...
    accuracy: 0.9365 - val_loss: 0.1807 - val_accuracy: 0.9406


.........


Epoch 190/600
120/120 [==============================] - ETA: 0s - loss: 0.1964 - ...
    accuracy: 0.9333
Epoch 190: val_accuracy did not improve from 0.95208
120/120 [==============================] - 11s 89ms/step - loss: 0.1964 - ...
    accuracy: 0.9333 - val_loss: 0.3252 - val_accuracy: 0.9042
Epoch 191/600
120/120 [==============================] - ETA: 0s - loss: 0.2103 - ...
    accuracy: 0.9307
Epoch 191: val_accuracy did not improve from 0.95208
120/120 [==============================] - 11s 89ms/step - loss: 0.2103 - ...
    accuracy: 0.9307 - val_loss: 0.2768 - val_accuracy: 0.9125
Epoch 192/600
120/120 [==============================] - ETA: 0s - loss: 0.2188 - ...
    accuracy: 0.9232
Epoch 192: val_accuracy did not improve from 0.95208
120/120 [==============================] - 11s 89ms/step - loss: 0.2188 - ...
    accuracy: 0.9232 - val_loss: 0.2038 - val_accuracy: 0.9229
Epoch 193/600
120/120 [==============================] - ETA: 0s - loss: 0.1953 - ...
    accuracy: 0.9289
Epoch 193: val_accuracy did not improve from 0.95208
120/120 [==============================] - 11s 90ms/step - loss: 0.1953 - ...
    accuracy: 0.9289 - val_loss: 0.2201 - val_accuracy: 0.9250
Epoch 194/600
120/120 [==============================] - ETA: 0s - loss: 0.2138 - ...
    accuracy: 0.9273
Epoch 194: val_accuracy did not improve from 0.95208
120/120 [==============================] - 11s 89ms/step - loss: 0.2138 - ...
    accuracy: 0.9273 - val_loss: 0.2286 - val_accuracy: 0.9156
Epoch 195/600
120/120 [==============================] - ETA: 0s - loss: 0.2252 - ...
    accuracy: 0.9240
Epoch 195: val_accuracy did not improve from 0.95208
120/120 [==============================] - 11s 90ms/step - loss: 0.2252 - ...
    accuracy: 0.9240 - val_loss: 0.2023 - val_accuracy: 0.9344
Epoch 196/600
120/120 [==============================] - ETA: 0s - loss: 0.1861 - ...
    accuracy: 0.9315
Epoch 196: val_accuracy did not improve from 0.95208
120/120 [==============================] - 11s 88ms/step - loss: 0.1861 - ...
    accuracy: 0.9315 - val_loss: 0.1890 - val_accuracy: 0.9344
Epoch 197/600
120/120 [==============================] - ETA: 0s - loss: 0.2129 - ...
    accuracy: 0.9292
Epoch 197: val_accuracy did not improve from 0.95208
120/120 [==============================] - 11s 90ms/step - loss: 0.2129 - ...
    accuracy: 0.9292 - val_loss: 0.2647 - val_accuracy: 0.9073
Epoch 198/600
120/120 [==============================] - ETA: 0s - loss: 0.2337 - ...
    accuracy: 0.9151
Epoch 198: val_accuracy did not improve from 0.95208
120/120 [==============================] - 11s 89ms/step - loss: 0.2337 - ...
    accuracy: 0.9151 - val_loss: 0.1931 - val_accuracy: 0.9417
```

```
Epoch 199/600
120/120 [==============================] - ETA: 0s - loss: 0.1887 - ...
    accuracy: 0.9354
Epoch 199: val_accuracy did not improve from 0.95208
120/120 [==============================] - 11s 88ms/step - loss: 0.1887 - ...
    accuracy: 0.9354 - val_loss: 0.2118 - val_accuracy: 0.9396
Epoch 200/600
120/120 [==============================] - ETA: 0s - loss: 0.2078 - ...
    accuracy: 0.9305
Epoch 200: val_accuracy did not improve from 0.95208
120/120 [==============================] - 11s 90ms/step - loss: 0.2078 - ...
    accuracy: 0.9305 - val_loss: 0.1911 - val_accuracy: 0.9438
Epoch 201/600
120/120 [==============================] - ETA: 0s - loss: 0.1852 - ...
    accuracy: 0.9372
Epoch 201: val_accuracy improved from 0.95208 to 0.95417, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 130ms/step - loss: 0.1852 - ...
    accuracy: 0.9372 - val_loss: 0.1319 - val_accuracy: 0.9542
Epoch 202/600
120/120 [==============================] - ETA: 0s - loss: 0.1542 - ...
    accuracy: 0.9427
Epoch 202: val_accuracy did not improve from 0.95417
120/120 [==============================] - 11s 89ms/step - loss: 0.1542 - ...
    accuracy: 0.9427 - val_loss: 0.1706 - val_accuracy: 0.9500
Epoch 203/600
120/120 [==============================] - ETA: 0s - loss: 0.2007 - ...
    accuracy: 0.9307
Epoch 203: val_accuracy did not improve from 0.95417
120/120 [==============================] - 11s 90ms/step - loss: 0.2007 - ...
    accuracy: 0.9307 - val_loss: 0.2242 - val_accuracy: 0.9167
Epoch 204/600
120/120 [==============================] - ETA: 0s - loss: 0.2027 - ...
    accuracy: 0.9258
Epoch 204: val_accuracy did not improve from 0.95417
120/120 [==============================] - 11s 88ms/step - loss: 0.2027 - ...
    accuracy: 0.9258 - val_loss: 0.2091 - val_accuracy: 0.9229
Epoch 205/600
120/120 [==============================] - ETA: 0s - loss: 0.1884 - ...
    accuracy: 0.9315
Epoch 205: val_accuracy did not improve from 0.95417
120/120 [==============================] - 11s 89ms/step - loss: 0.1884 - ...
    accuracy: 0.9315 - val_loss: 0.2769 - val_accuracy: 0.8990
Epoch 206/600
120/120 [==============================] - ETA: 0s - loss: 0.2486 - ...
    accuracy: 0.9115
Epoch 206: val_accuracy did not improve from 0.95417
120/120 [==============================] - 11s 90ms/step - loss: 0.2486 - ...
    accuracy: 0.9115 - val_loss: 0.2015 - val_accuracy: 0.9281

.................

Epoch 294/600
120/120 [==============================] - ETA: 0s - loss: 0.1532 - ...
    accuracy: 0.9445
Epoch 294: val_accuracy did not improve from 0.96771
120/120 [==============================] - 11s 89ms/step - loss: 0.1532 - ...
    accuracy: 0.9445 - val_loss: 0.1396 - val_accuracy: 0.9417
Epoch 295/600
120/120 [==============================] - ETA: 0s - loss: 0.1664 - ...
    accuracy: 0.9391
```

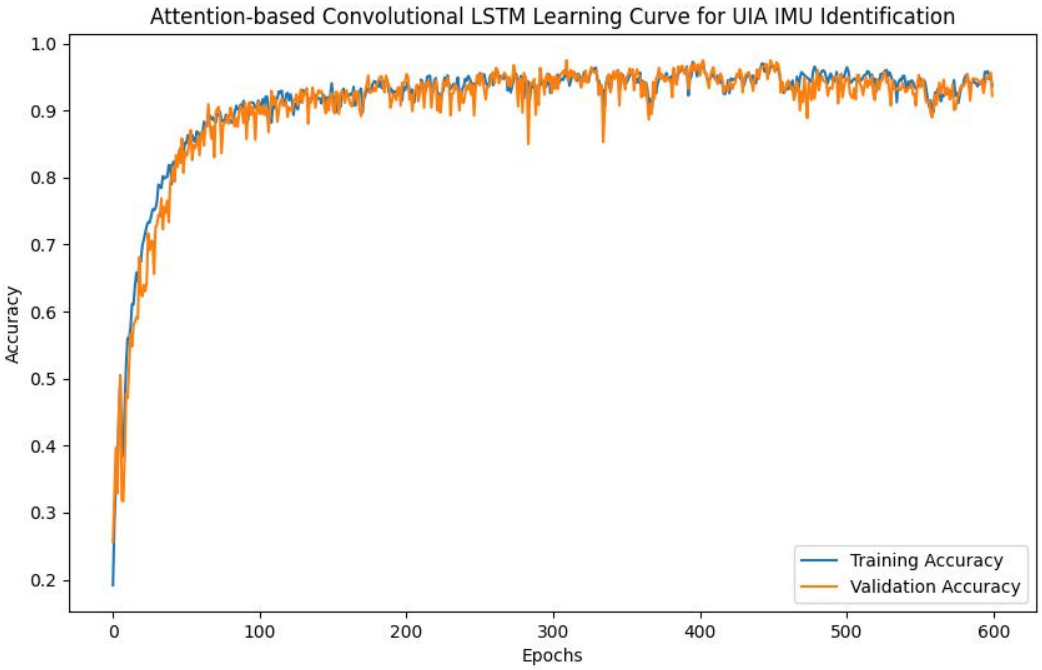
```
Epoch 295: val_accuracy did not improve from 0.96771
120/120 [==============================] - 11s 91ms/step - loss: 0.1664 - ...
    accuracy: 0.9391 - val_loss: 0.1901 - val_accuracy: 0.9240
Epoch 296/600
120/120 [==============================] - ETA: 0s - loss: 0.1791 - ...
    accuracy: 0.9354
Epoch 296: val_accuracy did not improve from 0.96771
120/120 [==============================] - 11s 89ms/step - loss: 0.1791 - ...
    accuracy: 0.9354 - val_loss: 0.1410 - val_accuracy: 0.9500
Epoch 297/600
120/120 [==============================] - ETA: 0s - loss: 0.1621 - ...
    accuracy: 0.9409
Epoch 297: val_accuracy did not improve from 0.96771
120/120 [==============================] - 11s 89ms/step - loss: 0.1621 - ...
    accuracy: 0.9409 - val_loss: 0.1073 - val_accuracy: 0.9552
Epoch 298/600
120/120 [==============================] - ETA: 0s - loss: 0.1247 - ...
    accuracy: 0.9544
Epoch 298: val_accuracy did not improve from 0.96771
120/120 [==============================] - 11s 91ms/step - loss: 0.1247 - ...
    accuracy: 0.9544 - val_loss: 0.1788 - val_accuracy: 0.9438
Epoch 299/600
120/120 [==============================] - ETA: 0s - loss: 0.2076 - ...
    accuracy: 0.9260
Epoch 299: val_accuracy did not improve from 0.96771
120/120 [==============================] - 11s 88ms/step - loss: 0.2076 - ...
    accuracy: 0.9260 - val_loss: 0.1487 - val_accuracy: 0.9521
Epoch 300/600
120/120 [==============================] - ETA: 0s - loss: 0.1770 - ...
    accuracy: 0.9424
Epoch 300: val_accuracy did not improve from 0.96771
120/120 [==============================] - 11s 88ms/step - loss: 0.1770 - ...
    accuracy: 0.9424 - val_loss: 0.2201 - val_accuracy: 0.9260
Epoch 301/600
120/120 [==============================] - ETA: 0s - loss: 0.1682 - ...
    accuracy: 0.9414
Epoch 301: val_accuracy did not improve from 0.96771
120/120 [==============================] - 11s 90ms/step - loss: 0.1682 - ...
    accuracy: 0.9414 - val_loss: 0.1383 - val_accuracy: 0.9500
Epoch 302/600
120/120 [==============================] - ETA: 0s - loss: 0.1653 - ...
    accuracy: 0.9438
Epoch 302: val_accuracy did not improve from 0.96771
120/120 [==============================] - 11s 88ms/step - loss: 0.1653 - ...
    accuracy: 0.9438 - val_loss: 0.1895 - val_accuracy: 0.9323
Epoch 303/600
120/120 [==============================] - ETA: 0s - loss: 0.1521 - ...
    accuracy: 0.9484
Epoch 303: val_accuracy did not improve from 0.96771
120/120 [==============================] - 11s 88ms/step - loss: 0.1521 - ...
    accuracy: 0.9484 - val_loss: 0.1370 - val_accuracy: 0.9521
Epoch 304/600
120/120 [==============================] - ETA: 0s - loss: 0.1349 - ...
    accuracy: 0.9536
Epoch 304: val_accuracy did not improve from 0.96771
120/120 [==============================] - 11s 91ms/step - loss: 0.1349 - ...
    accuracy: 0.9536 - val_loss: 0.2007 - val_accuracy: 0.9323
Epoch 305/600
120/120 [==============================] - ETA: 0s - loss: 0.1630 - ...
    accuracy: 0.9411
Epoch 305: val_accuracy did not improve from 0.96771
```

```
120/120 [==============================] - 11s 88ms/step - loss: 0.1630 - ...
    accuracy: 0.9411 - val_loss: 0.1469 - val_accuracy: 0.9458
Epoch 306/600
120/120 [==============================] - ETA: 0s - loss: 0.1456 - ...
    accuracy: 0.9471
Epoch 306: val_accuracy did not improve from 0.96771
120/120 [==============================] - 11s 89ms/step - loss: 0.1456 - ...
    accuracy: 0.9471 - val_loss: 0.1344 - val_accuracy: 0.9542
Epoch 307/600
120/120 [==============================] - ETA: 0s - loss: 0.1407 - ...
    accuracy: 0.9458
Epoch 307: val_accuracy did not improve from 0.96771
120/120 [==============================] - 11s 89ms/step - loss: 0.1407 - ...
    accuracy: 0.9458 - val_loss: 0.1417 - val_accuracy: 0.9563
Epoch 308/600
120/120 [==============================] - ETA: 0s - loss: 0.1480 - ...
    accuracy: 0.9458
Epoch 308: val_accuracy did not improve from 0.96771
120/120 [==============================] - 11s 88ms/step - loss: 0.1480 - ...
    accuracy: 0.9458 - val_loss: 0.1323 - val_accuracy: 0.9594
Epoch 309/600
120/120 [==============================] - ETA: 0s - loss: 0.1153 - ...
    accuracy: 0.9607
Epoch 309: val_accuracy did not improve from 0.96771
120/120 [==============================] - 11s 89ms/step - loss: 0.1153 - ...
    accuracy: 0.9607 - val_loss: 0.1286 - val_accuracy: 0.9542
Epoch 310/600
120/120 [==============================] - ETA: 0s - loss: 0.1331 - ...
    accuracy: 0.9503
Epoch 310: val_accuracy improved from 0.96771 to 0.97500, saving model to ...
    Attentive_Conv_LSTM_model.tf
120/120 [==============================] - 16s 132ms/step - loss: 0.1331 - ...
    accuracy: 0.9503 - val_loss: 0.0820 - val_accuracy: 0.9750
Epoch 311/600
120/120 [==============================] - ETA: 0s - loss: 0.1326 - ...
    accuracy: 0.9544
Epoch 311: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 88ms/step - loss: 0.1326 - ...
    accuracy: 0.9544 - val_loss: 0.1352 - val_accuracy: 0.9448
Epoch 312/600
120/120 [==============================] - ETA: 0s - loss: 0.1718 - ...
    accuracy: 0.9370
Epoch 312: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 90ms/step - loss: 0.1718 - ...
    accuracy: 0.9370 - val_loss: 0.1607 - val_accuracy: 0.9552
Epoch 313/600
120/120 [==============================] - ETA: 0s - loss: 0.1451 - ...
    accuracy: 0.9451
Epoch 313: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 88ms/step - loss: 0.1451 - ...
    accuracy: 0.9451 - val_loss: 0.1228 - val_accuracy: 0.9604
Epoch 314/600
120/120 [==============================] - ETA: 0s - loss: 0.1276 - ...
    accuracy: 0.9542
Epoch 314: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 88ms/step - loss: 0.1276 - ...
    accuracy: 0.9542 - val_loss: 0.1453 - val_accuracy: 0.9510
Epoch 315/600
120/120 [==============================] - ETA: 0s - loss: 0.1305 - ...
    accuracy: 0.9529
Epoch 315: val_accuracy did not improve from 0.97500
```

```
120/120 [==============================] - 11s 90ms/step - loss: 0.1305 - ...
    accuracy: 0.9529 - val_loss: 0.1096 - val_accuracy: 0.9583
Epoch 316/600
120/120 [==============================] - ETA: 0s - loss: 0.1223 - ...
    accuracy: 0.9547
Epoch 316: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 88ms/step - loss: 0.1223 - ...
    accuracy: 0.9547 - val_loss: 0.1434 - val_accuracy: 0.9458
Epoch 317/600
120/120 [==============================] - ETA: 0s - loss: 0.1561 - ...
    accuracy: 0.9440
Epoch 317: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 90ms/step - loss: 0.1561 - ...
    accuracy: 0.9440 - val_loss: 0.1345 - val_accuracy: 0.9542

.............

Epoch 524/600
120/120 [==============================] - ETA: 0s - loss: 0.1340 - ...
    accuracy: 0.9547
Epoch 524: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 89ms/step - loss: 0.1340 - ...
    accuracy: 0.9547 - val_loss: 0.1467 - val_accuracy: 0.9438
Epoch 525/600
120/120 [==============================] - ETA: 0s - loss: 0.1357 - ...
    accuracy: 0.9539
Epoch 525: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 89ms/step - loss: 0.1357 - ...
    accuracy: 0.9539 - val_loss: 0.1705 - val_accuracy: 0.9448
Epoch 526/600
120/120 [==============================] - ETA: 0s - loss: 0.1206 - ...
    accuracy: 0.9565
Epoch 526: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 89ms/step - loss: 0.1206 - ...
    accuracy: 0.9565 - val_loss: 0.1266 - val_accuracy: 0.9542
Epoch 527/600
120/120 [==============================] - ETA: 0s - loss: 0.1106 - ...
    accuracy: 0.9594
Epoch 527: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 88ms/step - loss: 0.1106 - ...
    accuracy: 0.9594 - val_loss: 0.1548 - val_accuracy: 0.9573
Epoch 528/600
120/120 [==============================] - ETA: 0s - loss: 0.1076 - ...
    accuracy: 0.9607
Epoch 528: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 88ms/step - loss: 0.1076 - ...
    accuracy: 0.9607 - val_loss: 0.1968 - val_accuracy: 0.9417
Epoch 529/600
120/120 [==============================] - ETA: 0s - loss: 0.1322 - ...
    accuracy: 0.9513
Epoch 529: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 89ms/step - loss: 0.1322 - ...
    accuracy: 0.9513 - val_loss: 0.1892 - val_accuracy: 0.9490
Epoch 530/600
120/120 [==============================] - ETA: 0s - loss: 0.1280 - ...
    accuracy: 0.9570
Epoch 530: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 90ms/step - loss: 0.1280 - ...
    accuracy: 0.9570 - val_loss: 0.2033 - val_accuracy: 0.9260
Epoch 531/600
120/120 [==============================] - ETA: 0s - loss: 0.1340 - ...
```

```
    accuracy: 0.9518
Epoch 531: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 89ms/step - loss: 0.1340 - ...
    accuracy: 0.9518 - val_loss: 0.2865 - val_accuracy: 0.9073
Epoch 532/600
120/120 [==============================] - ETA: 0s - loss: 0.1503 - ...
    accuracy: 0.9443
Epoch 532: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 89ms/step - loss: 0.1503 - ...
    accuracy: 0.9443 - val_loss: 0.1556 - val_accuracy: 0.9458
Epoch 533/600
120/120 [==============================] - ETA: 0s - loss: 0.1472 - ...
    accuracy: 0.9451
Epoch 533: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 88ms/step - loss: 0.1472 - ...
    accuracy: 0.9451 - val_loss: 0.1729 - val_accuracy: 0.9323
Epoch 534/600
120/120 [==============================] - ETA: 0s - loss: 0.1521 - ...
    accuracy: 0.9422
Epoch 534: val_accuracy did not improve from 0.97500
120/120 [==============================] - 10s 87ms/step - loss: 0.1521 - ...
    accuracy: 0.9422 - val_loss: 0.1704 - val_accuracy: 0.9323
Epoch 535/600
120/120 [==============================] - ETA: 0s - loss: 0.2423 - ...
    accuracy: 0.9122
Epoch 535: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 88ms/step - loss: 0.2423 - ...
    accuracy: 0.9122 - val_loss: 0.1928 - val_accuracy: 0.9312

.........

Epoch 592/600
120/120 [==============================] - ETA: 0s - loss: 0.1632 - ...
    accuracy: 0.9391
Epoch 592: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 89ms/step - loss: 0.1632 - ...
    accuracy: 0.9391 - val_loss: 0.1482 - val_accuracy: 0.9396
Epoch 593/600
120/120 [==============================] - ETA: 0s - loss: 0.1548 - ...
    accuracy: 0.9419
Epoch 593: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 89ms/step - loss: 0.1548 - ...
    accuracy: 0.9419 - val_loss: 0.1364 - val_accuracy: 0.9458
Epoch 594/600
120/120 [==============================] - ETA: 0s - loss: 0.1722 - ...
    accuracy: 0.9388
Epoch 594: val_accuracy did not improve from 0.97500
120/120 [==============================] - 10s 87ms/step - loss: 0.1722 - ...
    accuracy: 0.9388 - val_loss: 0.1750 - val_accuracy: 0.9260
Epoch 595/600
120/120 [==============================] - ETA: 0s - loss: 0.1266 - ...
    accuracy: 0.9578
Epoch 595: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 88ms/step - loss: 0.1266 - ...
    accuracy: 0.9578 - val_loss: 0.1564 - val_accuracy: 0.9385
Epoch 596/600
120/120 [==============================] - ETA: 0s - loss: 0.1198 - ...
    accuracy: 0.9560
Epoch 596: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 89ms/step - loss: 0.1198 - ...
    accuracy: 0.9560 - val_loss: 0.1400 - val_accuracy: 0.9500
```

```
Epoch 597/600
120/120 [==============================] - ETA: 0s - loss: 0.1095 - ...
    accuracy: 0.9586
Epoch 597: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 88ms/step - loss: 0.1095 - ...
    accuracy: 0.9586 - val_loss: 0.1483 - val_accuracy: 0.9479
Epoch 598/600
120/120 [==============================] - ETA: 0s - loss: 0.1439 - ...
    accuracy: 0.9466
Epoch 598: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 89ms/step - loss: 0.1439 - ...
    accuracy: 0.9466 - val_loss: 0.1408 - val_accuracy: 0.9531
Epoch 599/600
120/120 [==============================] - ETA: 0s - loss: 0.1453 - ...
    accuracy: 0.9516
Epoch 599: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 89ms/step - loss: 0.1453 - ...
    accuracy: 0.9516 - val_loss: 0.1346 - val_accuracy: 0.9563
Epoch 600/600
120/120 [==============================] - ETA: 0s - loss: 0.1636 - ...
    accuracy: 0.9370
Epoch 600: val_accuracy did not improve from 0.97500
120/120 [==============================] - 11s 88ms/step - loss: 0.1636 - ...
    accuracy: 0.9370 - val_loss: 0.2372 - val_accuracy: 0.9219
```




```
30/30 [==============================] - 1s 18ms/step - loss: 0.0820 - accuracy: 0.9750
Test Loss: 0.08203219622373581, Test Accuracy: 0.9750000238418579
30/30 [==============================] - 1s 16ms/step
<tf.Tensor: shape=(13, 13), dtype=int32, numpy=
array([[100,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,  77,   0,   0,   0,   0,   0,   1,   0,   0,   0,   0,   0],
       [  0,   0,  66,   0,   0,   0,   0,   4,   0,   0,   0,   0,   0],
       [  0,   0,   0,  64,   0,   2,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,  67,   0,   0,   0,   0,   0,   0,   0,   0],
```

```
 [  0,   0,   0,   0,   0,  68,   0,   0,   0,   0,   0,   0,   0],
 [  0,   0,   0,   0,   0,   0,  73,   0,   0,   2,   0,   0,   0],
 [  0,   0,   0,   0,   0,   0,   0,  68,   0,   0,   0,   0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,  70,   0,   0,   0,   0],
 [  0,   0,   0,   0,   3,   0,   0,   0,   0,  76,   1,   0,   0],
 [  0,   1,   0,   0,   1,   1,   1,   0,   0,   0,  58,   0,   0],
 [  0,   0,   0,   0,   0,   2,   2,   0,   0,   0,   0,  67,   3],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  82]],
dtype=int32)>
```

LSTM - Long Short Term Memory model

```python
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_ID_Walking_Gait_Dataset_8_1_2024
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_IMU_9ax_WG_Dataset_W_Calibration_Data_U_21_Des_23.csv
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_IMU_9ax_WG_Dataset_U_19_Des_23
/kaggle/input/uia_id_tcn_simplified_100_val/keras/v1.1/1/TCN_UIA_IMU_Simplified_100_acc.keras
/kaggle/input/uia_id_tcn_simplified_100_val/keras/.h5/1/TCN_Model.h5
```

```python
df = pd.read_csv('/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_ID_Walking_Gait_Dataset_8_1_2024')
```

```python
df
```

```
       Unnamed: 0   Timestamp        dt  Subject_no        q0        q1  \
0               0    0.034056  0.034056           0  0.999981 -0.005295
1               1    0.068112  0.034056           0  0.999967 -0.007995
2               2    0.102168  0.034056           0  0.999951 -0.009922
3               3    0.142416  0.040248           0  0.999923 -0.012365
4               4    0.182664  0.040248           0  0.999878 -0.015495
...           ...         ...       ...         ...       ...       ...
38754       38754  132.178021  0.039903          12 -0.978575 -0.198041
38755       38755  132.217924  0.039903          12 -0.980173 -0.194980
38756       38756  132.257827  0.039903          12 -0.982184 -0.185914
38757       38757  132.297729  0.039902          12 -0.984022 -0.175340
38758       38758  132.337632  0.039903          12 -0.985002 -0.166676
```

```
             q2        q3     Pitch      Roll         Yaw      accX      accY  \
0     -0.003050 -0.000229 -0.349617 -0.024413   -0.606688  0.001269 -0.007415
1     -0.001483 -0.000205 -0.170067 -0.022105   -0.916112 -0.014983 -0.010159
2     -0.000083 -0.000221 -0.009767 -0.025253   -1.136978 -0.024849 -0.013850
3      0.000831 -0.000712  0.094222 -0.082716   -1.417035 -0.025351 -0.018286
4      0.000539 -0.001899  0.058380 -0.218530   -1.775726 -0.018526 -0.021769
...         ...       ...       ...       ...         ...       ...       ...
38754 -0.020191  0.052557  3.458979 -5.437485 -337.283108  0.181484  0.031365
38755 -0.004102  0.035017  1.243237 -3.843961 -337.540503  0.114871 -0.046202
38756  0.004001  0.027092  0.126920 -3.136616 -338.566529  0.002566 -0.098474
38757  0.004847  0.030541  0.067100 -3.544386 -339.795425 -0.110981 -0.107848
38758  0.002149  0.044560  0.608490 -5.078503 -340.818421 -0.178788 -0.079801

           accZ      gyrX      gyrY      gyrZ  magX  magY  magZ
0      0.007225 -0.015511 -0.128565 -0.013460    72   -42     7
1      0.003744  0.009013 -0.156497 -0.001248    73   -43     7
2     -0.001352  0.032911 -0.179508 -0.004759    72   -43     8
3     -0.007889  0.054869 -0.196698 -0.028749    72   -43     7
4     -0.014033  0.074107 -0.206459 -0.063698    72   -43     5
...         ...       ...       ...       ...   ...   ...   ...
38754  0.091784  0.130059 -0.464335  1.226585    56   -27    41
38755  0.108562  0.023426 -0.404651  0.938618    55   -27    40
38756  0.088519 -0.176852 -0.326740  0.471631    55   -28    41
38757  0.041200 -0.331125 -0.271021 -0.084528    56   -30    39
38758 -0.012090 -0.334927 -0.259580 -0.616271    56   -31    36

[38759 rows x 20 columns]
```

From feature selection analysis using svm and random forest, top features:

#Feats_ID = ['Pitch', 'q1', 'Yaw', 'MotionDeg', 'q2', 'q3']

```
Feats = ['Pitch', 'Yaw', 'q1', 'magX', 'q2', 'magZ']

df['Gait_Identification'] = df['Subject_no']

Label = ['Gait_Identification']
```

```
Columns = ['Gait_Identification', 'Pitch', 'Yaw', 'q1', 'magX', 'q2', 'magZ']

df = df[Columns]

df
```

```
       Gait_Identification     Pitch         Yaw        q1  magX        q2  \
0                        0 -0.349617   -0.606688 -0.005295    72 -0.003050
1                        0 -0.170067   -0.916112 -0.007995    73 -0.001483
2                        0 -0.009767   -1.136978 -0.009922    72 -0.000083
3                        0  0.094222   -1.417035 -0.012365    72  0.000831
4                        0  0.058380   -1.775726 -0.015495    72  0.000539
...                    ...       ...         ...       ...   ...       ...
38754                   12  3.458979 -337.283108 -0.198041    56 -0.020191
38755                   12  1.243237 -337.540503 -0.194980    55 -0.004102
38756                   12  0.126920 -338.566529 -0.185914    55  0.004001
38757                   12  0.067100 -339.795425 -0.175340    56  0.004847
38758                   12  0.608490 -340.818421 -0.166676    56  0.002149

       magZ
0         7
1         7
2         8
3         7
4         5
...     ...
38754    41
38755    40
38756    41
38757    39
38758    36

[38759 rows x 7 columns]
```

```
df_ID = df.copy()

import pandas as pd
```

```
import numpy as np
from sklearn.preprocessing import StandardScaler


sequence_length = 30
overlap_percentage = 0.75


X_columns = Feats
y_column = Label

# Extrapolate features
X = df[X_columns].values


# Scale
scaler = StandardScaler()
df[X_columns] = scaler.fit_transform(X)


# Calculate the overlap and step size
overlap_size = int(sequence_length * overlap_percentage)
step_size = sequence_length - overlap_size

# Initialize lists to store sequences and labels
sequences = []
labels = []

# Iterate through the dataframe to create sequences
for i in range(0, len(df) - sequence_length + 1, step_size):
    sequence = df[X_columns].values[i:i + sequence_length]
    label_values = tuple(tuple(row) for row in df[y_column].values[i:i + sequence_length])  # Convert nested arrays to tuples

    '''
```

```
    Only include sequences with uniform label value, i.e.
    drop data in transition from one subject to another.
    Several users for same prediction is not a real-world
    scenario.

    '''

    if len(set(label_values)) == 1:
        label = label_values[0]
        sequences.append(sequence)
        labels.append(label)

# Convert lists to numpy arrays
X = np.array(sequences)
y = np.array(labels)
```

```
/tmp/ipykernel_26/4041143955.py:19: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[X_columns] = scaler.fit_transform(X)
```

```
print(X.shape)

print(y.shape)
```

```
(4800, 30, 6)
(4800, 1)
```

```
np.unique(y)
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

```
from sklearn.model_selection import train_test_split
```

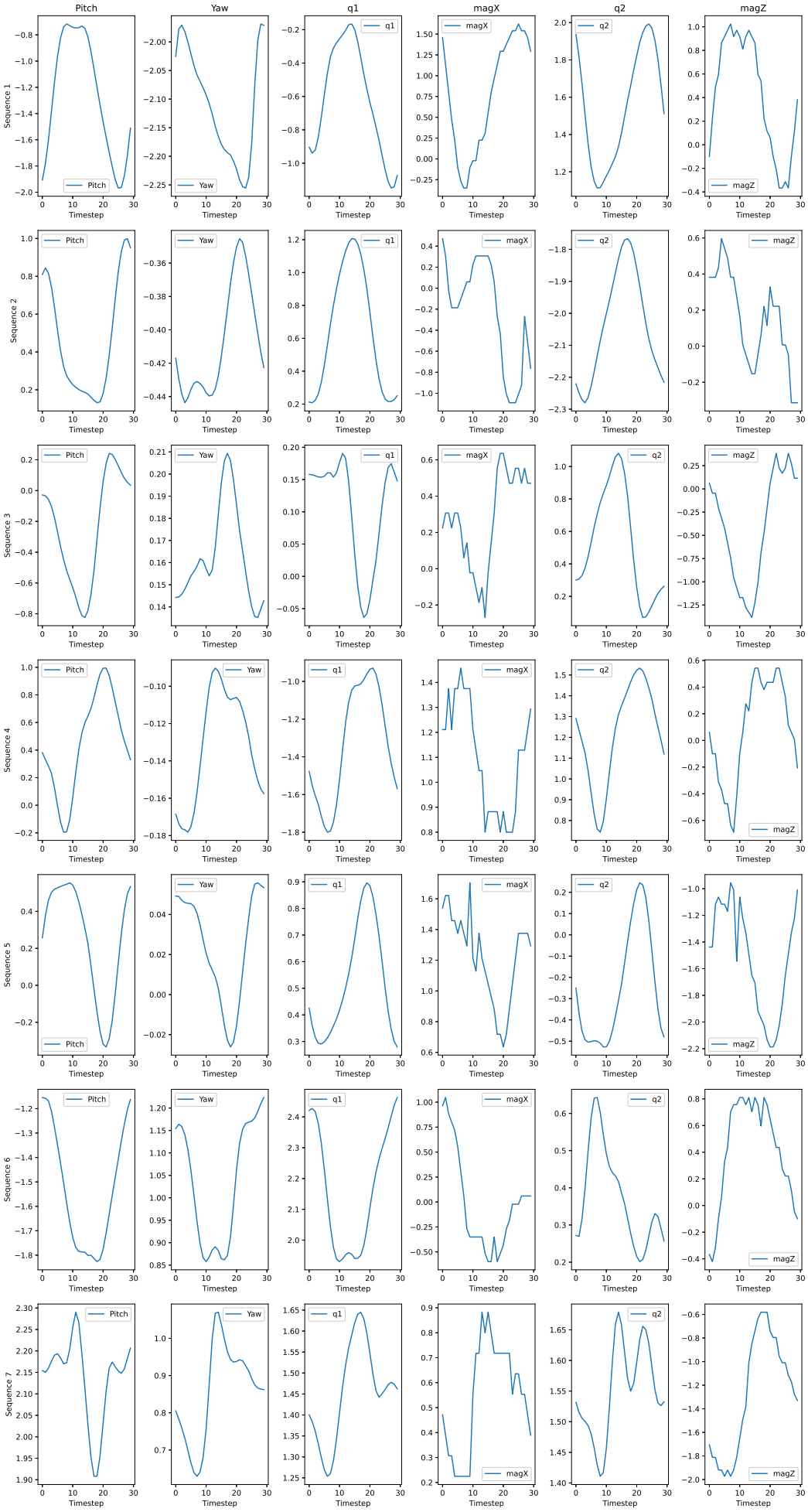
```
# Flatten y for stratification
y_flat = y.ravel()

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y_flat, random_state=42)

# Reshape labels so that they are 1 dinemnisonal

y_train = y_train.ravel()
y_test = y_test.ravel()

# Print the shapes of the resulting sets
print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
print("X_test shape:", X_test.shape)
print("y_test shape:", y_test.shape)
```



```
X_train shape: (3840, 30, 6)
y_train shape: (3840,)
X_test shape: (960, 30, 6)
y_test shape: (960,)
```

```
import matplotlib.pyplot as plt
import numpy as np

def plot_sequences(X, y, num_sequences=7):
    """
    Plot a random selection of sequences with individual spines for each feature.

    Parameters:
    - X: Feature sequences
    - y: Labels
    - num_sequences: Number of sequences to plot
    """
    num_sequences_to_plot = min(num_sequences, len(X))
```

```
    num_features = X.shape[2]
    figsize_height = 4 * num_sequences_to_plot  # Adjust the height based on your preference
    plt.figure(figsize=(15, figsize_height))
    indices = np.random.choice(len(X), num_sequences_to_plot, replace=False)

    for i, idx in enumerate(indices, 1):
        for j in range(num_features):
            plt.subplot(num_sequences_to_plot, num_features, (i - 1) * num_features + j + 1)
            plt.plot(X[idx, :, j], label=f"{Feats[j]}") #Extrapolate name from list of feats
            plt.xlabel("Timestep")

            if i == 1:
                plt.title(f"{Feats[j]}")
                plt.xlabel("Timestep")


            if j == 0:
                plt.ylabel(f"Sequence {i}")
                plt.xlabel("Timestep")

            plt.legend()

    plt.tight_layout()
    plt.savefig('Sequence_Samples_UIA_IMU.eps', format='eps')
    plt.savefig('Sequence_Samples_UIA_IMU.jpeg', format='jpeg')
    plt.show()


# Plot some random sequences from the training set
plot_sequences(X_train, y_train)


#Remember: Feats = ['Pitch' (1), 'Yaw'(2), 'q1'(3), 'magX'(4), 'q2'(5), 'magZ'(6)]
```


Pitch
Yaw
q1
magX
q2
magZ
Sequence 1
Sequence 2
Sequence 3
Sequence 4
Sequence 5
Sequence 6
Sequence 7
Timestep

```
input_dim = X.shape[2]
sequence_length = X.shape[1]
input_shape = (sequence_length, input_dim)
output_dim = len(np.unique(y))

'''
nb_filters = 64
kernel_size = 3
dilations = [1, 2, 3, 4, 5, 6]
nb_stacks = 1
'''

Units = 64 # units per lstm cell

DR_Layer_DR = 0.1

LSTM_Layer_DR = 0.0

LSTM_Layer_Recurr_DR = 0.1


input_shape
```

```
(30, 6)
```

```
from keras.models import Sequential
from keras.layers import LSTM, Flatten, Dense, Dropout, BatchNormalization
from keras.optimizers import Adam
from keras.regularizers import l2


def build_pure_LSTM(input_shape, num_classes, LSTM_Units, dropout_layer_dropout, lstm_dropout, lstm_recurr_dropout):

    Pure_LSTM_model = Sequential()
```

```
'''

Strict_CNN_model.add(Conv1D(filters=32, kernel_size=1, padding='same', input_shape=input_shape))
Strict_CNN_model.add(Conv1D(filters=32, kernel_size=2, padding='same', activation='relu', kernel_regularizer=l2(0.001), data_format='c
Strict_CNN_model.add(MaxPooling1D(pool_size=2, data_format='channels_first'))
Strict_CNN_model.add(BatchNormalization())

'''

'''
Pure_LSTM_model.add(LSTM(units=32,  # Positive integer, dimensionality of the output space
          activation='tanh',  # Activation function to use. Default: 'tanh'
          recurrent_activation='sigmoid',  # Activation function for the recurrent step. Default: 'sigmoid'
          use_bias=True,  # Boolean, whether the layer uses a bias vector. Default: True
          kernel_initializer='glorot_uniform',  # Initializer for the kernel weights matrix. Default: 'glorot_uniform'
          recurrent_initializer='orthogonal',  # Initializer for the recurrent kernel weights matrix. Default: 'orthogonal'
          bias_initializer='zeros',  # Initializer for the bias vector. Default: 'zeros'
          unit_forget_bias=True,  # Boolean, whether to add 1 to the bias of the forget gate at initialization. Default: True
          kernel_regularizer=None,  # Regularizer function applied to the kernel weights matrix. Default: None
          recurrent_regularizer=None,  # Regularizer function applied to the recurrent kernel weights matrix. Default: None
          bias_regularizer=None,  # Regularizer function applied to the bias vector. Default: None
          activity_regularizer=None,  # Regularizer function applied to the output of the layer. Default: None
          kernel_constraint=None,  # Constraint function applied to the kernel weights matrix. Default: None
          recurrent_constraint=None,  # Constraint function applied to the recurrent kernel weights matrix. Default: None
          bias_constraint=None,  # Constraint function applied to the bias vector. Default: None
          dropout=0.0,  # Float between 0 and 1, fraction of the units to drop for the linear transformation of the inputs. Default:
          recurrent_dropout=0.0,  # Float between 0 and 1, fraction of the units to drop for the linear transformation of the recurre
          implementation=1,  # Implementation mode, either 1 or 2. Default: 1
          return_sequences=False,  # Boolean, whether to return the last output in the output sequence, or the full sequence. Default
          return_state=False,  # Boolean, whether to return the last state in addition to the output. Default: False
          go_backwards=False,  # Boolean, whether the input sequence is processed backwards. Default: False
```

```
            stateful=False,  # Boolean, whether the layer is stateful. Default: False
            unroll=False,  # Boolean, whether the network will be unrolled. Default: False
            input_shape=(input_dim)  # Shape tuple (not including the batch size). Default: (None, None)
           ))

'''
```

```
Pure_LSTM_model.add(LSTM(units=Units,
                         activation='tanh',
                         recurrent_activation='sigmoid',
                         use_bias=True,
                         kernel_regularizer=l2(0.001),
                         dropout=LSTM_Layer_DR,
                         recurrent_dropout=LSTM_Layer_Recurr_DR,
                         implementation=2,
                         return_sequences=True,
                         input_shape=input_shape))

Pure_LSTM_model.add(LSTM(units=Units,
                         activation='tanh',
                         recurrent_activation='sigmoid',
                         use_bias=True,
                         kernel_regularizer=l2(0.001),
                         dropout=LSTM_Layer_DR,
                         recurrent_dropout=LSTM_Layer_Recurr_DR,
                         implementation=1,
                         return_sequences=True))
```

```
Pure_LSTM_model.add(Dropout(rate=DR_Layer_DR))


Pure_LSTM_model.add(LSTM(units=Units,
                         activation='tanh',
                         recurrent_activation='sigmoid',
                         use_bias=True,
                         kernel_regularizer=l2(0.001),
                         dropout=LSTM_Layer_DR,
                         recurrent_dropout=LSTM_Layer_Recurr_DR,
                         implementation=2,
                         return_sequences=True,
                         input_shape=input_shape))

Pure_LSTM_model.add(LSTM(units=Units,
                         activation='tanh',
                         recurrent_activation='sigmoid',
                         use_bias=True,
                         kernel_regularizer=l2(0.001),
                         dropout=LSTM_Layer_DR,
                         recurrent_dropout=LSTM_Layer_Recurr_DR,
                         implementation=1,
                         return_sequences=True))

Pure_LSTM_model.add(Dropout(rate=DR_Layer_DR))


Pure_LSTM_model.add(LSTM(units=Units,
                         activation='tanh',
                         recurrent_activation='sigmoid',
                         use_bias=True,
                         kernel_regularizer=l2(0.001),
                         dropout=LSTM_Layer_DR,
                         recurrent_dropout=LSTM_Layer_Recurr_DR,
```

```
                              implementation=1,
                              return_sequences=False))

    Pure_LSTM_model.add(Dropout(rate=DR_Layer_DR))



    Pure_LSTM_model.add(Dense(output_dim, activation='softmax', kernel_regularizer=l2(0.001)))

    optimizer = Adam(learning_rate=0.01)  # You can adjust the learning rate

    Pure_LSTM_model.compile(loss='sparse_categorical_crossentropy',
                  optimizer=optimizer,
                  metrics=['accuracy'])

    return Pure_LSTM_model
```



```
Pure_LSTM_model = build_pure_LSTM(input_shape, output_dim, Units, DR_Layer_DR, LSTM_Layer_DR, LSTM_Layer_Recurr_DR)

Pure_LSTM_model.summary()

Model: "sequential_39"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_219 (LSTM)             (None, 30, 64)            18176

 lstm_220 (LSTM)             (None, 30, 64)            33024

 dropout_103 (Dropout)       (None, 30, 64)            0

 lstm_221 (LSTM)             (None, 30, 64)            33024

 lstm_222 (LSTM)             (None, 30, 64)            33024
```

```
 dropout_104 (Dropout)       (None, 30, 64)            0

 lstm_223 (LSTM)             (None, 64)                33024

 dropout_105 (Dropout)       (None, 64)                0

 dense_20 (Dense)            (None, 13)                845

=================================================================
Total params: 151117 (590.30 KB)
Trainable params: 151117 (590.30 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
from keras.callbacks import ModelCheckpoint

best_validation_callback = ModelCheckpoint(
    'Pure_LSTM_model.tf',  # Filename to save the model in the SavedModel format
    monitor='val_accuracy',        # Monitor the validation accuracy
    verbose=1,                     # Logging level
    save_best_only=True,           # Save only when the validation accuracy improves
    mode='max',                    # Mode 'max' because we are monitoring 'val_accuracy'
    save_format='tf'               # Explicitly state to save in TensorFlow SavedModel format
)


# Fit CNN to data (training)
Pure_LSTM_model_history = Pure_LSTM_model.fit(X_train, y_train,
                         epochs=600, batch_size=32,
                         validation_data=(X_test, y_test),
                        callbacks=[best_validation_callback])


import matplotlib.pyplot as plt
```

```
import numpy as np


# Plotting the learning curve
plt.figure(figsize=(10, 6))
plt.plot(Pure_LSTM_model_history.history['accuracy'], label='Training Accuracy')
plt.plot(Pure_LSTM_model_history.history['val_accuracy'], label='Validation Accuracy')
plt.title('LSTM Network Learning Curve for UIA IMU Identification')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.savefig('UIA_IMU_Pure_LSTM_Gait_ID_14_1_2024_trials.eps', format='eps')
plt.savefig('UIA_IMU_Pure_LSTM_Gait_ID_14_1_2024_trials.jpeg', format='jpeg')
plt.show()

# Save the Pure_LSTM_model summary to a text file
with open('summary_UIA_IMU_Pure_LSTM_Gait_ID_14_1_2024_trials.txt', 'w') as f:
    Pure_LSTM_model.summary(print_fn=lambda x: f.write(x + '\n'))


# Load the best version of model

from tensorflow.keras.models import load_model

Pure_LSTM_model = load_model('Pure_LSTM_model.tf')



# Evaluate best version of model on the test set

test_loss, test_accuracy = Pure_LSTM_model.evaluate(X_test, y_test)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')

import tensorflow as tf
```

```
classes_X = Pure_LSTM_model.predict(X_test)

classes_X.shape

#classes_X[0]
'''The predict function outputs a probability for each label.
Need to extrapolate the max value per prediction.'''

classes_X = np.argmax(classes_X, axis=1)


cm_Pure_LSTM_model = tf.math.confusion_matrix(labels=y_test, predictions=classes_X, num_classes=output_dim)

cm_Pure_LSTM_model
```

```
    Epoch 1/600
120/120 [==============================] - ETA: 0s - loss: 2.3913 - accuracy: 0.2174
Epoch 1: val_accuracy improved from -inf to 0.20729, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 46s 315ms/step - loss: 2.3913 - accuracy: 0.2174 - val_loss: 2.1045 - ...
   val_accuracy: 0.2073
Epoch 2/600
120/120 [==============================] - ETA: 0s - loss: 2.0363 - accuracy: 0.2669
Epoch 2: val_accuracy improved from 0.20729 to 0.30521, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 35s 289ms/step - loss: 2.0363 - accuracy: 0.2669 - val_loss: 2.0431 - ...
   val_accuracy: 0.3052
Epoch 3/600
120/120 [==============================] - ETA: 0s - loss: 1.8514 - accuracy: 0.3724
Epoch 3: val_accuracy improved from 0.30521 to 0.39896, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 35s 290ms/step - loss: 1.8514 - accuracy: 0.3724 - val_loss: 1.6778 - ...
   val_accuracy: 0.3990
Epoch 4/600
120/120 [==============================] - ETA: 0s - loss: 1.7128 - accuracy: 0.4099
Epoch 4: val_accuracy improved from 0.39896 to 0.43646, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 36s 304ms/step - loss: 1.7128 - accuracy: 0.4099 - val_loss: 1.6339 - ...
   val_accuracy: 0.4365
Epoch 5/600
120/120 [==============================] - ETA: 0s - loss: 1.7836 - accuracy: 0.3893
```

```
Epoch 5: val_accuracy did not improve from 0.43646
120/120 [==============================] - 25s 205ms/step - loss: 1.7836 - accuracy: 0.3893 - val_loss: 1.7567 - ...
   val_accuracy: 0.3927
Epoch 6/600
120/120 [==============================] - ETA: 0s - loss: 1.7294 - accuracy: 0.3924
Epoch 6: val_accuracy did not improve from 0.43646
120/120 [==============================] - 25s 205ms/step - loss: 1.7294 - accuracy: 0.3924 - val_loss: 1.8003 - ...
   val_accuracy: 0.3979
Epoch 7/600
120/120 [==============================] - ETA: 0s - loss: 1.7037 - accuracy: 0.4190
Epoch 7: val_accuracy improved from 0.43646 to 0.48958, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 34s 287ms/step - loss: 1.7037 - accuracy: 0.4190 - val_loss: 1.5085 - ...
   val_accuracy: 0.4896
Epoch 8/600
120/120 [==============================] - ETA: 0s - loss: 1.8606 - accuracy: 0.4091
Epoch 8: val_accuracy did not improve from 0.48958
120/120 [==============================] - 25s 205ms/step - loss: 1.8606 - accuracy: 0.4091 - val_loss: 1.7336 - ...
   val_accuracy: 0.4583
Epoch 9/600
120/120 [==============================] - ETA: 0s - loss: 1.8619 - accuracy: 0.4229
Epoch 9: val_accuracy did not improve from 0.48958
120/120 [==============================] - 25s 206ms/step - loss: 1.8619 - accuracy: 0.4229 - val_loss: 1.9186 - ...
   val_accuracy: 0.4094
Epoch 10/600
120/120 [==============================] - ETA: 0s - loss: 1.7691 - accuracy: 0.4443
Epoch 10: val_accuracy did not improve from 0.48958
120/120 [==============================] - 25s 204ms/step - loss: 1.7691 - accuracy: 0.4443 - val_loss: 1.9696 - ...
   val_accuracy: 0.4365
Epoch 11/600
120/120 [==============================] - ETA: 0s - loss: 1.9191 - accuracy: 0.4495
Epoch 11: val_accuracy improved from 0.48958 to 0.50833, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 36s 299ms/step - loss: 1.9191 - accuracy: 0.4495 - val_loss: 1.6271 - ...
   val_accuracy: 0.5083
Epoch 12/600
120/120 [==============================] - ETA: 0s - loss: 1.6865 - accuracy: 0.4771
Epoch 12: val_accuracy improved from 0.50833 to 0.51458, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 35s 290ms/step - loss: 1.6865 - accuracy: 0.4771 - val_loss: 1.5915 - ...
   val_accuracy: 0.5146
Epoch 13/600
```

```
120/120 [==============================] - ETA: 0s - loss: 1.8965 - accuracy: 0.4701
Epoch 13: val_accuracy did not improve from 0.51458
120/120 [==============================] - 25s 205ms/step - loss: 1.8965 - accuracy: 0.4701 - val_loss: 1.9913 - ...
   val_accuracy: 0.4688
Epoch 14/600
120/120 [==============================] - ETA: 0s - loss: 1.9301 - accuracy: 0.4607
Epoch 14: val_accuracy did not improve from 0.51458
120/120 [==============================] - 25s 206ms/step - loss: 1.9301 - accuracy: 0.4607 - val_loss: 1.6785 - ...
   val_accuracy: 0.5042
Epoch 15/600
120/120 [==============================] - ETA: 0s - loss: 1.9409 - accuracy: 0.4253
Epoch 15: val_accuracy did not improve from 0.51458
120/120 [==============================] - 25s 206ms/step - loss: 1.9409 - accuracy: 0.4253 - val_loss: 1.7497 - ...
   val_accuracy: 0.4865
Epoch 16/600
120/120 [==============================] - ETA: 0s - loss: 1.6186 - accuracy: 0.4904
Epoch 16: val_accuracy did not improve from 0.51458
120/120 [==============================] - 25s 209ms/step - loss: 1.6186 - accuracy: 0.4904 - val_loss: 1.6394 - ...
   val_accuracy: 0.4677
Epoch 17/600
120/120 [==============================] - ETA: 0s - loss: 1.9953 - accuracy: 0.3958
Epoch 17: val_accuracy did not improve from 0.51458
120/120 [==============================] - 25s 206ms/step - loss: 1.9953 - accuracy: 0.3958 - val_loss: 2.1050 - ...
   val_accuracy: 0.3781
Epoch 18/600
120/120 [==============================] - ETA: 0s - loss: 2.5834 - accuracy: 0.3557
Epoch 18: val_accuracy did not improve from 0.51458
120/120 [==============================] - 25s 208ms/step - loss: 2.5834 - accuracy: 0.3557 - val_loss: 2.2306 - ...
   val_accuracy: 0.4115
Epoch 19/600
120/120 [==============================] - ETA: 0s - loss: 2.2282 - accuracy: 0.4057
Epoch 19: val_accuracy did not improve from 0.51458
120/120 [==============================] - 25s 207ms/step - loss: 2.2282 - accuracy: 0.4057 - val_loss: 2.5442 - ...
   val_accuracy: 0.3521
Epoch 20/600
120/120 [==============================] - ETA: 0s - loss: 2.3120 - accuracy: 0.3898
Epoch 20: val_accuracy did not improve from 0.51458
120/120 [==============================] - 25s 210ms/step - loss: 2.3120 - accuracy: 0.3898 - val_loss: 2.0665 - ...
   val_accuracy: 0.4021
```

```
Epoch 21/600
120/120 [==============================] - ETA: 0s - loss: 2.3520 - accuracy: 0.3503
Epoch 21: val_accuracy did not improve from 0.51458
120/120 [==============================] - 25s 209ms/step - loss: 2.3520 - accuracy: 0.3503 - val_loss: 2.2037 - ...
   val_accuracy: 0.3885
Epoch 22/600
120/120 [==============================] - ETA: 0s - loss: 2.1140 - accuracy: 0.4195
Epoch 22: val_accuracy did not improve from 0.51458
120/120 [==============================] - 25s 206ms/step - loss: 2.1140 - accuracy: 0.4195 - val_loss: 1.9344 - ...
   val_accuracy: 0.4573
Epoch 23/600
120/120 [==============================] - ETA: 0s - loss: 1.9109 - accuracy: 0.4505
Epoch 23: val_accuracy did not improve from 0.51458
120/120 [==============================] - 25s 206ms/step - loss: 1.9109 - accuracy: 0.4505 - val_loss: 1.7820 - ...
   val_accuracy: 0.4344
Epoch 24/600
120/120 [==============================] - ETA: 0s - loss: 1.8427 - accuracy: 0.4500
Epoch 24: val_accuracy did not improve from 0.51458
120/120 [==============================] - 25s 208ms/step - loss: 1.8427 - accuracy: 0.4500 - val_loss: 1.7362 - ...
   val_accuracy: 0.4729
Epoch 25/600
120/120 [==============================] - ETA: 0s - loss: 1.7757 - accuracy: 0.4729
Epoch 25: val_accuracy did not improve from 0.51458
120/120 [==============================] - 25s 206ms/step - loss: 1.7757 - accuracy: 0.4729 - val_loss: 1.6631 - ...
   val_accuracy: 0.4573
Epoch 26/600
120/120 [==============================] - ETA: 0s - loss: 1.6840 - accuracy: 0.4768
Epoch 26: val_accuracy did not improve from 0.51458
120/120 [==============================] - 25s 209ms/step - loss: 1.6840 - accuracy: 0.4768 - val_loss: 1.5845 - ...
   val_accuracy: 0.5125
Epoch 27/600
120/120 [==============================] - ETA: 0s - loss: 1.6280 - accuracy: 0.4922
Epoch 27: val_accuracy did not improve from 0.51458
120/120 [==============================] - 25s 207ms/step - loss: 1.6280 - accuracy: 0.4922 - val_loss: 1.5172 - ...
   val_accuracy: 0.5063
Epoch 28/600
120/120 [==============================] - ETA: 0s - loss: 1.5807 - accuracy: 0.4935
Epoch 28: val_accuracy improved from 0.51458 to 0.52917, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 37s 306ms/step - loss: 1.5807 - accuracy: 0.4935 - val_loss: 1.4918 - ...
```

```
   val_accuracy: 0.5292
Epoch 29/600
120/120 [==============================] - ETA: 0s - loss: 1.5688 - accuracy: 0.4901
Epoch 29: val_accuracy improved from 0.52917 to 0.55521, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 35s 294ms/step - loss: 1.5688 - accuracy: 0.4901 - val_loss: 1.4645 - ...
   val_accuracy: 0.5552
Epoch 30/600
120/120 [==============================] - ETA: 0s - loss: 1.6503 - accuracy: 0.5221
Epoch 30: val_accuracy improved from 0.55521 to 0.56042, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 35s 293ms/step - loss: 1.6503 - accuracy: 0.5221 - val_loss: 1.5829 - ...
   val_accuracy: 0.5604
Epoch 31/600
120/120 [==============================] - ETA: 0s - loss: 1.5856 - accuracy: 0.5503
Epoch 31: val_accuracy improved from 0.56042 to 0.59271, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 37s 309ms/step - loss: 1.5856 - accuracy: 0.5503 - val_loss: 1.4402 - ...
   val_accuracy: 0.5927
Epoch 32/600
120/120 [==============================] - ETA: 0s - loss: 1.8328 - accuracy: 0.4932
Epoch 32: val_accuracy did not improve from 0.59271
120/120 [==============================] - 25s 207ms/step - loss: 1.8328 - accuracy: 0.4932 - val_loss: 2.1283 - ...
   val_accuracy: 0.4417
Epoch 33/600
120/120 [==============================] - ETA: 0s - loss: 2.0227 - accuracy: 0.4357
Epoch 33: val_accuracy did not improve from 0.59271
120/120 [==============================] - 25s 206ms/step - loss: 2.0227 - accuracy: 0.4357 - val_loss: 1.7900 - ...
   val_accuracy: 0.4573
Epoch 34/600
120/120 [==============================] - ETA: 0s - loss: 1.7423 - accuracy: 0.4979
Epoch 34: val_accuracy did not improve from 0.59271
120/120 [==============================] - 25s 207ms/step - loss: 1.7423 - accuracy: 0.4979 - val_loss: 1.5774 - ...
   val_accuracy: 0.5479
Epoch 35/600
120/120 [==============================] - ETA: 0s - loss: 1.8159 - accuracy: 0.4802
Epoch 35: val_accuracy did not improve from 0.59271
120/120 [==============================] - 25s 208ms/step - loss: 1.8159 - accuracy: 0.4802 - val_loss: 2.0117 - ...
   val_accuracy: 0.4771
Epoch 36/600
120/120 [==============================] - ETA: 0s - loss: 1.8285 - accuracy: 0.5128
Epoch 36: val_accuracy did not improve from 0.59271
```

```
120/120 [==============================] - 25s 207ms/step - loss: 1.8285 - accuracy: 0.5128 - val_loss: 1.7194 - ...
    val_accuracy: 0.5427
Epoch 37/600
120/120 [==============================] - ETA: 0s - loss: 1.7366 - accuracy: 0.5125
Epoch 37: val_accuracy did not improve from 0.59271
120/120 [==============================] - 25s 208ms/step - loss: 1.7366 - accuracy: 0.5125 - val_loss: 1.7103 - ...
    val_accuracy: 0.5177
Epoch 38/600
120/120 [==============================] - ETA: 0s - loss: 1.6763 - accuracy: 0.5203
Epoch 38: val_accuracy did not improve from 0.59271
120/120 [==============================] - 25s 206ms/step - loss: 1.6763 - accuracy: 0.5203 - val_loss: 1.6009 - ...
    val_accuracy: 0.5562
Epoch 39/600
120/120 [==============================] - ETA: 0s - loss: 1.6993 - accuracy: 0.5154
Epoch 39: val_accuracy did not improve from 0.59271
120/120 [==============================] - 25s 207ms/step - loss: 1.6993 - accuracy: 0.5154 - val_loss: 1.6569 - ...
    val_accuracy: 0.5531
Epoch 40/600
120/120 [==============================] - ETA: 0s - loss: 1.6550 - accuracy: 0.5354
Epoch 40: val_accuracy did not improve from 0.59271
120/120 [==============================] - 25s 208ms/step - loss: 1.6550 - accuracy: 0.5354 - val_loss: 1.6134 - ...
    val_accuracy: 0.5562
Epoch 41/600
120/120 [==============================] - ETA: 0s - loss: 1.6208 - accuracy: 0.5141
Epoch 41: val_accuracy did not improve from 0.59271
120/120 [==============================] - 25s 207ms/step - loss: 1.6208 - accuracy: 0.5141 - val_loss: 1.5495 - ...
    val_accuracy: 0.5500
Epoch 42/600
120/120 [==============================] - ETA: 0s - loss: 1.5107 - accuracy: 0.5466
Epoch 42: val_accuracy did not improve from 0.59271
120/120 [==============================] - 25s 209ms/step - loss: 1.5107 - accuracy: 0.5466 - val_loss: 1.4358 - ...
    val_accuracy: 0.5448
Epoch 43/600
120/120 [==============================] - ETA: 0s - loss: 1.4882 - accuracy: 0.5466
Epoch 43: val_accuracy did not improve from 0.59271
120/120 [==============================] - 25s 206ms/step - loss: 1.4882 - accuracy: 0.5466 - val_loss: 1.4385 - ...
    val_accuracy: 0.5531
Epoch 44/600
120/120 [==============================] - ETA: 0s - loss: 1.4672 - accuracy: 0.5617
```

```
Epoch 44: val_accuracy did not improve from 0.59271
120/120 [==============================] - 25s 207ms/step - loss: 1.4672 - accuracy: 0.5617 - val_loss: 1.3409 - ...
   val_accuracy: 0.5906
Epoch 45/600
120/120 [==============================] - ETA: 0s - loss: 1.3865 - accuracy: 0.5943
Epoch 45: val_accuracy improved from 0.59271 to 0.60729, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 35s 293ms/step - loss: 1.3865 - accuracy: 0.5943 - val_loss: 1.3936 - ...
   val_accuracy: 0.6073
Epoch 46/600
120/120 [==============================] - ETA: 0s - loss: 1.4361 - accuracy: 0.5753
Epoch 46: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 205ms/step - loss: 1.4361 - accuracy: 0.5753 - val_loss: 1.3743 - ...
   val_accuracy: 0.5813
Epoch 47/600
120/120 [==============================] - ETA: 0s - loss: 1.4238 - accuracy: 0.5716
Epoch 47: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 207ms/step - loss: 1.4238 - accuracy: 0.5716 - val_loss: 1.3841 - ...
   val_accuracy: 0.5813
Epoch 48/600
120/120 [==============================] - ETA: 0s - loss: 2.0894 - accuracy: 0.4810
Epoch 48: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 207ms/step - loss: 2.0894 - accuracy: 0.4810 - val_loss: 1.7900 - ...
   val_accuracy: 0.5448
Epoch 49/600
120/120 [==============================] - ETA: 0s - loss: 1.7909 - accuracy: 0.5286
Epoch 49: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 207ms/step - loss: 1.7909 - accuracy: 0.5286 - val_loss: 1.7613 - ...
   val_accuracy: 0.5156
Epoch 50/600
120/120 [==============================] - ETA: 0s - loss: 1.7782 - accuracy: 0.5250
Epoch 50: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 209ms/step - loss: 1.7782 - accuracy: 0.5250 - val_loss: 1.6441 - ...
   val_accuracy: 0.5625
Epoch 51/600
120/120 [==============================] - ETA: 0s - loss: 2.0279 - accuracy: 0.4453
Epoch 51: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 208ms/step - loss: 2.0279 - accuracy: 0.4453 - val_loss: 1.9082 - ...
   val_accuracy: 0.4844
Epoch 52/600
```

```
120/120 [==============================] - ETA: 0s - loss: 2.7672 - accuracy: 0.4177
Epoch 52: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 210ms/step - loss: 2.7672 - accuracy: 0.4177 - val_loss: 3.2127 - ...
   val_accuracy: 0.3906
Epoch 53/600
120/120 [==============================] - ETA: 0s - loss: 3.0408 - accuracy: 0.4203
Epoch 53: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 209ms/step - loss: 3.0408 - accuracy: 0.4203 - val_loss: 2.7326 - ...
   val_accuracy: 0.4500
Epoch 54/600
120/120 [==============================] - ETA: 0s - loss: 2.8996 - accuracy: 0.4289
Epoch 54: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 209ms/step - loss: 2.8996 - accuracy: 0.4289 - val_loss: 4.2621 - ...
   val_accuracy: 0.2729
Epoch 55/600
120/120 [==============================] - ETA: 0s - loss: 4.3414 - accuracy: 0.1638
Epoch 55: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 208ms/step - loss: 4.3414 - accuracy: 0.1638 - val_loss: 3.9421 - ...
   val_accuracy: 0.2844
Epoch 56/600
120/120 [==============================] - ETA: 0s - loss: 3.5933 - accuracy: 0.2354
Epoch 56: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 208ms/step - loss: 3.5933 - accuracy: 0.2354 - val_loss: 3.1825 - ...
   val_accuracy: 0.2990
Epoch 57/600
120/120 [==============================] - ETA: 0s - loss: 3.1218 - accuracy: 0.2833
Epoch 57: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 207ms/step - loss: 3.1218 - accuracy: 0.2833 - val_loss: 2.8570 - ...
   val_accuracy: 0.3260
Epoch 58/600
120/120 [==============================] - ETA: 0s - loss: 2.7569 - accuracy: 0.3740
Epoch 58: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 210ms/step - loss: 2.7569 - accuracy: 0.3740 - val_loss: 2.5792 - ...
   val_accuracy: 0.3906
Epoch 59/600
120/120 [==============================] - ETA: 0s - loss: 2.7085 - accuracy: 0.3651
Epoch 59: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 208ms/step - loss: 2.7085 - accuracy: 0.3651 - val_loss: 2.4721 - ...
   val_accuracy: 0.3896
```

```
Epoch 60/600
120/120 [==============================] - ETA: 0s - loss: 2.6115 - accuracy: 0.3708
Epoch 60: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 209ms/step - loss: 2.6115 - accuracy: 0.3708 - val_loss: 2.4545 - ...
   val_accuracy: 0.3583
Epoch 61/600
120/120 [==============================] - ETA: 0s - loss: 2.6879 - accuracy: 0.3729
Epoch 61: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 209ms/step - loss: 2.6879 - accuracy: 0.3729 - val_loss: 2.4810 - ...
   val_accuracy: 0.4250
Epoch 62/600
120/120 [==============================] - ETA: 0s - loss: 2.4116 - accuracy: 0.4354
Epoch 62: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 208ms/step - loss: 2.4116 - accuracy: 0.4354 - val_loss: 2.2401 - ...
   val_accuracy: 0.4583
Epoch 63/600
120/120 [==============================] - ETA: 0s - loss: 2.1730 - accuracy: 0.4792
Epoch 63: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 208ms/step - loss: 2.1730 - accuracy: 0.4792 - val_loss: 2.1367 - ...
   val_accuracy: 0.4719
Epoch 64/600
120/120 [==============================] - ETA: 0s - loss: 2.1032 - accuracy: 0.4706
Epoch 64: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 208ms/step - loss: 2.1032 - accuracy: 0.4706 - val_loss: 1.9566 - ...
   val_accuracy: 0.5542
Epoch 65/600
120/120 [==============================] - ETA: 0s - loss: 1.9534 - accuracy: 0.5388
Epoch 65: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 207ms/step - loss: 1.9534 - accuracy: 0.5388 - val_loss: 1.8830 - ...
   val_accuracy: 0.5615
Epoch 66/600
120/120 [==============================] - ETA: 0s - loss: 1.8643 - accuracy: 0.5302
Epoch 66: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 208ms/step - loss: 1.8643 - accuracy: 0.5302 - val_loss: 1.6662 - ...
   val_accuracy: 0.5958
Epoch 67/600
120/120 [==============================] - ETA: 0s - loss: 1.7454 - accuracy: 0.5576
Epoch 67: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 208ms/step - loss: 1.7454 - accuracy: 0.5576 - val_loss: 1.5618 - ...
```

```
   val_accuracy: 0.6073
Epoch 68/600
120/120 [==============================] - ETA: 0s - loss: 1.6648 - accuracy: 0.5651
Epoch 68: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 209ms/step - loss: 1.6648 - accuracy: 0.5651 - val_loss: 1.5093 - ...
   val_accuracy: 0.5990
Epoch 69/600
120/120 [==============================] - ETA: 0s - loss: 1.7498 - accuracy: 0.5424
Epoch 69: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 207ms/step - loss: 1.7498 - accuracy: 0.5424 - val_loss: 1.8015 - ...
   val_accuracy: 0.5115
Epoch 70/600
120/120 [==============================] - ETA: 0s - loss: 1.6813 - accuracy: 0.5526
Epoch 70: val_accuracy did not improve from 0.60729
120/120 [==============================] - 25s 208ms/step - loss: 1.6813 - accuracy: 0.5526 - val_loss: 1.5114 - ...
   val_accuracy: 0.6042
Epoch 71/600
120/120 [==============================] - ETA: 0s - loss: 1.6052 - accuracy: 0.5820
Epoch 71: val_accuracy improved from 0.60729 to 0.62083, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 36s 305ms/step - loss: 1.6052 - accuracy: 0.5820 - val_loss: 1.4845 - ...
   val_accuracy: 0.6208
Epoch 72/600
120/120 [==============================] - ETA: 0s - loss: 1.7785 - accuracy: 0.5531
Epoch 72: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 210ms/step - loss: 1.7785 - accuracy: 0.5531 - val_loss: 2.9254 - ...
   val_accuracy: 0.4708
Epoch 73/600
120/120 [==============================] - ETA: 0s - loss: 3.5021 - accuracy: 0.1805
Epoch 73: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 208ms/step - loss: 3.5021 - accuracy: 0.1805 - val_loss: 3.3080 - ...
   val_accuracy: 0.2323
Epoch 74/600
120/120 [==============================] - ETA: 0s - loss: 2.8718 - accuracy: 0.2570
Epoch 74: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 209ms/step - loss: 2.8718 - accuracy: 0.2570 - val_loss: 2.4805 - ...
   val_accuracy: 0.3198
Epoch 75/600
120/120 [==============================] - ETA: 0s - loss: 2.4341 - accuracy: 0.3740
Epoch 75: val_accuracy did not improve from 0.62083
```

```
120/120 [==============================] - 25s 208ms/step - loss: 2.4341 - accuracy: 0.3740 - val_loss: 3.0427 - ...
    val_accuracy: 0.2406
Epoch 76/600
120/120 [==============================] - ETA: 0s - loss: 4.0052 - accuracy: 0.2461
Epoch 76: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 209ms/step - loss: 4.0052 - accuracy: 0.2461 - val_loss: 3.4371 - ...
    val_accuracy: 0.3042
Epoch 77/600
120/120 [==============================] - ETA: 0s - loss: 3.2852 - accuracy: 0.3180
Epoch 77: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 207ms/step - loss: 3.2852 - accuracy: 0.3180 - val_loss: 3.2150 - ...
    val_accuracy: 0.2354
Epoch 78/600
120/120 [==============================] - ETA: 0s - loss: 2.8874 - accuracy: 0.3776
Epoch 78: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 207ms/step - loss: 2.8874 - accuracy: 0.3776 - val_loss: 2.6504 - ...
    val_accuracy: 0.4604
Epoch 79/600
120/120 [==============================] - ETA: 0s - loss: 2.7143 - accuracy: 0.4159
Epoch 79: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 208ms/step - loss: 2.7143 - accuracy: 0.4159 - val_loss: 2.5119 - ...
    val_accuracy: 0.4885
Epoch 80/600
120/120 [==============================] - ETA: 0s - loss: 2.5405 - accuracy: 0.4404
Epoch 80: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 210ms/step - loss: 2.5405 - accuracy: 0.4404 - val_loss: 2.3898 - ...
    val_accuracy: 0.4615
Epoch 81/600
120/120 [==============================] - ETA: 0s - loss: 2.4547 - accuracy: 0.4109
Epoch 81: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 208ms/step - loss: 2.4547 - accuracy: 0.4109 - val_loss: 2.4594 - ...
    val_accuracy: 0.4073
Epoch 82/600
120/120 [==============================] - ETA: 0s - loss: 2.5371 - accuracy: 0.4039
Epoch 82: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 208ms/step - loss: 2.5371 - accuracy: 0.4039 - val_loss: 2.3978 - ...
    val_accuracy: 0.4250
Epoch 83/600
120/120 [==============================] - ETA: 0s - loss: 2.2915 - accuracy: 0.4385
```

```
Epoch 83: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 207ms/step - loss: 2.2915 - accuracy: 0.4385 - val_loss: 2.0871 - ...
   val_accuracy: 0.5156
Epoch 84/600
120/120 [==============================] - ETA: 0s - loss: 2.3235 - accuracy: 0.4289
Epoch 84: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 209ms/step - loss: 2.3235 - accuracy: 0.4289 - val_loss: 2.1484 - ...
   val_accuracy: 0.4729
Epoch 85/600
120/120 [==============================] - ETA: 0s - loss: 2.2714 - accuracy: 0.4339
Epoch 85: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 207ms/step - loss: 2.2714 - accuracy: 0.4339 - val_loss: 2.2257 - ...
   val_accuracy: 0.4458
Epoch 86/600
120/120 [==============================] - ETA: 0s - loss: 2.1016 - accuracy: 0.4633
Epoch 86: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 209ms/step - loss: 2.1016 - accuracy: 0.4633 - val_loss: 1.9030 - ...
   val_accuracy: 0.5115
Epoch 87/600
120/120 [==============================] - ETA: 0s - loss: 1.9519 - accuracy: 0.4919
Epoch 87: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 208ms/step - loss: 1.9519 - accuracy: 0.4919 - val_loss: 1.8411 - ...
   val_accuracy: 0.5083
Epoch 88/600
120/120 [==============================] - ETA: 0s - loss: 1.9714 - accuracy: 0.4846
Epoch 88: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 207ms/step - loss: 1.9714 - accuracy: 0.4846 - val_loss: 1.8260 - ...
   val_accuracy: 0.5115
Epoch 89/600
120/120 [==============================] - ETA: 0s - loss: 1.8672 - accuracy: 0.5031
Epoch 89: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 208ms/step - loss: 1.8672 - accuracy: 0.5031 - val_loss: 1.7439 - ...
   val_accuracy: 0.5208
Epoch 90/600
120/120 [==============================] - ETA: 0s - loss: 1.7462 - accuracy: 0.5255
Epoch 90: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 211ms/step - loss: 1.7462 - accuracy: 0.5255 - val_loss: 1.7309 - ...
   val_accuracy: 0.5406
Epoch 91/600
```

```
120/120 [==============================] - ETA: 0s - loss: 1.8326 - accuracy: 0.4977
Epoch 91: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 209ms/step - loss: 1.8326 - accuracy: 0.4977 - val_loss: 1.6658 - ...
   val_accuracy: 0.5688
Epoch 92/600
120/120 [==============================] - ETA: 0s - loss: 1.7354 - accuracy: 0.5258
Epoch 92: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 209ms/step - loss: 1.7354 - accuracy: 0.5258 - val_loss: 1.6118 - ...
   val_accuracy: 0.5646
Epoch 93/600
120/120 [==============================] - ETA: 0s - loss: 1.7798 - accuracy: 0.4919
Epoch 93: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 209ms/step - loss: 1.7798 - accuracy: 0.4919 - val_loss: 1.6560 - ...
   val_accuracy: 0.5052
Epoch 94/600
120/120 [==============================] - ETA: 0s - loss: 2.2007 - accuracy: 0.4016
Epoch 94: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 208ms/step - loss: 2.2007 - accuracy: 0.4016 - val_loss: 1.8695 - ...
   val_accuracy: 0.5302
Epoch 95/600
120/120 [==============================] - ETA: 0s - loss: 1.8982 - accuracy: 0.4906
Epoch 95: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 208ms/step - loss: 1.8982 - accuracy: 0.4906 - val_loss: 1.6414 - ...
   val_accuracy: 0.5635
Epoch 96/600
120/120 [==============================] - ETA: 0s - loss: 1.7054 - accuracy: 0.5333
Epoch 96: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 208ms/step - loss: 1.7054 - accuracy: 0.5333 - val_loss: 1.5529 - ...
   val_accuracy: 0.5490
Epoch 97/600
120/120 [==============================] - ETA: 0s - loss: 1.5869 - accuracy: 0.5602
Epoch 97: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 208ms/step - loss: 1.5869 - accuracy: 0.5602 - val_loss: 1.4753 - ...
   val_accuracy: 0.5917
Epoch 98/600
120/120 [==============================] - ETA: 0s - loss: 1.5631 - accuracy: 0.5708
Epoch 98: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 208ms/step - loss: 1.5631 - accuracy: 0.5708 - val_loss: 1.4161 - ...
   val_accuracy: 0.5896
```

```
Epoch 99/600
120/120 [==============================] - ETA: 0s - loss: 1.4814 - accuracy: 0.5807
Epoch 99: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 208ms/step - loss: 1.4814 - accuracy: 0.5807 - val_loss: 1.4245 - ...
   val_accuracy: 0.5969
Epoch 100/600
120/120 [==============================] - ETA: 0s - loss: 1.5008 - accuracy: 0.5734
Epoch 100: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 208ms/step - loss: 1.5008 - accuracy: 0.5734 - val_loss: 1.4503 - ...
   val_accuracy: 0.5990
Epoch 101/600
120/120 [==============================] - ETA: 0s - loss: 1.5120 - accuracy: 0.5831
Epoch 101: val_accuracy did not improve from 0.62083
120/120 [==============================] - 25s 207ms/step - loss: 1.5120 - accuracy: 0.5831 - val_loss: 1.3537 - ...
   val_accuracy: 0.6198
Epoch 102/600
120/120 [==============================] - ETA: 0s - loss: 1.4013 - accuracy: 0.6049
Epoch 102: val_accuracy improved from 0.62083 to 0.62708, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 35s 294ms/step - loss: 1.4013 - accuracy: 0.6049 - val_loss: 1.2878 - ...
   val_accuracy: 0.6271
Epoch 103/600
120/120 [==============================] - ETA: 0s - loss: 1.5107 - accuracy: 0.5904
Epoch 103: val_accuracy did not improve from 0.62708
120/120 [==============================] - 25s 207ms/step - loss: 1.5107 - accuracy: 0.5904 - val_loss: 2.0667 - ...
   val_accuracy: 0.4938
Epoch 104/600
120/120 [==============================] - ETA: 0s - loss: 1.8633 - accuracy: 0.5320
Epoch 104: val_accuracy did not improve from 0.62708
120/120 [==============================] - 25s 208ms/step - loss: 1.8633 - accuracy: 0.5320 - val_loss: 1.5514 - ...
   val_accuracy: 0.6073
Epoch 105/600
120/120 [==============================] - ETA: 0s - loss: 1.8370 - accuracy: 0.5174
Epoch 105: val_accuracy did not improve from 0.62708
120/120 [==============================] - 25s 208ms/step - loss: 1.8370 - accuracy: 0.5174 - val_loss: 2.0931 - ...
   val_accuracy: 0.4563
Epoch 106/600
120/120 [==============================] - ETA: 0s - loss: 2.0552 - accuracy: 0.4242
Epoch 106: val_accuracy did not improve from 0.62708
120/120 [==============================] - 25s 209ms/step - loss: 2.0552 - accuracy: 0.4242 - val_loss: 1.8127 - ...
```

```
   val_accuracy: 0.5167
Epoch 107/600
120/120 [==============================] - ETA: 0s - loss: 1.7473 - accuracy: 0.5279
Epoch 107: val_accuracy did not improve from 0.62708
120/120 [==============================] - 25s 210ms/step - loss: 1.7473 - accuracy: 0.5279 - val_loss: 1.4828 - ...
   val_accuracy: 0.6083
Epoch 108/600
120/120 [==============================] - ETA: 0s - loss: 1.5065 - accuracy: 0.5909
Epoch 108: val_accuracy did not improve from 0.62708
120/120 [==============================] - 25s 209ms/step - loss: 1.5065 - accuracy: 0.5909 - val_loss: 1.4115 - ...
   val_accuracy: 0.6229
Epoch 109/600
120/120 [==============================] - ETA: 0s - loss: 1.5133 - accuracy: 0.5773
Epoch 109: val_accuracy did not improve from 0.62708
120/120 [==============================] - 25s 207ms/step - loss: 1.5133 - accuracy: 0.5773 - val_loss: 1.4605 - ...
   val_accuracy: 0.5542
Epoch 110/600
120/120 [==============================] - ETA: 0s - loss: 1.4448 - accuracy: 0.5898
Epoch 110: val_accuracy did not improve from 0.62708
120/120 [==============================] - 25s 206ms/step - loss: 1.4448 - accuracy: 0.5898 - val_loss: 1.2814 - ...
   val_accuracy: 0.6167
Epoch 111/600
120/120 [==============================] - ETA: 0s - loss: 1.3226 - accuracy: 0.6250
Epoch 111: val_accuracy improved from 0.62708 to 0.62917, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 37s 307ms/step - loss: 1.3226 - accuracy: 0.6250 - val_loss: 1.2512 - ...
   val_accuracy: 0.6292
Epoch 112/600
120/120 [==============================] - ETA: 0s - loss: 1.3070 - accuracy: 0.6214
Epoch 112: val_accuracy improved from 0.62917 to 0.63646, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 35s 296ms/step - loss: 1.3070 - accuracy: 0.6214 - val_loss: 1.2422 - ...
   val_accuracy: 0.6365
Epoch 113/600
120/120 [==============================] - ETA: 0s - loss: 1.2607 - accuracy: 0.6391
Epoch 113: val_accuracy improved from 0.63646 to 0.65521, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 37s 306ms/step - loss: 1.2607 - accuracy: 0.6391 - val_loss: 1.2009 - ...
   val_accuracy: 0.6552
Epoch 114/600
120/120 [==============================] - ETA: 0s - loss: 1.2749 - accuracy: 0.6193
Epoch 114: val_accuracy did not improve from 0.65521
```

```
120/120 [==============================] - 25s 207ms/step - loss: 1.2749 - accuracy: 0.6193 - val_loss: 1.3650 - ...
   val_accuracy: 0.6125
Epoch 115/600
120/120 [==============================] - ETA: 0s - loss: 1.2733 - accuracy: 0.6328
Epoch 115: val_accuracy did not improve from 0.65521
120/120 [==============================] - 25s 206ms/step - loss: 1.2733 - accuracy: 0.6328 - val_loss: 1.2586 - ...
   val_accuracy: 0.6208
Epoch 116/600
120/120 [==============================] - ETA: 0s - loss: 1.2487 - accuracy: 0.6339
Epoch 116: val_accuracy did not improve from 0.65521
120/120 [==============================] - 25s 208ms/step - loss: 1.2487 - accuracy: 0.6339 - val_loss: 1.2229 - ...
   val_accuracy: 0.6115
Epoch 117/600
120/120 [==============================] - ETA: 0s - loss: 1.2163 - accuracy: 0.6260
Epoch 117: val_accuracy improved from 0.65521 to 0.66667, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 35s 293ms/step - loss: 1.2163 - accuracy: 0.6260 - val_loss: 1.0975 - ...
   val_accuracy: 0.6667
Epoch 118/600
120/120 [==============================] - ETA: 0s - loss: 1.1952 - accuracy: 0.6427
Epoch 118: val_accuracy improved from 0.66667 to 0.67708, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 35s 292ms/step - loss: 1.1952 - accuracy: 0.6427 - val_loss: 1.0909 - ...
   val_accuracy: 0.6771
Epoch 119/600
120/120 [==============================] - ETA: 0s - loss: 1.3066 - accuracy: 0.6115
Epoch 119: val_accuracy did not improve from 0.67708
120/120 [==============================] - 25s 209ms/step - loss: 1.3066 - accuracy: 0.6115 - val_loss: 1.2893 - ...
   val_accuracy: 0.6385
Epoch 120/600
120/120 [==============================] - ETA: 0s - loss: 1.3002 - accuracy: 0.6333
Epoch 120: val_accuracy did not improve from 0.67708
120/120 [==============================] - 25s 208ms/step - loss: 1.3002 - accuracy: 0.6333 - val_loss: 1.1936 - ...
   val_accuracy: 0.6448
Epoch 121/600
120/120 [==============================] - ETA: 0s - loss: 1.3186 - accuracy: 0.6229
Epoch 121: val_accuracy did not improve from 0.67708
120/120 [==============================] - 25s 205ms/step - loss: 1.3186 - accuracy: 0.6229 - val_loss: 1.4394 - ...
   val_accuracy: 0.6115
Epoch 122/600
120/120 [==============================] - ETA: 0s - loss: 1.7620 - accuracy: 0.5130
```

```
Epoch 122: val_accuracy did not improve from 0.67708
120/120 [==============================] - 25s 204ms/step - loss: 1.7620 - accuracy: 0.5130 - val_loss: 1.4939 - ...
   val_accuracy: 0.5813
Epoch 123/600
120/120 [==============================] - ETA: 0s - loss: 1.4209 - accuracy: 0.5922
Epoch 123: val_accuracy did not improve from 0.67708
120/120 [==============================] - 24s 204ms/step - loss: 1.4209 - accuracy: 0.5922 - val_loss: 1.2970 - ...
   val_accuracy: 0.6573
Epoch 124/600
120/120 [==============================] - ETA: 0s - loss: 1.4180 - accuracy: 0.6062
Epoch 124: val_accuracy did not improve from 0.67708
120/120 [==============================] - 25s 204ms/step - loss: 1.4180 - accuracy: 0.6062 - val_loss: 1.3699 - ...
   val_accuracy: 0.6125
Epoch 125/600
120/120 [==============================] - ETA: 0s - loss: 1.2905 - accuracy: 0.6323
Epoch 125: val_accuracy did not improve from 0.67708
120/120 [==============================] - 24s 204ms/step - loss: 1.2905 - accuracy: 0.6323 - val_loss: 1.1537 - ...
   val_accuracy: 0.6677
Epoch 126/600
120/120 [==============================] - ETA: 0s - loss: 1.2980 - accuracy: 0.6427
Epoch 126: val_accuracy did not improve from 0.67708
120/120 [==============================] - 25s 206ms/step - loss: 1.2980 - accuracy: 0.6427 - val_loss: 1.1319 - ...
   val_accuracy: 0.6760
Epoch 127/600
120/120 [==============================] - ETA: 0s - loss: 1.1970 - accuracy: 0.6578
Epoch 127: val_accuracy improved from 0.67708 to 0.69271, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 36s 303ms/step - loss: 1.1970 - accuracy: 0.6578 - val_loss: 1.1071 - ...
   val_accuracy: 0.6927
Epoch 128/600
120/120 [==============================] - ETA: 0s - loss: 1.1518 - accuracy: 0.6698
Epoch 128: val_accuracy improved from 0.69271 to 0.72500, saving model to Pure_LSTM_model.tf
120/120 [==============================] - 35s 290ms/step - loss: 1.1518 - accuracy: 0.6698 - val_loss: 1.0400 - ...
   val_accuracy: 0.7250
Epoch 129/600
120/120 [==============================] - ETA: 0s - loss: 1.1661 - accuracy: 0.6672
Epoch 129: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 204ms/step - loss: 1.1661 - accuracy: 0.6672 - val_loss: 1.2459 - ...
   val_accuracy: 0.6448
Epoch 130/600
```

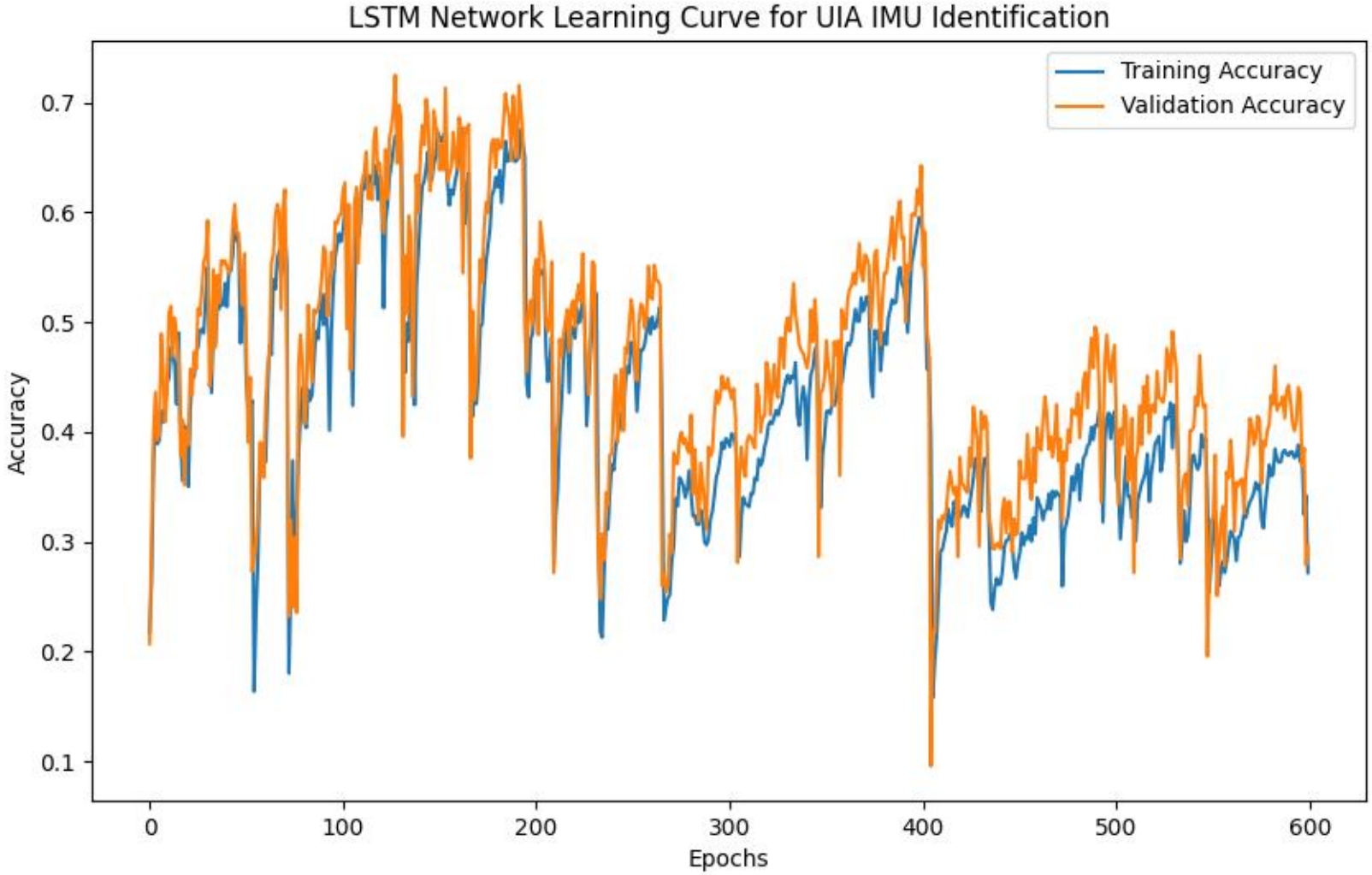
```
120/120 [==============================] - ETA: 0s - loss: 1.1738 - accuracy: 0.6682
Epoch 130: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 205ms/step - loss: 1.1738 - accuracy: 0.6682 - val_loss: 1.1022 - ...
   val_accuracy: 0.6979
Epoch 131/600
120/120 [==============================] - ETA: 0s - loss: 1.2441 - accuracy: 0.6661
Epoch 131: val_accuracy did not improve from 0.72500

...........

120/120 [==============================] - 25s 206ms/step - loss: 2.2773 - accuracy: 0.5555 - val_loss: 2.0364 - ...
   val_accuracy: 0.6094
Epoch 176/600
120/120 [==============================] - ETA: 0s - loss: 2.0949 - accuracy: 0.5698
Epoch 176: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 209ms/step - loss: 2.0949 - accuracy: 0.5698 - val_loss: 1.9862 - ...
   val_accuracy: 0.6000
Epoch 177/600
120/120 [==============================] - ETA: 0s - loss: 1.9823 - accuracy: 0.5849
Epoch 177: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 210ms/step - loss: 1.9823 - accuracy: 0.5849 - val_loss: 1.7815 - ...
   val_accuracy: 0.6438
Epoch 178/600
120/120 [==============================] - ETA: 0s - loss: 1.8434 - accuracy: 0.6156
Epoch 178: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 208ms/step - loss: 1.8434 - accuracy: 0.6156 - val_loss: 1.7091 - ...
   val_accuracy: 0.6646
Epoch 179/600
120/120 [==============================] - ETA: 0s - loss: 1.7554 - accuracy: 0.6201
Epoch 179: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 209ms/step - loss: 1.7554 - accuracy: 0.6201 - val_loss: 1.6109 - ...
   val_accuracy: 0.6667
Epoch 180/600
120/120 [==============================] - ETA: 0s - loss: 1.6739 - accuracy: 0.6320
Epoch 180: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 209ms/step - loss: 1.6739 - accuracy: 0.6320 - val_loss: 1.5731 - ...
   val_accuracy: 0.6406
Epoch 181/600
120/120 [==============================] - ETA: 0s - loss: 1.6446 - accuracy: 0.6232
```

```
Epoch 181: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 211ms/step - loss: 1.6446 - accuracy: 0.6232 - val_loss: 1.5536 - ...
   val_accuracy: 0.6667
Epoch 182/600
120/120 [==============================] - ETA: 0s - loss: 1.6305 - accuracy: 0.6383
Epoch 182: val_accuracy did not improve from 0.72500
120/120 [==============================] - 28s 231ms/step - loss: 1.6305 - accuracy: 0.6383 - val_loss: 1.5036 - ...
   val_accuracy: 0.6583
Epoch 183/600
120/120 [==============================] - ETA: 0s - loss: 1.7100 - accuracy: 0.6089
Epoch 183: val_accuracy did not improve from 0.72500
120/120 [==============================] - 26s 214ms/step - loss: 1.7100 - accuracy: 0.6089 - val_loss: 1.5707 - ...
   val_accuracy: 0.6594
Epoch 184/600
120/120 [==============================] - ETA: 0s - loss: 1.5624 - accuracy: 0.6339
Epoch 184: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 208ms/step - loss: 1.5624 - accuracy: 0.6339 - val_loss: 1.4031 - ...
   val_accuracy: 0.6823
Epoch 185/600
120/120 [==============================] - ETA: 0s - loss: 1.4681 - accuracy: 0.6648
Epoch 185: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 208ms/step - loss: 1.4681 - accuracy: 0.6648 - val_loss: 1.3519 - ...
   val_accuracy: 0.7083
Epoch 186/600
120/120 [==============================] - ETA: 0s - loss: 1.5278 - accuracy: 0.6464
Epoch 186: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 207ms/step - loss: 1.5278 - accuracy: 0.6464 - val_loss: 1.4004 - ...
   val_accuracy: 0.6948
Epoch 187/600
120/120 [==============================] - ETA: 0s - loss: 1.4776 - accuracy: 0.6477
Epoch 187: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 209ms/step - loss: 1.4776 - accuracy: 0.6477 - val_loss: 1.3486 - ...
   val_accuracy: 0.6875
Epoch 188/600
120/120 [==============================] - ETA: 0s - loss: 1.4068 - accuracy: 0.6607
Epoch 188: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 208ms/step - loss: 1.4068 - accuracy: 0.6607 - val_loss: 1.4287 - ...
   val_accuracy: 0.6469
Epoch 189/600
```

```
120/120 [==============================] - ETA: 0s - loss: 1.4745 - accuracy: 0.6505
Epoch 189: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 212ms/step - loss: 1.4745 - accuracy: 0.6505 - val_loss: 1.3626 - ...
   val_accuracy: 0.7063
Epoch 190/600
120/120 [==============================] - ETA: 0s - loss: 1.4803 - accuracy: 0.6461
Epoch 190: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 207ms/step - loss: 1.4803 - accuracy: 0.6461 - val_loss: 1.4271 - ...
   val_accuracy: 0.6750
Epoch 191/600
120/120 [==============================] - ETA: 0s - loss: 1.5310 - accuracy: 0.6479
Epoch 191: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 208ms/step - loss: 1.5310 - accuracy: 0.6479 - val_loss: 1.5216 - ...
   val_accuracy: 0.6500
Epoch 192/600
120/120 [==============================] - ETA: 0s - loss: 1.4377 - accuracy: 0.6500
Epoch 192: val_accuracy did not improve from 0.72500

.......................

Epoch 594/600
120/120 [==============================] - ETA: 0s - loss: 23.1833 - accuracy: 0.3784
Epoch 594: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 209ms/step - loss: 23.1833 - accuracy: 0.3784 - val_loss: 23.0323 - ...
   val_accuracy: 0.4094
Epoch 595/600
120/120 [==============================] - ETA: 0s - loss: 23.0687 - accuracy: 0.3883
Epoch 595: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 208ms/step - loss: 23.0687 - accuracy: 0.3883 - val_loss: 22.8137 - ...
   val_accuracy: 0.4406
Epoch 596/600
120/120 [==============================] - ETA: 0s - loss: 22.8781 - accuracy: 0.3729
Epoch 596: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 206ms/step - loss: 22.8781 - accuracy: 0.3729 - val_loss: 22.5901 - ...
   val_accuracy: 0.4354
Epoch 597/600
120/120 [==============================] - ETA: 0s - loss: 22.6803 - accuracy: 0.3786
Epoch 597: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 207ms/step - loss: 22.6803 - accuracy: 0.3786 - val_loss: 22.7672 - ...
```

```
   val_accuracy: 0.3688
Epoch 598/600
120/120 [==============================] - ETA: 0s - loss: 23.0878 - accuracy: 0.3247
Epoch 598: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 209ms/step - loss: 23.0878 - accuracy: 0.3247 - val_loss: 22.9770 - ...
   val_accuracy: 0.3854
Epoch 599/600
120/120 [==============================] - ETA: 0s - loss: 23.2130 - accuracy: 0.3422
Epoch 599: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 206ms/step - loss: 23.2130 - accuracy: 0.3422 - val_loss: 23.9470 - ...
   val_accuracy: 0.2792
Epoch 600/600
120/120 [==============================] - ETA: 0s - loss: 24.1886 - accuracy: 0.2719
Epoch 600: val_accuracy did not improve from 0.72500
120/120 [==============================] - 25s 208ms/step - loss: 24.1886 - accuracy: 0.2719 - val_loss: 24.0651 - ...
   val_accuracy: 0.2958
```

```
    30/30 [==============================] - 2s 42ms/step - loss: 1.0400 - accuracy: 0.7250
Test Loss: 1.0399501323699951, Test Accuracy: 0.7250000238418579
30/30 [==============================] - 2s 39ms/step
<tf.Tensor: shape=(13, 13), dtype=int32, numpy=
array([[100,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,  64,   0,   0,   0,   2,   0,   9,   3,   0,   0,   0,   0],
       [  0,   7,  45,   0,   1,   0,   0,   9,   8,   0,   0,   0,   0],
       [  0,   8,   2,  50,   0,   6,   0,   0,   0,   0,   0,   0,   0],
       [  2,   0,   0,  11,  46,   1,   0,   0,   0,   6,   1,   0,   0],
       [  0,   7,   0,   3,   0,  52,   0,   1,   1,   0,   0,   4,   0],
       [  0,   0,   0,   1,   0,   0,  72,   0,   0,   0,   2,   0,   0],
       [  0,   2,   2,   0,   0,   0,   0,  21,  43,   0,   0,   0,   0],
       [  0,   1,   0,   0,   0,   0,   0,   2,  67,   0,   0,   0,   0],
       [  0,   1,   2,  28,  37,   3,   0,   0,   0,   5,   3,   1,   0],
       [  0,   3,   0,   1,   2,   6,   0,   0,   0,  10,  35,   5,   0],
       [  3,   6,   0,   1,   0,   1,   1,   0,   0,   2,   2,  58,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   1,  81]],
      dtype=int32)>
```


LSTM Network Learning Curve for UIA IMU Identification
Training Accuracy
Validation Accuracy
Accuracy
Epochs
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
100
200
300
400
500
600

UIA IMU: Max F1-Averaged Stratified k-Fold validation of GI models

Simplified TCN model validation

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```



```
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_ID_Walking_Gait_Dataset_8_1_2024
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_IMU_9ax_WG_Dataset_W_Calibration_Data_U_21_Des_23.csv
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_IMU_9ax_WG_Dataset_U_19_Des_23
/kaggle/input/tcn-trained-23/kaggle/working/TCN_H_Format.h5
/kaggle/input/a-c-lstm-uia-imu-id/Attentive_Conv_LSTM_model_24_1.h5
/kaggle/input/h-model-tcn-1/TCN_Model_HH.h5
/kaggle/input/cnn-uia-imu-id-25-1/Strict_CNN_model_24_1.h5
```

```
df = pd.read_csv('/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_ID_Walking_Gait_Dataset_8_1_2024')
```

Running through preparation very quickly, this is all a repeat from the code where the models are trained

```
Feats = ['Pitch', 'Yaw', 'q1', 'magX', 'q2', 'magZ']


df['Gait_Identification'] = df['Subject_no']

Label = ['Gait_Identification']
```

```
Columns = ['Gait_Identification', 'Pitch', 'Yaw', 'q1', 'magX', 'q2', 'magZ']

df = df[Columns]

df
```

```
       Gait_Identification     Pitch         Yaw        q1  magX        q2  \
0                        0 -0.349617   -0.606688 -0.005295    72 -0.003050
1                        0 -0.170067   -0.916112 -0.007995    73 -0.001483
2                        0 -0.009767   -1.136978 -0.009922    72 -0.000083
3                        0  0.094222   -1.417035 -0.012365    72  0.000831
4                        0  0.058380   -1.775726 -0.015495    72  0.000539
...                    ...       ...         ...       ...   ...       ...
38754                   12  3.458979 -337.283108 -0.198041    56 -0.020191
38755                   12  1.243237 -337.540503 -0.194980    55 -0.004102
38756                   12  0.126920 -338.566529 -0.185914    55  0.004001
38757                   12  0.067100 -339.795425 -0.175340    56  0.004847
38758                   12  0.608490 -340.818421 -0.166676    56  0.002149

       magZ
0         7
1         7
2         8
3         7
4         5
...     ...
38754    41
38755    40
38756    41
38757    39
38758    36

[38759 rows x 7 columns]
```

```
df_ID = df.copy()
```

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler


sequence_length = 32
overlap_percentage = 0.70


X_columns = Feats
y_column = Label

# Extrapolate features
X = df[X_columns].values


# Scale
scaler = StandardScaler()
df[X_columns] = scaler.fit_transform(X)


# Calculate the overlap and step size
overlap_size = int(sequence_length * overlap_percentage)
step_size = sequence_length - overlap_size

# Initialize lists to store sequences and labels
sequences = []
labels = []

# Iterate through the dataframe to create sequences
for i in range(0, len(df) - sequence_length + 1, step_size):
    sequence = df[X_columns].values[i:i + sequence_length]
    label_values = tuple(tuple(row) for row in df[y_column].values[i:i + sequence_length])  # Convert nested arrays to tuples
```

```
    '''

    Only include sequences with uniform label value, i.e.
    drop data in transition from one subject to another.
    Several users for same prediction is not a real-world
    scenario.

    '''

    if len(set(label_values)) == 1:
        label = label_values[0]
        sequences.append(sequence)
        labels.append(label)

# Convert lists to numpy arrays
X = np.array(sequences)
y = np.array(labels)
```


Running combined k-fold cross-validation and F1 scoring for all fold. End result is an average F1 score for all folds, this robust testing scheme should give a realistic validation of model performance.

The costum layer in the model is causing some trouble when saving, and loading the model.

This was solved by first builidng the model from scratch, and then loading the saved file from the training session.

```
from keras.layers import Multiply
from keras.models import Sequential
from keras.layers import Conv1D, Dense, Activation, Input, Reshape, Lambda, concatenate, Layer
from keras.layers import Add, Dropout, BatchNormalization, GlobalAveragePooling1D
from keras.optimizers import Adam

class ResidualBlock(Layer):
    def __init__(self, dilation_rate, nb_filters, kernel_size, padding, dropout_rate, **kwargs):
        super(ResidualBlock, self).__init__(**kwargs)
        self.dilation_rate = dilation_rate
```

```
        self.nb_filters = nb_filters
        self.kernel_size = kernel_size
        self.padding = padding
        self.dropout_rate = dropout_rate

    def build(self, input_shape):
        super(ResidualBlock, self).build(input_shape)
        self.tanh_conv = Conv1D(filters=self.nb_filters, kernel_size=self.kernel_size, dilation_rate=self.dilation_rate,
                                padding=self.padding, activation='tanh')
        self.sigm_conv = Conv1D(filters=self.nb_filters, kernel_size=self.kernel_size, dilation_rate=self.dilation_rate,
                                padding=self.padding, activation='sigmoid')
        self.multiply = Multiply()
        self.one_by_one_conv = Conv1D(filters=self.nb_filters, kernel_size=1, padding='same')
        self.add_layer = Add()
        self.activation = Activation('relu')

    def call(self, x):
        prev_x = x
        tanh_out = self.tanh_conv(x)
        sigm_out = self.sigm_conv(x)
        x = self.multiply([tanh_out, sigm_out])
        x = self.one_by_one_conv(x)
        res_x = self.add_layer([prev_x, x])
        activation = self.activation(res_x)
        return activation


    # For saving model in Keras:
    def get_config(self):
        config = super(ResidualBlock, self).get_config()
        config.update({
            'dilation_rate': self.dilation_rate,
            'nb_filters': self.nb_filters,
            'kernel_size': self.kernel_size,
```

```
            'padding': self.padding,
            'dropout_rate': self.dropout_rate
        })
        return config

def build_tcn_model(input_shape, nb_filters, kernel_size, dilations, nb_stacks, dropout_rate=0.0, output_dim=30):
    model = Sequential()
    model.add(Conv1D(filters=nb_filters, kernel_size=1, padding='same', input_shape=input_shape))
    for _ in range(nb_stacks):
        for dilation_rate in dilations:
            model.add(ResidualBlock(dilation_rate=dilation_rate,
                                    nb_filters=nb_filters, kernel_size=kernel_size,
                                    padding='causal', dropout_rate=dropout_rate))
    model.add(GlobalAveragePooling1D())
    model.add(Dense(units=output_dim, activation='softmax'))
    model.compile(optimizer=Adam(lr=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model

input_dim = X.shape[2]
sequence_length = X.shape[1]
input_shape = (sequence_length, input_dim)
output_dim = len(np.unique(y))
nb_filters = 36
kernel_size = 3
dilations = [1,2,4,5,6]
nb_stacks = 3
dropout_rate = 0.7

TCN_Model = build_tcn_model(input_shape, nb_filters, kernel_size, dilations, nb_stacks, dropout_rate, output_dim)
TCN_Model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

```
conv1d (Conv1D)              (None, 32, 36)            252

residual_block (ResidualBl   (None, 32, 36)            9180
ock)

residual_block_1 (Residual   (None, 32, 36)            9180
Block)

residual_block_2 (Residual   (None, 32, 36)            9180
Block)

residual_block_3 (Residual   (None, 32, 36)            9180
Block)

residual_block_4 (Residual   (None, 32, 36)            9180
Block)

residual_block_5 (Residual   (None, 32, 36)            9180
Block)

residual_block_6 (Residual   (None, 32, 36)            9180
Block)

residual_block_7 (Residual   (None, 32, 36)            9180
Block)

residual_block_8 (Residual   (None, 32, 36)            9180
Block)

residual_block_9 (Residual   (None, 32, 36)            9180
Block)

residual_block_10 (Residua   (None, 32, 36)            9180
lBlock)
```

```
 residual_block_11 (Residua  (None, 32, 36)            9180
 lBlock)

 residual_block_12 (Residua  (None, 32, 36)            9180
 lBlock)

 residual_block_13 (Residua  (None, 32, 36)            9180
 lBlock)

 residual_block_14 (Residua  (None, 32, 36)            9180
 lBlock)

 global_average_pooling1d (  (None, 36)                0
 GlobalAveragePooling1D)

 dense (Dense)               (None, 13)                481

=================================================================
Total params: 138433 (540.75 KB)
Trainable params: 138433 (540.75 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```python
from keras.models import load_model

# Define the custom object
custom_objects = {'ResidualBlock': ResidualBlock}

# Load the model
TCN_Model = load_model('/kaggle/input/h-model-tcn-1/TCN_Model_HH.h5', custom_objects=custom_objects)

from keras.models import Sequential
from keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Dropout, BatchNormalization
from keras.optimizers import RMSprop
```

```
from keras.regularizers import l2
from keras.callbacks import EarlyStopping

#Variables:

input_dim = X.shape[2]
sequence_length = X.shape[1]
input_shape = (sequence_length, input_dim)
output_dim = len(np.unique(y))
nb_filters = 64
kernel_size = 3
dilations = [1, 2, 3, 6]
nb_stacks = 3
dropout_rate = 0.2

#model-specific variables:

#Dense layer, to exand num combinations
Units_Dens_Expand_Dim = 80

Units_Unity_Connected_Layer = 64

#Unity_Conected_Layer = Conv

#Conv layer, create new abstractions from features
Filters = 100

Kernel_Size = 2


# Pooling
Pool_Size = 2
```

```
#LSTM, learn patterns in data
LSTM_Units = 64

Recurrent_Dropout = 0.2




# dropout, prevent overfitting
DropOut_Rate = 0.4

from keras.layers import Input, Conv1D, LSTM, Concatenate, Flatten, Layer
from tensorflow.keras.models import Model
import tensorflow as tf

class MyAttention(Layer):
    def __init__(self, **kwargs):
        super(MyAttention, self).__init__(**kwargs)

    def build(self, input_shape):
        sequence_dim = input_shape[1] if input_shape[1] is not None else 1
        feature_dim = input_shape[-1] if input_shape[-1] is not None else 1
        initializer = tf.keras.initializers.GlorotUniform(seed=None)  # You can choose a different initializer if needed
        self.kernel = self.add_weight(
            name='kernel',
            shape=(feature_dim, 1),
            initializer=initializer,
            trainable=True
        )
        super(MyAttention, self).build(input_shape)

    def call(self, x):
        scores = tf.matmul(x, self.kernel)
        attention_weights = tf.nn.softmax(scores, axis=1)
        attended_output = x * attention_weights
```

```
        return attended_output


# Input layer
In_Layer = Input(shape=input_shape)

# Convolutional layer

C_Layer = Conv1D(data_format= 'channels_last', padding= 'same', filters = Filters, kernel_size = Kernel_Size)(In_Layer)

C_Layer_2 = Conv1D(data_format= 'channels_last', padding= 'same', filters = Filters*2, kernel_size = Kernel_Size)(C_Layer)

C_Layer_3 = Conv1D(data_format= 'channels_last', padding= 'same', filters = Filters*2, kernel_size = Kernel_Size)(C_Layer_2)

Max_Pool = MaxPooling1D(data_format='channels_first',
                            padding= 'same',
                            pool_size=4)(C_Layer_3)

dropout_layer = Dropout(rate=DropOut_Rate)(Max_Pool)

Unity_Connected_Layer = Conv1D(filters=Units_Unity_Connected_Layer,
                             kernel_size=1,
                             use_bias=False,
                            data_format='channels_last')(dropout_layer)

# LSTM layer
lstm_layer_ = LSTM(units=LSTM_Units, return_sequences=True, activation='tanh',
                   recurrent_activation='sigmoid', recurrent_dropout=Recurrent_Dropout,
                   unroll=False, use_bias=True, return_state=False)(Unity_Connected_Layer)
'''
lstm_layer_1 = LSTM(units=LSTM_Units, return_sequences=True, activation='tanh',
                   recurrent_activation='sigmoid', recurrent_dropout=Recurrent_Dropout,
                   unroll=False, use_bias=True, return_state=False)(lstm_layer_)
'''
```

```
attention_layer = MyAttention()

attended_output = attention_layer(lstm_layer_)


# Dropout layer
dropout_layer_2 = Dropout(rate=DropOut_Rate)(attended_output)


# LSTM layer
lstm_layer_1 = LSTM(units=LSTM_Units, return_sequences=True, activation='tanh',
                    recurrent_activation='sigmoid', recurrent_dropout=Recurrent_Dropout,
                    unroll=False, use_bias=True, return_state=False)(dropout_layer_2)
'''
lstm_layer_1 = LSTM(units=LSTM_Units, return_sequences=True, activation='tanh',
                    recurrent_activation='sigmoid', recurrent_dropout=Recurrent_Dropout,
                    unroll=False, use_bias=True, return_state=False)(lstm_layer_)
'''

attention_layer_1 = MyAttention()

attended_output_1 = attention_layer_1(lstm_layer_1)


# Dropout layer
dropout_layer_2 = Dropout(rate=DropOut_Rate)(attended_output_1)


# Flatten layer
flat_layer = Flatten()(dropout_layer_2)

# Output layer
OUT = Dense(output_dim, activation='softmax')(flat_layer)
```

```
# Build the Attentive_Conv_LSTM
Attentive_Conv_LSTM_model = Model(inputs=In_Layer, outputs=OUT)



# Compile the Attentive_Conv_LSTM
Attentive_Conv_LSTM_model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

from keras.models import load_model

# Define the custom object
custom_objects = {'MyAttention': MyAttention}

# Load the model
#TCN_Model = load_model('/kaggle/input/h-model-tcn-1/TCN_Model_HH.h5', custom_objects=custom_objects)

A_C_LSTM_Model = load_model('/kaggle/input/a-c-lstm-uia-imu-id/Attentive_Conv_LSTM_model_24_1.h5', custom_objects=custom_objects)

CNN_Model = load_model('/kaggle/input/cnn-uia-imu-id-25-1/Strict_CNN_model_24_1.h5')

CNN_Model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 32, 32)            224

 conv1d_1 (Conv1D)           (None, 32, 32)            2080
```

```
max_pooling1d (MaxPooling1  (None, 32, 16)           0
D)

batch_normalization (Batch  (None, 32, 16)           64
Normalization)

conv1d_2 (Conv1D)           (None, 32, 64)           2112

max_pooling1d_1 (MaxPoolin  (None, 32, 32)           0
g1D)

batch_normalization_1 (Bat  (None, 32, 32)           128
chNormalization)

conv1d_3 (Conv1D)           (None, 32, 64)           4160

max_pooling1d_2 (MaxPoolin  (None, 32, 32)           0
g1D)

batch_normalization_2 (Bat  (None, 32, 32)           128
chNormalization)

conv1d_4 (Conv1D)           (None, 32, 64)           4160

max_pooling1d_3 (MaxPoolin  (None, 32, 32)           0
g1D)

batch_normalization_3 (Bat  (None, 32, 32)           128
chNormalization)

conv1d_5 (Conv1D)           (None, 32, 64)           4160

max_pooling1d_4 (MaxPoolin  (None, 32, 32)           0
g1D)
```

```
 batch_normalization_4 (Bat  (None, 32, 32)            128
 chNormalization)

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 256)               262400

 dropout (Dropout)           (None, 256)               0

 batch_normalization_5 (Bat  (None, 256)               1024
 chNormalization)

 dense_1 (Dense)             (None, 13)                3341

=================================================================
Total params: 284237 (1.08 MB)
Trainable params: 283437 (1.08 MB)
Non-trainable params: 800 (3.12 KB)
_________________________________________________________________


test_loss, test_accuracy = TCN_Model.evaluate(X, y)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')

test_loss, test_accuracy = A_C_LSTM_Model.evaluate(X, y)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')

test_loss, test_accuracy = CNN_Model.evaluate(X, y)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')
```

```
120/120 [==============================] - 3s 7ms/step - loss: 0.0014 - accuracy: 1.0000
Test Loss: 0.0013765807962045074, Test Accuracy: 1.0
120/120 [==============================] - 3s 19ms/step - loss: 0.0726 - accuracy: 0.9755
Test Loss: 0.07255330681800842, Test Accuracy: 0.9754825234413147
```

```
120/120 [==============================] - 1s 4ms/step - loss: 0.5121 - accuracy: 0.9679
Test Loss: 0.5121251344680786, Test Accuracy: 0.9679186344146729
```

All three models loaded with weights from training session.

Load the model eval kit, declaring the model_metrics dictionary:

```
#The combined Average F1 - K-fold cross validation, Model Validation kit by Martin B Gresli. Made in 2024

### PS, only run once, as you will reset the dictonary holding model metrics if run again.

from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import f1_score
from tensorflow.keras.models import Sequential
from keras.optimizers import Adam
import tensorflow as tf


import matplotlib.pyplot as plt
import matplotlib.colors as colors
import numpy as np
import tensorflow as tf
import os

def plot_training_history(training_history, title, filename):

    import matplotlib.pyplot as plt
    fig, ax1 = plt.subplots()

    # Plot training and validation loss on the left y-axis
    ax1.plot(training_history.history['loss'], 'b--', alpha=0.3, label='Training Loss')
    ax1.plot(training_history.history['val_loss'], 'b-', label='Validation Loss')
    ax1.set_xlabel('Epochs')
    ax1.set_ylabel('Loss', color='b')
    ax1.tick_params(axis='y', labelcolor='b')
```

```
    # Create a second y-axis
    ax2 = ax1.twinx()

    # Plot training and validation accuracy on the right y-axis
    ax2.plot(training_history.history['accuracy'], 'r--', alpha=0.3, label='Training Accuracy')
    ax2.plot(training_history.history['val_accuracy'], 'r-', label='Validation Accuracy')
    ax2.set_ylabel('Accuracy', color='r')
    ax2.tick_params(axis='y', labelcolor='r')


    lines, labels = ax1.get_legend_handles_labels()
    lines2, labels2 = ax2.get_legend_handles_labels()
    ax2.legend(lines + lines2, labels + labels2, loc='best')

    # Set the title
    plt.title(f'Training History for {title}')

    # Save the plot as a file
    fig.savefig(f'{filename}.eps', format='eps')
    plt.show()
    plt.close()

    import matplotlib.pyplot as plt
    import numpy as np

def save_f1_scores_grouped_as_eps(model_metrics, filename='f1_scores_grouped_performance.eps'):

    import matplotlib.pyplot as plt

    model_names = list(model_metrics.keys())
    n_models = len(model_names)
    n_folds = len(model_metrics[model_names[0]]['F1_Scores'])  # Assuming each model has the same number of folds

    # Data preparation
```

```
f1_scores = np.array([model_metrics[model_name]['F1_Scores'] for model_name in model_names]).T  # Transposed for grouping
average_f1_scores = [model_metrics[model_name]['Average_F1'] for model_name in model_names]

fig, ax = plt.subplots(figsize=(10 + n_models, 6))  # Dynamic width based on number of models

# Calculate the width of each bar and the positions for the groups
bar_width = 0.8 / n_folds  # The total width for each group is 0.8, leaving some space between groups
indices = np.arange(n_models)

# Plot each fold's F1 scores
for i in range(n_folds):
    ax.bar(indices - 0.4 + i * bar_width, f1_scores[i], bar_width, label=f'Fold {i+1}')

# Plot the average F1 score
ax.bar(indices + 0.4, average_f1_scores, bar_width, label='Average', color='skyblue', edgecolor='black')

# Add model names to the x-axis
ax.set_xticks(indices)
ax.set_xticklabels(model_names, rotation=45)
ax.set_ylim(0.9, 1)


# Set labels and title
ax.set_xlabel('Model Name')
ax.set_ylabel('F1 Score')
ax.set_title('F1 Scores per Fold and Average F1 Score of Models')

# Add a legend
ax.legend(loc='lower right')

# Save the plot as a file
plt.tight_layout()
plt.savefig(filename, format='eps')
plt.savefig(filename.replace('.eps', '.jpeg'), format='jpeg')
```

```
    plt.show()
    plt.close()

def save_f1_scores_table_as_eps(model_metrics, filename='f1_scores_table.eps'):
    # Prepare data for the table
    columns = ['Model Name'] + [f'Fold {i+1}' for i in range(len(next(iter(model_metrics.values()))['F1_Scores']))] + ['Average F1']
    cell_text = []

    for model_name, metrics in model_metrics.items():
        row = [model_name] + metrics['F1_Scores'] + [metrics['Average_F1']]
        cell_text.append(row)



    column_width = 3
    table_width = column_width * (len(columns))
    table_height = 0.1 * len(cell_text)


    fig, ax = plt.subplots(figsize=(table_width, table_height))
    ax.axis('off')


    the_table = ax.table(cellText=cell_text, colLabels=columns, loc='center', cellLoc='center', colLoc='center')

    for cell in the_table.get_celld().values():
        if cell.row == 0:
            cell.set_fontsize(12)
            cell.set_text_props(weight='bold')


    fig.tight_layout()

    # Save the plot as a file with tight bounding box
```

```
    plt.savefig(filename, format='eps', bbox_inches='tight')
    plt.savefig(filename.replace('.eps', '.jpeg'), format='jpeg', bbox_inches='tight')
    plt.show()
    plt.close()




def plot_confusion_matrix_heatmap(model, model_name, X, y, title, filename_prop):




    predict = model.predict(X)
    predictions = np.argmax(predict, axis=1)

    output_dim = len(np.unique(y))

    confusion_matrix = tf.math.confusion_matrix(labels=y,
                                                predictions=predictions,
                                                num_classes=output_dim)

# Calculate the correct prediction rate
    total_predictions_per_class = np.sum(confusion_matrix, axis=1)  # Sum over rows for total predictions
    correct_predictions = np.diag(confusion_matrix)
    correct_prediction_rate = correct_predictions / total_predictions_per_class.astype(float)

    # Create a matrix to hold the values for coloring the heatmap
    heatmap_data = np.full_like(confusion_matrix, np.nan, dtype=float)  # Fill with NaN for off-diagonal
    np.fill_diagonal(heatmap_data, correct_prediction_rate)  # Set the diagonal with correct prediction rate
```

```
# Create a colormap that is light blue for NaN and dark blue for high correct prediction rate
cmap = colors.LinearSegmentedColormap.from_list(
    'custom blue',
    [(0, 'lightblue'), (0.5, 'skyblue'), (0.99, 'blue'), (1, 'darkblue')]
)
cmap.set_bad('lightgrey', 1.0)  # Color for NaN values

# Plotting the heatmap
fig, ax = plt.subplots()
cax = ax.matshow(heatmap_data, cmap=cmap, vmin=0, vmax=1)

# Add colorbar
plt.colorbar(cax)

# Annotate all cells with the actual count, but color the text differently for diagonal vs. off-diagonal
for (i, j), val in np.ndenumerate(confusion_matrix):
    if i == j:  # Diagonal: correct predictions
        text_color = 'white' if correct_prediction_rate[i] > 0.5 else 'black'
    else:  # Off-diagonal: incorrect predictions
        text_color = 'grey'
    ax.text(j, i, f'{val}', ha='center', va='center', color=text_color)

plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title(f'{title}')
plt.savefig(f'{filename_prop}.eps', format='eps')

plt.show()

plt.close()
```

```
# plot_training_history(training_history, 'training_history')
# plot_confusion_matrix_heatmap(confusion_matrix, 'confusion_matrix_heatmap')

#### Create dict to store F1 score validation results as we manually iterate through the models


model_metrics = {}




def validate_and_store_metrics(model_list, X, y, k=5, learning_rate=0.01, epochs=15, batch_size=32):
    # Ensure model_metrics is accessible within this function
    global model_metrics

    for model, model_name, custom_object in model_list:

        # Initialize StratifiedKFold
        skf = StratifiedKFold(n_splits=k, shuffle=True, random_state=55)

        # Initialize a list to store F1 scores for each fold
        f1_scores = []

        j = 0
        for train_index, test_index in skf.split(X, y):
            # Split data
            X_train, X_test = X[train_index], X[test_index]
            y_train, y_test = y[train_index], y[test_index]

            # Compile and fit the model
            model.compile(optimizer=Adam(lr=learning_rate), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

            from keras.callbacks import ModelCheckpoint
```

```
checkpoint_filepath =  os.path.join('/kaggle/working/', f'{model_name}_Fold_{j}_Checkpoint.h5')

best_validation_callback = ModelCheckpoint(
    checkpoint_filepath,
    monitor='val_accuracy',
    verbose=1,
    save_best_only=True,
    mode='max',
    save_format='h5'
)


history = model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, validation_data=(X_test, y_test), callbacks=[best_


'''

if os.path.exists(checkpoint_filepath):
    print(f"Checkpoint file created: {checkpoint_filepath}")
    model_loaded = load_model(checkpoint_filepath)
else:
    print(f"Checkpoint file not found at: {checkpoint_filepath}. Saving the last model state manually.")
    model.save(checkpoint_filepath)
    model_loaded = model  # Use the current model state

'''

# Plot training history
plot_training_history(history, f'{model_name}, fold {j}', f'Training_History_fold{j}_{model_name}')



model = load_model(checkpoint_filepath, custom_object) # This ensures we only choose from versions within the fold
```

```
        # Evaluate the model
        test_loss, test_accuracy = model.evaluate(X_test, y_test)
        print(f'Test Loss {model_name}, fold {j}: {test_loss}, Test Accuracy {model_name}, fold {j}: {test_accuracy}')
        y_pred = np.argmax(model.predict(X_test), axis=1)
        y_pred_classes = np.round(y_pred).astype(int)

        # Calculate and store F1 score
        f1 = f1_score(y_test, y_pred_classes, average='macro')
        if model_name not in model_metrics:
            model_metrics[model_name] = {'F1_Scores': [], 'Average_F1': 0}
        model_metrics[model_name]['F1_Scores'].append(f1)

        print(f'F1 score for {model_name}, fold {j}: {f1}')
        f1_scores.append(f1)

        # Plot confusion matrix heatmap
        plot_confusion_matrix_heatmap(model, model_name, X_test, y_test, f'Confusion matrix for {model_name}, fold {j}', f'Prop_Confus

        j += 1


    average_f1_score = np.mean(model_metrics[model_name]['F1_Scores'])
    model_metrics[model_name]['Average_F1'] = average_f1_score

    print(f'RESULTS for {model_name}')

    print(f"Average F1 Score for {model_name}: {average_f1_score}")
    print(f'F1 scores for all folds for {model_name}:', model_metrics[model_name]['F1_Scores'])
    print('') # Make room


save_f1_scores_table_as_eps(model_metrics, filename='f1_scores_table.eps')
save_f1_scores_grouped_as_eps(model_metrics, filename='f1_scores_grouped_performance.eps')
```

```
'''

Please use this format:


model_list = [

    (Attentive_LSTM_Model, 'A_C_LSTM_Model', {'MyAttention': MyAttention}),
    (TCN_Model, 'TCN_Model', {'ResidualBlock': ResidualBlock}),
    (CNN_Model, 'CNN_Model', None)

]

validate_and_store_metrics(model_list, X, y, k=5, epochs=50) #Example

'''
```

```
"\n\nPlease use this format:\n\n\nmodel_list = [\n    \n    (Attentive_LSTM_Model, 'A_C_LSTM_Model', {'MyAttention': MyAttention}),\n    (
```

Run validation for all models, and dump all eps files in zip for report

```
model_list = [

    (A_C_LSTM_Model, 'A_C_LSTM_Model', {'MyAttention': MyAttention}),
    (TCN_Model, 'TCN_Model', {'ResidualBlock': ResidualBlock}),
    (CNN_Model, 'CNN_Model', None)

]
```

```
validate_and_store_metrics(model_list, X, y, k=5, epochs=50)
```

```
!zip -r UIA_IMU_ID_Models_Val.zip /kaggle/working
```

```
Epoch 1/50
96/96 [==============================] - ETA: 0s - loss: 0.1387 - accuracy: 0.9530
Epoch 1: val_accuracy improved from -inf to 0.96349, saving model to /kaggle/working/A_C_LSTM_Model_Fold_0_Checkpoint.h5
96/96 [==============================] - 29s 209ms/step - loss: 0.1387 - accuracy: 0.9530 - val_loss: 0.1099 - ...
   val_accuracy: 0.9635
Epoch 2/50



96/96 [==============================] - ETA: 0s - loss: 0.1433 - accuracy: 0.9550
Epoch 2: val_accuracy did not improve from 0.96349
96/96 [==============================] - 19s 194ms/step - loss: 0.1433 - accuracy: 0.9550 - val_loss: 0.1360 - ...
   val_accuracy: 0.9531
Epoch 3/50
96/96 [==============================] - ETA: 0s - loss: 0.1523 - accuracy: 0.9462
Epoch 3: val_accuracy did not improve from 0.96349
96/96 [==============================] - 18s 192ms/step - loss: 0.1523 - accuracy: 0.9462 - val_loss: 0.1348 - ...
   val_accuracy: 0.9518
Epoch 4/50
96/96 [==============================] - ETA: 0s - loss: 0.1192 - accuracy: 0.9615
Epoch 4: val_accuracy improved from 0.96349 to 0.97132, saving model to /kaggle/working/A_C_LSTM_Model_Fold_0_Checkpoint.h5
96/96 [==============================] - 19s 193ms/step - loss: 0.1192 - accuracy: 0.9615 - val_loss: 0.0902 - ...
   val_accuracy: 0.9713
Epoch 5/50
96/96 [==============================] - ETA: 0s - loss: 0.0933 - accuracy: 0.9680
Epoch 5: val_accuracy improved from 0.97132 to 0.97914, saving model to /kaggle/working/A_C_LSTM_Model_Fold_0_Checkpoint.h5
96/96 [==============================] - 18s 192ms/step - loss: 0.0933 - accuracy: 0.9680 - val_loss: 0.0862 - ...
   val_accuracy: 0.9791
Epoch 6/50
96/96 [==============================] - ETA: 0s - loss: 0.1099 - accuracy: 0.9612
Epoch 6: val_accuracy did not improve from 0.97914
96/96 [==============================] - 19s 194ms/step - loss: 0.1099 - accuracy: 0.9612 - val_loss: 0.0994 - ...
   val_accuracy: 0.9609
Epoch 7/50
```

```
96/96 [==============================] - ETA: 0s - loss: 0.1246 - accuracy: 0.9566
Epoch 7: val_accuracy did not improve from 0.97914
96/96 [==============================] - 19s 198ms/step - loss: 0.1246 - accuracy: 0.9566 - val_loss: 0.1152 - ...
   val_accuracy: 0.9583
Epoch 8/50
96/96 [==============================] - ETA: 0s - loss: 0.1217 - accuracy: 0.9544
Epoch 8: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 191ms/step - loss: 0.1217 - accuracy: 0.9544 - val_loss: 0.1302 - ...
   val_accuracy: 0.9622
Epoch 9/50
96/96 [==============================] - ETA: 0s - loss: 0.1168 - accuracy: 0.9583
Epoch 9: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 192ms/step - loss: 0.1168 - accuracy: 0.9583 - val_loss: 0.0994 - ...
   val_accuracy: 0.9648
Epoch 10/50
96/96 [==============================] - ETA: 0s - loss: 0.1070 - accuracy: 0.9586
Epoch 10: val_accuracy did not improve from 0.97914
96/96 [==============================] - 19s 193ms/step - loss: 0.1070 - accuracy: 0.9586 - val_loss: 0.1017 - ...
   val_accuracy: 0.9648
Epoch 11/50
96/96 [==============================] - ETA: 0s - loss: 0.1249 - accuracy: 0.9534
Epoch 11: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 192ms/step - loss: 0.1249 - accuracy: 0.9534 - val_loss: 0.1094 - ...
   val_accuracy: 0.9648
Epoch 12/50
96/96 [==============================] - ETA: 0s - loss: 0.1073 - accuracy: 0.9625
Epoch 12: val_accuracy did not improve from 0.97914
96/96 [==============================] - 19s 196ms/step - loss: 0.1073 - accuracy: 0.9625 - val_loss: 0.1297 - ...
   val_accuracy: 0.9648
Epoch 13/50
96/96 [==============================] - ETA: 0s - loss: 0.1589 - accuracy: 0.9488
Epoch 13: val_accuracy did not improve from 0.97914
96/96 [==============================] - 19s 194ms/step - loss: 0.1589 - accuracy: 0.9488 - val_loss: 0.1491 - ...
   val_accuracy: 0.9439
Epoch 14/50
96/96 [==============================] - ETA: 0s - loss: 0.1351 - accuracy: 0.9478
Epoch 14: val_accuracy did not improve from 0.97914
96/96 [==============================] - 19s 193ms/step - loss: 0.1351 - accuracy: 0.9478 - val_loss: 0.1661 - ...
   val_accuracy: 0.9426
```

```
Epoch 15/50
96/96 [==============================] - ETA: 0s - loss: 0.1425 - accuracy: 0.9488
Epoch 15: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 192ms/step - loss: 0.1425 - accuracy: 0.9488 - val_loss: 0.1660 - ...
   val_accuracy: 0.9426
Epoch 16/50
96/96 [==============================] - ETA: 0s - loss: 0.1510 - accuracy: 0.9469
Epoch 16: val_accuracy did not improve from 0.97914
96/96 [==============================] - 19s 194ms/step - loss: 0.1510 - accuracy: 0.9469 - val_loss: 0.4335 - ...
   val_accuracy: 0.8905
Epoch 17/50
96/96 [==============================] - ETA: 0s - loss: 0.2074 - accuracy: 0.9361
Epoch 17: val_accuracy did not improve from 0.97914
96/96 [==============================] - 19s 193ms/step - loss: 0.2074 - accuracy: 0.9361 - val_loss: 0.1997 - ...
   val_accuracy: 0.9413
Epoch 18/50
96/96 [==============================] - ETA: 0s - loss: 0.1351 - accuracy: 0.9511
Epoch 18: val_accuracy did not improve from 0.97914
96/96 [==============================] - 19s 195ms/step - loss: 0.1351 - accuracy: 0.9511 - val_loss: 0.2001 - ...
   val_accuracy: 0.9413
Epoch 19/50
96/96 [==============================] - ETA: 0s - loss: 0.1767 - accuracy: 0.9377
Epoch 19: val_accuracy did not improve from 0.97914
96/96 [==============================] - 19s 194ms/step - loss: 0.1767 - accuracy: 0.9377 - val_loss: 0.1295 - ...
   val_accuracy: 0.9557
Epoch 20/50
96/96 [==============================] - ETA: 0s - loss: 0.1423 - accuracy: 0.9511
Epoch 20: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 192ms/step - loss: 0.1423 - accuracy: 0.9511 - val_loss: 0.1881 - ...
   val_accuracy: 0.9452
Epoch 21/50
96/96 [==============================] - ETA: 0s - loss: 0.1106 - accuracy: 0.9573
Epoch 21: val_accuracy did not improve from 0.97914
96/96 [==============================] - 19s 193ms/step - loss: 0.1106 - accuracy: 0.9573 - val_loss: 0.1109 - ...
   val_accuracy: 0.9713
Epoch 22/50
96/96 [==============================] - ETA: 0s - loss: 0.1248 - accuracy: 0.9550
Epoch 22: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 193ms/step - loss: 0.1248 - accuracy: 0.9550 - val_loss: 0.1914 - ...
```

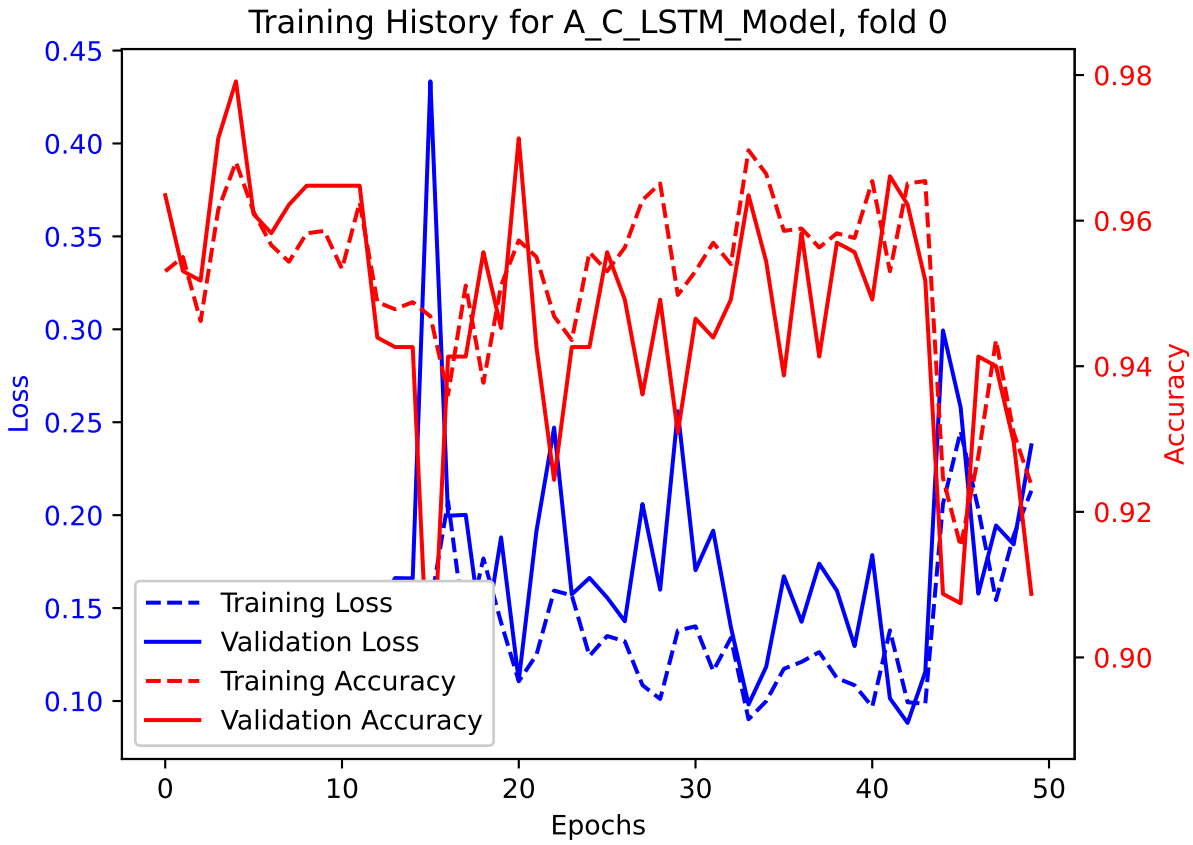
```
   val_accuracy: 0.9426
Epoch 23/50
96/96 [==============================] - ETA: 0s - loss: 0.1594 - accuracy: 0.9469
Epoch 23: val_accuracy did not improve from 0.97914
96/96 [==============================] - 19s 195ms/step - loss: 0.1594 - accuracy: 0.9469 - val_loss: 0.2471 - ...
   val_accuracy: 0.9244
Epoch 24/50
96/96 [==============================] - ETA: 0s - loss: 0.1569 - accuracy: 0.9436
Epoch 24: val_accuracy did not improve from 0.97914
96/96 [==============================] - 19s 198ms/step - loss: 0.1569 - accuracy: 0.9436 - val_loss: 0.1574 - ...
   val_accuracy: 0.9426
Epoch 25/50
96/96 [==============================] - ETA: 0s - loss: 0.1243 - accuracy: 0.9557
Epoch 25: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 193ms/step - loss: 0.1243 - accuracy: 0.9557 - val_loss: 0.1662 - ...
   val_accuracy: 0.9426
Epoch 26/50
96/96 [==============================] - ETA: 0s - loss: 0.1350 - accuracy: 0.9530
Epoch 26: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 192ms/step - loss: 0.1350 - accuracy: 0.9530 - val_loss: 0.1557 - ...
   val_accuracy: 0.9557
Epoch 27/50
96/96 [==============================] - ETA: 0s - loss: 0.1321 - accuracy: 0.9563
Epoch 27: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 186ms/step - loss: 0.1321 - accuracy: 0.9563 - val_loss: 0.1429 - ...
   val_accuracy: 0.9492
Epoch 28/50
96/96 [==============================] - ETA: 0s - loss: 0.1085 - accuracy: 0.9628
Epoch 28: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 183ms/step - loss: 0.1085 - accuracy: 0.9628 - val_loss: 0.2060 - ...
   val_accuracy: 0.9361
Epoch 29/50
96/96 [==============================] - ETA: 0s - loss: 0.1011 - accuracy: 0.9651
Epoch 29: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 185ms/step - loss: 0.1011 - accuracy: 0.9651 - val_loss: 0.1599 - ...
   val_accuracy: 0.9492
Epoch 30/50
96/96 [==============================] - ETA: 0s - loss: 0.1379 - accuracy: 0.9498
Epoch 30: val_accuracy did not improve from 0.97914
```

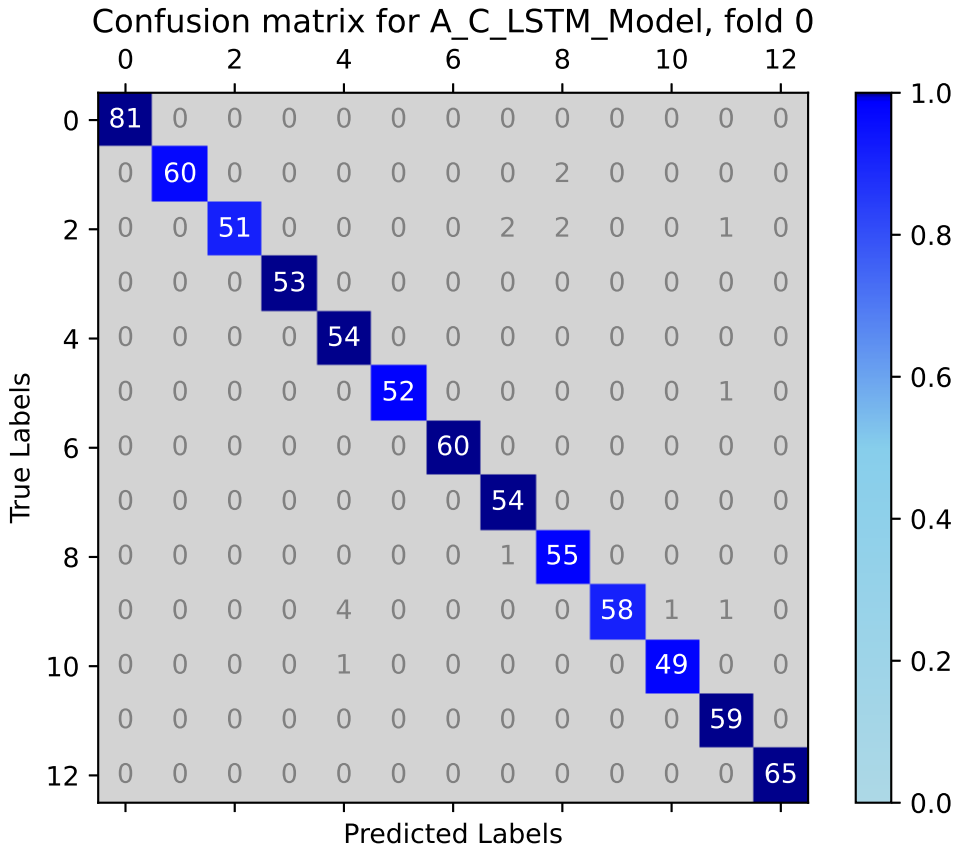
```
96/96 [==============================] - 18s 187ms/step - loss: 0.1379 - accuracy: 0.9498 - val_loss: 0.2559 - ...
    val_accuracy: 0.9309
Epoch 31/50
96/96 [==============================] - ETA: 0s - loss: 0.1402 - accuracy: 0.9530
Epoch 31: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 191ms/step - loss: 0.1402 - accuracy: 0.9530 - val_loss: 0.1704 - ...
    val_accuracy: 0.9465
Epoch 32/50
96/96 [==============================] - ETA: 0s - loss: 0.1162 - accuracy: 0.9570
Epoch 32: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 189ms/step - loss: 0.1162 - accuracy: 0.9570 - val_loss: 0.1917 - ...
    val_accuracy: 0.9439
Epoch 33/50
96/96 [==============================] - ETA: 0s - loss: 0.1339 - accuracy: 0.9540
Epoch 33: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 186ms/step - loss: 0.1339 - accuracy: 0.9540 - val_loss: 0.1398 - ...
    val_accuracy: 0.9492
Epoch 34/50
96/96 [==============================] - ETA: 0s - loss: 0.0902 - accuracy: 0.9697
Epoch 34: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 185ms/step - loss: 0.0902 - accuracy: 0.9697 - val_loss: 0.0981 - ...
    val_accuracy: 0.9635
Epoch 35/50
96/96 [==============================] - ETA: 0s - loss: 0.0999 - accuracy: 0.9664
Epoch 35: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 188ms/step - loss: 0.0999 - accuracy: 0.9664 - val_loss: 0.1184 - ...
    val_accuracy: 0.9544
Epoch 36/50
96/96 [==============================] - ETA: 0s - loss: 0.1173 - accuracy: 0.9586
Epoch 36: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 188ms/step - loss: 0.1173 - accuracy: 0.9586 - val_loss: 0.1671 - ...
    val_accuracy: 0.9387
Epoch 37/50
96/96 [==============================] - ETA: 0s - loss: 0.1211 - accuracy: 0.9589
Epoch 37: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 183ms/step - loss: 0.1211 - accuracy: 0.9589 - val_loss: 0.1426 - ...
    val_accuracy: 0.9583
Epoch 38/50
96/96 [==============================] - ETA: 0s - loss: 0.1263 - accuracy: 0.9563
```

```
Epoch 38: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 185ms/step - loss: 0.1263 - accuracy: 0.9563 - val_loss: 0.1739 - ...
   val_accuracy: 0.9413
Epoch 39/50
96/96 [==============================] - ETA: 0s - loss: 0.1124 - accuracy: 0.9583
Epoch 39: val_accuracy did not improve from 0.97914
96/96 [==============================] - 17s 181ms/step - loss: 0.1124 - accuracy: 0.9583 - val_loss: 0.1594 - ...
   val_accuracy: 0.9570
Epoch 40/50
96/96 [==============================] - ETA: 0s - loss: 0.1085 - accuracy: 0.9576
Epoch 40: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 186ms/step - loss: 0.1085 - accuracy: 0.9576 - val_loss: 0.1296 - ...
   val_accuracy: 0.9557
Epoch 41/50
96/96 [==============================] - ETA: 0s - loss: 0.0969 - accuracy: 0.9654
Epoch 41: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 183ms/step - loss: 0.0969 - accuracy: 0.9654 - val_loss: 0.1785 - ...
   val_accuracy: 0.9492
Epoch 42/50
96/96 [==============================] - ETA: 0s - loss: 0.1380 - accuracy: 0.9530
Epoch 42: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 184ms/step - loss: 0.1380 - accuracy: 0.9530 - val_loss: 0.1014 - ...
   val_accuracy: 0.9661
Epoch 43/50
96/96 [==============================] - ETA: 0s - loss: 0.0993 - accuracy: 0.9651
Epoch 43: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 184ms/step - loss: 0.0993 - accuracy: 0.9651 - val_loss: 0.0882 - ...
   val_accuracy: 0.9622
Epoch 44/50
96/96 [==============================] - ETA: 0s - loss: 0.0985 - accuracy: 0.9654
Epoch 44: val_accuracy did not improve from 0.97914
96/96 [==============================] - 17s 182ms/step - loss: 0.0985 - accuracy: 0.9654 - val_loss: 0.1157 - ...
   val_accuracy: 0.9518
Epoch 45/50
96/96 [==============================] - ETA: 0s - loss: 0.2062 - accuracy: 0.9247
Epoch 45: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 186ms/step - loss: 0.2062 - accuracy: 0.9247 - val_loss: 0.2994 - ...
   val_accuracy: 0.9087
Epoch 46/50
```

```
96/96 [==============================] - ETA: 0s - loss: 0.2452 - accuracy: 0.9152
Epoch 46: val_accuracy did not improve from 0.97914
96/96 [==============================] - 17s 181ms/step - loss: 0.2452 - accuracy: 0.9152 - val_loss: 0.2580 - ...
   val_accuracy: 0.9074
Epoch 47/50
96/96 [==============================] - ETA: 0s - loss: 0.2032 - accuracy: 0.9276
Epoch 47: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 184ms/step - loss: 0.2032 - accuracy: 0.9276 - val_loss: 0.1577 - ...
   val_accuracy: 0.9413
Epoch 48/50
96/96 [==============================] - ETA: 0s - loss: 0.1543 - accuracy: 0.9436
Epoch 48: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 183ms/step - loss: 0.1543 - accuracy: 0.9436 - val_loss: 0.1945 - ...
   val_accuracy: 0.9400
Epoch 49/50
96/96 [==============================] - ETA: 0s - loss: 0.1890 - accuracy: 0.9309
Epoch 49: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 183ms/step - loss: 0.1890 - accuracy: 0.9309 - val_loss: 0.1843 - ...
   val_accuracy: 0.9296
Epoch 50/50
96/96 [==============================] - ETA: 0s - loss: 0.2131 - accuracy: 0.9237
Epoch 50: val_accuracy did not improve from 0.97914
96/96 [==============================] - 18s 182ms/step - loss: 0.2131 - accuracy: 0.9237 - val_loss: 0.2376 - ...
   val_accuracy: 0.9087
```

```
24/24 [==============================] - 1s 18ms/step - loss: 0.0862 - accuracy: 0.9791
Test Loss A_C_LSTM_Model, fold 0: 0.08617115765810013, Test Accuracy A_C_LSTM_Model, fold 0: 0.979139506816864
24/24 [==============================] - 1s 17ms/step
F1 score for A_C_LSTM_Model, fold 0: 0.9783559218526776
24/24 [==============================] - 0s 17ms/step
```


Training History for A_C_LSTM_Model, fold 0
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy


Confusion matrix for A_C_LSTM_Model, fold 0
True Labels
Predicted Labels

```
Epoch 1/50
96/96 [==============================] - ETA: 0s - loss: 0.1160 - accuracy: 0.9622
Epoch 1: val_accuracy improved from -inf to 0.97001, saving model to /kaggle/working/A_C_LSTM_Model_Fold_1_Checkpoint.h5
96/96 [==============================] - 25s 193ms/step - loss: 0.1160 - accuracy: 0.9622 - val_loss: 0.0869 - ...
   val_accuracy: 0.9700
Epoch 2/50



96/96 [==============================] - ETA: 0s - loss: 0.1075 - accuracy: 0.9658
Epoch 2: val_accuracy improved from 0.97001 to 0.97784, saving model to /kaggle/working/A_C_LSTM_Model_Fold_1_Checkpoint.h5
96/96 [==============================] - 18s 184ms/step - loss: 0.1075 - accuracy: 0.9658 - val_loss: 0.0558 - ...
   val_accuracy: 0.9778
Epoch 3/50
96/96 [==============================] - ETA: 0s - loss: 0.1296 - accuracy: 0.9576
Epoch 3: val_accuracy improved from 0.97784 to 0.98044, saving model to /kaggle/working/A_C_LSTM_Model_Fold_1_Checkpoint.h5
96/96 [==============================] - 18s 183ms/step - loss: 0.1296 - accuracy: 0.9576 - val_loss: 0.0516 - ...
   val_accuracy: 0.9804
Epoch 4/50
96/96 [==============================] - ETA: 0s - loss: 0.1385 - accuracy: 0.9521
Epoch 4: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 186ms/step - loss: 0.1385 - accuracy: 0.9521 - val_loss: 0.1057 - ...
   val_accuracy: 0.9661
Epoch 5/50
96/96 [==============================] - ETA: 0s - loss: 0.1740 - accuracy: 0.9390
Epoch 5: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 186ms/step - loss: 0.1740 - accuracy: 0.9390 - val_loss: 0.0923 - ...
   val_accuracy: 0.9739
Epoch 6/50
96/96 [==============================] - ETA: 0s - loss: 0.1598 - accuracy: 0.9446
Epoch 6: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 186ms/step - loss: 0.1598 - accuracy: 0.9446 - val_loss: 0.1307 - ...
   val_accuracy: 0.9557
Epoch 7/50
96/96 [==============================] - ETA: 0s - loss: 0.1017 - accuracy: 0.9645
Epoch 7: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 185ms/step - loss: 0.1017 - accuracy: 0.9645 - val_loss: 0.1025 - ...
   val_accuracy: 0.9661
```

```
Epoch 8/50
96/96 [==============================] - ETA: 0s - loss: 0.0945 - accuracy: 0.9661
Epoch 8: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 185ms/step - loss: 0.0945 - accuracy: 0.9661 - val_loss: 0.0953 - ...
   val_accuracy: 0.9713
Epoch 9/50
96/96 [==============================] - ETA: 0s - loss: 0.1447 - accuracy: 0.9472
Epoch 9: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 184ms/step - loss: 0.1447 - accuracy: 0.9472 - val_loss: 0.1265 - ...
   val_accuracy: 0.9531
Epoch 10/50
96/96 [==============================] - ETA: 0s - loss: 0.1639 - accuracy: 0.9429
Epoch 10: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 185ms/step - loss: 0.1639 - accuracy: 0.9429 - val_loss: 0.1137 - ...
   val_accuracy: 0.9531
Epoch 11/50
96/96 [==============================] - ETA: 0s - loss: 0.1413 - accuracy: 0.9508
Epoch 11: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 186ms/step - loss: 0.1413 - accuracy: 0.9508 - val_loss: 0.1027 - ...
   val_accuracy: 0.9687
Epoch 12/50
96/96 [==============================] - ETA: 0s - loss: 0.1235 - accuracy: 0.9570
Epoch 12: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 183ms/step - loss: 0.1235 - accuracy: 0.9570 - val_loss: 0.1176 - ...
   val_accuracy: 0.9531
Epoch 13/50
96/96 [==============================] - ETA: 0s - loss: 0.1483 - accuracy: 0.9478
Epoch 13: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 186ms/step - loss: 0.1483 - accuracy: 0.9478 - val_loss: 0.0964 - ...
   val_accuracy: 0.9622
Epoch 14/50
96/96 [==============================] - ETA: 0s - loss: 0.1687 - accuracy: 0.9390
Epoch 14: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 186ms/step - loss: 0.1687 - accuracy: 0.9390 - val_loss: 0.1554 - ...
   val_accuracy: 0.9518
Epoch 15/50
96/96 [==============================] - ETA: 0s - loss: 0.1388 - accuracy: 0.9508
Epoch 15: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 184ms/step - loss: 0.1388 - accuracy: 0.9508 - val_loss: 0.1020 - ...
```

```
   val_accuracy: 0.9622
Epoch 16/50
96/96 [==============================] - ETA: 0s - loss: 0.1324 - accuracy: 0.9524
Epoch 16: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 184ms/step - loss: 0.1324 - accuracy: 0.9524 - val_loss: 0.1215 - ...
   val_accuracy: 0.9518
Epoch 17/50
96/96 [==============================] - ETA: 0s - loss: 0.1367 - accuracy: 0.9514
Epoch 17: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 183ms/step - loss: 0.1367 - accuracy: 0.9514 - val_loss: 0.0983 - ...
   val_accuracy: 0.9622
Epoch 18/50
96/96 [==============================] - ETA: 0s - loss: 0.1631 - accuracy: 0.9407
Epoch 18: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 186ms/step - loss: 0.1631 - accuracy: 0.9407 - val_loss: 0.1358 - ...
   val_accuracy: 0.9465
Epoch 19/50
96/96 [==============================] - ETA: 0s - loss: 0.1641 - accuracy: 0.9429
Epoch 19: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 187ms/step - loss: 0.1641 - accuracy: 0.9429 - val_loss: 0.1083 - ...
   val_accuracy: 0.9648
Epoch 20/50
96/96 [==============================] - ETA: 0s - loss: 0.1103 - accuracy: 0.9602
Epoch 20: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 183ms/step - loss: 0.1103 - accuracy: 0.9602 - val_loss: 0.0936 - ...
   val_accuracy: 0.9752
Epoch 21/50
96/96 [==============================] - ETA: 0s - loss: 0.1603 - accuracy: 0.9462
Epoch 21: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 184ms/step - loss: 0.1603 - accuracy: 0.9462 - val_loss: 0.1979 - ...
   val_accuracy: 0.9270
Epoch 22/50
96/96 [==============================] - ETA: 0s - loss: 0.1958 - accuracy: 0.9351
Epoch 22: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 191ms/step - loss: 0.1958 - accuracy: 0.9351 - val_loss: 0.1102 - ...
   val_accuracy: 0.9609
Epoch 23/50
96/96 [==============================] - ETA: 0s - loss: 0.1258 - accuracy: 0.9563
Epoch 23: val_accuracy did not improve from 0.98044
```

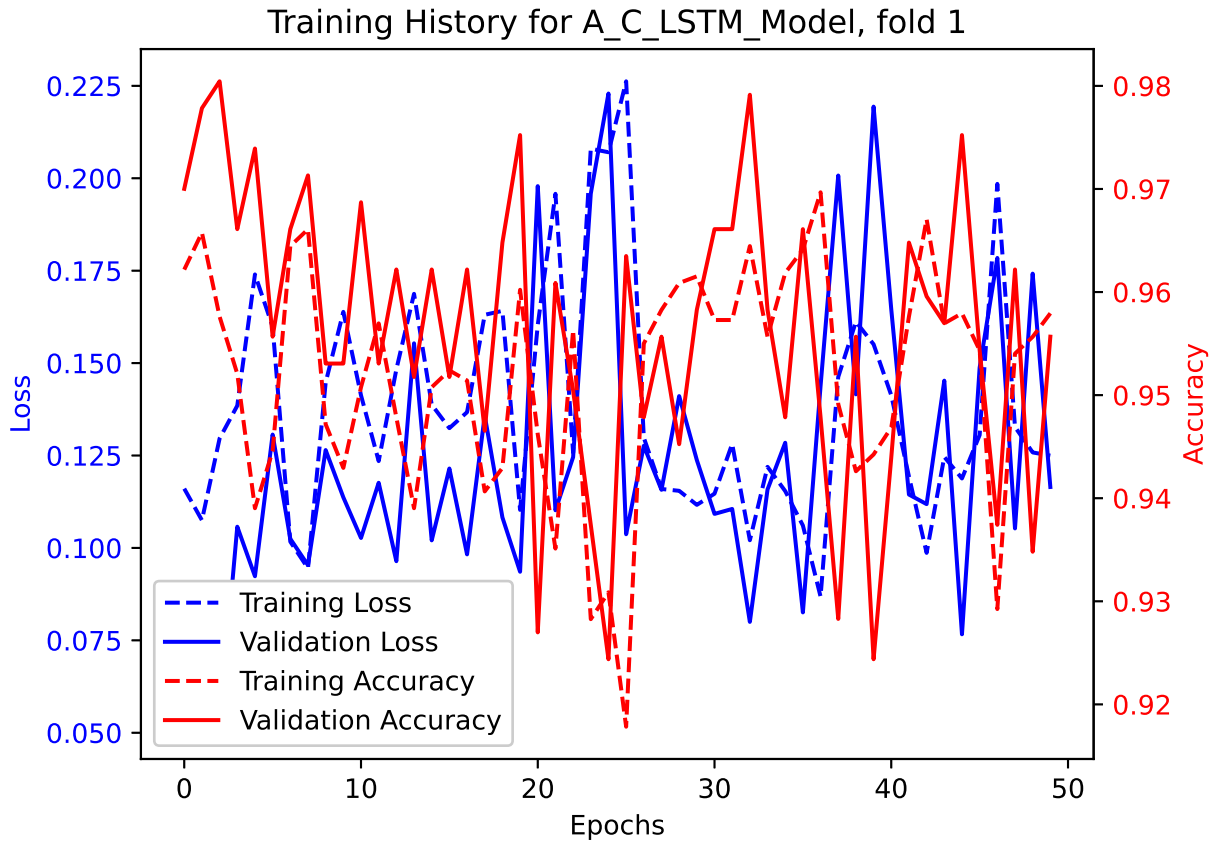
```
96/96 [==============================] - 18s 187ms/step - loss: 0.1258 - accuracy: 0.9563 - val_loss: 0.1246 - ...
   val_accuracy: 0.9505
Epoch 24/50
96/96 [==============================] - ETA: 0s - loss: 0.2080 - accuracy: 0.9283
Epoch 24: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 185ms/step - loss: 0.2080 - accuracy: 0.9283 - val_loss: 0.1958 - ...
   val_accuracy: 0.9374
Epoch 25/50
96/96 [==============================] - ETA: 0s - loss: 0.2070 - accuracy: 0.9309
Epoch 25: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 183ms/step - loss: 0.2070 - accuracy: 0.9309 - val_loss: 0.2229 - ...
   val_accuracy: 0.9244
Epoch 26/50
96/96 [==============================] - ETA: 0s - loss: 0.2262 - accuracy: 0.9178
Epoch 26: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 184ms/step - loss: 0.2262 - accuracy: 0.9178 - val_loss: 0.1037 - ...
   val_accuracy: 0.9635
Epoch 27/50
96/96 [==============================] - ETA: 0s - loss: 0.1298 - accuracy: 0.9550
Epoch 27: val_accuracy did not improve from 0.98044
96/96 [==============================] - 17s 182ms/step - loss: 0.1298 - accuracy: 0.9550 - val_loss: 0.1276 - ...
   val_accuracy: 0.9478
Epoch 28/50
96/96 [==============================] - ETA: 0s - loss: 0.1159 - accuracy: 0.9583
Epoch 28: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 187ms/step - loss: 0.1159 - accuracy: 0.9583 - val_loss: 0.1157 - ...
   val_accuracy: 0.9557
Epoch 29/50
96/96 [==============================] - ETA: 0s - loss: 0.1155 - accuracy: 0.9609
Epoch 29: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 185ms/step - loss: 0.1155 - accuracy: 0.9609 - val_loss: 0.1411 - ...
   val_accuracy: 0.9452
Epoch 30/50
96/96 [==============================] - ETA: 0s - loss: 0.1117 - accuracy: 0.9615
Epoch 30: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 187ms/step - loss: 0.1117 - accuracy: 0.9615 - val_loss: 0.1238 - ...
   val_accuracy: 0.9583
Epoch 31/50
96/96 [==============================] - ETA: 0s - loss: 0.1147 - accuracy: 0.9573
```

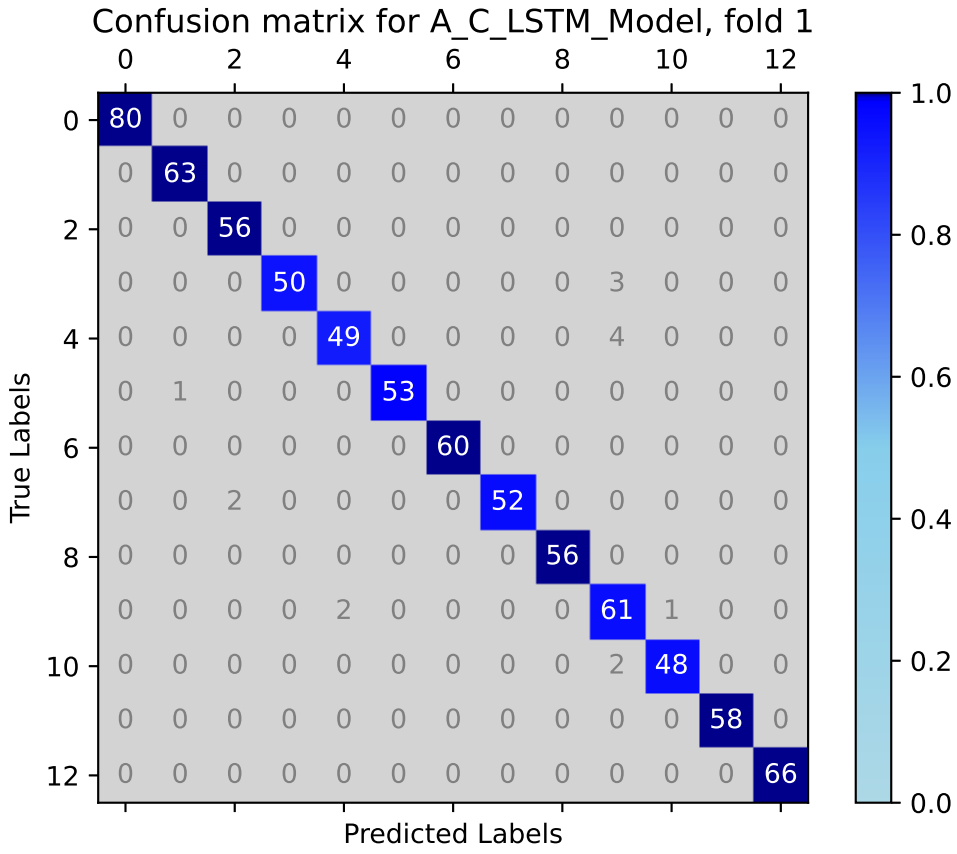
```
Epoch 31: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 188ms/step - loss: 0.1147 - accuracy: 0.9573 - val_loss: 0.1092 - ...
   val_accuracy: 0.9661
Epoch 32/50
96/96 [==============================] - ETA: 0s - loss: 0.1282 - accuracy: 0.9573
Epoch 32: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 184ms/step - loss: 0.1282 - accuracy: 0.9573 - val_loss: 0.1105 - ...
   val_accuracy: 0.9661
Epoch 33/50
96/96 [==============================] - ETA: 0s - loss: 0.1021 - accuracy: 0.9645
Epoch 33: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 188ms/step - loss: 0.1021 - accuracy: 0.9645 - val_loss: 0.0800 - ...
   val_accuracy: 0.9791
Epoch 34/50
96/96 [==============================] - ETA: 0s - loss: 0.1219 - accuracy: 0.9557
Epoch 34: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 185ms/step - loss: 0.1219 - accuracy: 0.9557 - val_loss: 0.1158 - ...
   val_accuracy: 0.9583
Epoch 35/50
96/96 [==============================] - ETA: 0s - loss: 0.1154 - accuracy: 0.9619
Epoch 35: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 187ms/step - loss: 0.1154 - accuracy: 0.9619 - val_loss: 0.1285 - ...
   val_accuracy: 0.9478
Epoch 36/50
96/96 [==============================] - ETA: 0s - loss: 0.1061 - accuracy: 0.9641
Epoch 36: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 184ms/step - loss: 0.1061 - accuracy: 0.9641 - val_loss: 0.0826 - ...
   val_accuracy: 0.9661
Epoch 37/50
96/96 [==============================] - ETA: 0s - loss: 0.0868 - accuracy: 0.9697
Epoch 37: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 188ms/step - loss: 0.0868 - accuracy: 0.9697 - val_loss: 0.1438 - ...
   val_accuracy: 0.9478
Epoch 38/50
96/96 [==============================] - ETA: 0s - loss: 0.1463 - accuracy: 0.9491
Epoch 38: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 189ms/step - loss: 0.1463 - accuracy: 0.9491 - val_loss: 0.2007 - ...
   val_accuracy: 0.9283
Epoch 39/50
```

```
96/96 [==============================] - ETA: 0s - loss: 0.1612 - accuracy: 0.9426
Epoch 39: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 184ms/step - loss: 0.1612 - accuracy: 0.9426 - val_loss: 0.1415 - ...
   val_accuracy: 0.9557
Epoch 40/50
96/96 [==============================] - ETA: 0s - loss: 0.1551 - accuracy: 0.9442
Epoch 40: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 187ms/step - loss: 0.1551 - accuracy: 0.9442 - val_loss: 0.2194 - ...
   val_accuracy: 0.9244
Epoch 41/50
96/96 [==============================] - ETA: 0s - loss: 0.1413 - accuracy: 0.9469
Epoch 41: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 185ms/step - loss: 0.1413 - accuracy: 0.9469 - val_loss: 0.1645 - ...
   val_accuracy: 0.9439
Epoch 42/50
96/96 [==============================] - ETA: 0s - loss: 0.1193 - accuracy: 0.9576
Epoch 42: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 188ms/step - loss: 0.1193 - accuracy: 0.9576 - val_loss: 0.1144 - ...
   val_accuracy: 0.9648
Epoch 43/50
96/96 [==============================] - ETA: 0s - loss: 0.0987 - accuracy: 0.9671
Epoch 43: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 186ms/step - loss: 0.0987 - accuracy: 0.9671 - val_loss: 0.1119 - ...
   val_accuracy: 0.9596
Epoch 44/50
96/96 [==============================] - ETA: 0s - loss: 0.1246 - accuracy: 0.9570
Epoch 44: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 185ms/step - loss: 0.1246 - accuracy: 0.9570 - val_loss: 0.1453 - ...
   val_accuracy: 0.9570
Epoch 45/50
96/96 [==============================] - ETA: 0s - loss: 0.1188 - accuracy: 0.9579
Epoch 45: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 187ms/step - loss: 0.1188 - accuracy: 0.9579 - val_loss: 0.0766 - ...
   val_accuracy: 0.9752
Epoch 46/50
96/96 [==============================] - ETA: 0s - loss: 0.1305 - accuracy: 0.9544
Epoch 46: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 186ms/step - loss: 0.1305 - accuracy: 0.9544 - val_loss: 0.1488 - ...
   val_accuracy: 0.9544
```

```
Epoch 47/50
96/96 [==============================] - ETA: 0s - loss: 0.1985 - accuracy: 0.9292
Epoch 47: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 188ms/step - loss: 0.1985 - accuracy: 0.9292 - val_loss: 0.1784 - ...
   val_accuracy: 0.9374
Epoch 48/50
96/96 [==============================] - ETA: 0s - loss: 0.1327 - accuracy: 0.9540
Epoch 48: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 185ms/step - loss: 0.1327 - accuracy: 0.9540 - val_loss: 0.1053 - ...
   val_accuracy: 0.9622
Epoch 49/50
96/96 [==============================] - ETA: 0s - loss: 0.1258 - accuracy: 0.9557
Epoch 49: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 186ms/step - loss: 0.1258 - accuracy: 0.9557 - val_loss: 0.1742 - ...
   val_accuracy: 0.9348
Epoch 50/50
96/96 [==============================] - ETA: 0s - loss: 0.1250 - accuracy: 0.9579
Epoch 50: val_accuracy did not improve from 0.98044
96/96 [==============================] - 18s 186ms/step - loss: 0.1250 - accuracy: 0.9579 - val_loss: 0.1165 - ...
   val_accuracy: 0.9557
```

```
24/24 [==============================] - 1s 18ms/step - loss: 0.0516 - accuracy: 0.9804
Test Loss A_C_LSTM_Model, fold 1: 0.05163516104221344, Test Accuracy A_C_LSTM_Model, fold 1: 0.9804432988166809
24/24 [==============================] - 1s 17ms/step
F1 score for A_C_LSTM_Model, fold 1: 0.9799765088683448
24/24 [==============================] - 0s 17ms/step
```


Training History for A_C_LSTM_Model, fold 1
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.98
0.97
0.96
0.95
0.94
0.93
0.92
0
10
20
30
40
50


Confusion matrix for A_C_LSTM_Model, fold 1
True Labels
Predicted Labels
0 2 4 6 8 10 12
80 0 0 0 0 0 0 0 0 0 0 0 0
0 63 0 0 0 0 0 0 0 0 0 0 0
0 0 56 0 0 0 0 0 0 0 0 0 0
0 0 0 50 0 0 0 0 0 3 0 0 0
0 0 0 0 49 0 0 0 0 4 0 0 0
0 1 0 0 0 53 0 0 0 0 0 0 0
0 0 0 0 0 0 60 0 0 0 0 0 0
0 0 2 0 0 0 0 52 0 0 0 0 0
0 0 0 0 0 0 0 0 56 0 0 0 0
0 0 0 0 2 0 0 0 0 61 1 0 0
0 0 0 0 0 0 0 0 0 2 48 0 0
0 0 0 0 0 0 0 0 0 0 0 58 0
0 0 0 0 0 0 0 0 0 0 0 0 66
1.0
0.8
0.6
0.4
0.2
0.0

```
Epoch 1/50
96/96 [==============================] - ETA: 0s - loss: 0.1186 - accuracy: 0.9586
Epoch 1: val_accuracy improved from -inf to 0.96610, saving model to /kaggle/working/A_C_LSTM_Model_Fold_2_Checkpoint.h5
96/96 [==============================] - 25s 194ms/step - loss: 0.1186 - accuracy: 0.9586 - val_loss: 0.1026 - ...
   val_accuracy: 0.9661
Epoch 2/50



96/96 [==============================] - ETA: 0s - loss: 0.1291 - accuracy: 0.9547
Epoch 2: val_accuracy did not improve from 0.96610
96/96 [==============================] - 18s 187ms/step - loss: 0.1291 - accuracy: 0.9547 - val_loss: 0.1085 - ...
   val_accuracy: 0.9661
Epoch 3/50
96/96 [==============================] - ETA: 0s - loss: 0.1226 - accuracy: 0.9537
Epoch 3: val_accuracy did not improve from 0.96610
96/96 [==============================] - 18s 187ms/step - loss: 0.1226 - accuracy: 0.9537 - val_loss: 0.1326 - ...
   val_accuracy: 0.9622
Epoch 4/50
96/96 [==============================] - ETA: 0s - loss: 0.1334 - accuracy: 0.9514
Epoch 4: val_accuracy did not improve from 0.96610
96/96 [==============================] - 18s 188ms/step - loss: 0.1334 - accuracy: 0.9514 - val_loss: 0.1428 - ...
   val_accuracy: 0.9413
Epoch 5/50
96/96 [==============================] - ETA: 0s - loss: 0.1406 - accuracy: 0.9527
Epoch 5: val_accuracy did not improve from 0.96610
96/96 [==============================] - 18s 190ms/step - loss: 0.1406 - accuracy: 0.9527 - val_loss: 0.0987 - ...
   val_accuracy: 0.9557
Epoch 6/50
96/96 [==============================] - ETA: 0s - loss: 0.1507 - accuracy: 0.9482
Epoch 6: val_accuracy did not improve from 0.96610
96/96 [==============================] - 18s 187ms/step - loss: 0.1507 - accuracy: 0.9482 - val_loss: 0.1851 - ...
   val_accuracy: 0.9413
Epoch 7/50
96/96 [==============================] - ETA: 0s - loss: 0.1308 - accuracy: 0.9576
Epoch 7: val_accuracy did not improve from 0.96610
96/96 [==============================] - 18s 187ms/step - loss: 0.1308 - accuracy: 0.9576 - val_loss: 0.1342 - ...
   val_accuracy: 0.9622
```

```
Epoch 8/50
96/96 [==============================] - ETA: 0s - loss: 0.1458 - accuracy: 0.9485
Epoch 8: val_accuracy improved from 0.96610 to 0.97001, saving model to /kaggle/working/A_C_LSTM_Model_Fold_2_Checkpoint.h5
96/96 [==============================] - 18s 187ms/step - loss: 0.1458 - accuracy: 0.9485 - val_loss: 0.0940 - ...
   val_accuracy: 0.9700
Epoch 9/50
96/96 [==============================] - ETA: 0s - loss: 0.0966 - accuracy: 0.9658
Epoch 9: val_accuracy improved from 0.97001 to 0.97523, saving model to /kaggle/working/A_C_LSTM_Model_Fold_2_Checkpoint.h5
96/96 [==============================] - 18s 187ms/step - loss: 0.0966 - accuracy: 0.9658 - val_loss: 0.0746 - ...
   val_accuracy: 0.9752
Epoch 10/50
96/96 [==============================] - ETA: 0s - loss: 0.0906 - accuracy: 0.9687
Epoch 10: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 187ms/step - loss: 0.0906 - accuracy: 0.9687 - val_loss: 0.1214 - ...
   val_accuracy: 0.9648
Epoch 11/50
96/96 [==============================] - ETA: 0s - loss: 0.1512 - accuracy: 0.9527
Epoch 11: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 187ms/step - loss: 0.1512 - accuracy: 0.9527 - val_loss: 0.1173 - ...
   val_accuracy: 0.9518
Epoch 12/50
96/96 [==============================] - ETA: 0s - loss: 0.1712 - accuracy: 0.9397
Epoch 12: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 188ms/step - loss: 0.1712 - accuracy: 0.9397 - val_loss: 0.1386 - ...
   val_accuracy: 0.9478
Epoch 13/50
96/96 [==============================] - ETA: 0s - loss: 0.1525 - accuracy: 0.9465
Epoch 13: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 188ms/step - loss: 0.1525 - accuracy: 0.9465 - val_loss: 0.1385 - ...
   val_accuracy: 0.9570
Epoch 14/50
96/96 [==============================] - ETA: 0s - loss: 0.1580 - accuracy: 0.9436
Epoch 14: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 188ms/step - loss: 0.1580 - accuracy: 0.9436 - val_loss: 0.1909 - ...
   val_accuracy: 0.9374
Epoch 15/50
96/96 [==============================] - ETA: 0s - loss: 0.1549 - accuracy: 0.9465
Epoch 15: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 187ms/step - loss: 0.1549 - accuracy: 0.9465 - val_loss: 0.1193 - ...
```

```
   val_accuracy: 0.9452
Epoch 16/50
96/96 [==============================] - ETA: 0s - loss: 0.1135 - accuracy: 0.9605
Epoch 16: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 186ms/step - loss: 0.1135 - accuracy: 0.9605 - val_loss: 0.0854 - ...
   val_accuracy: 0.9726
Epoch 17/50
96/96 [==============================] - ETA: 0s - loss: 0.0892 - accuracy: 0.9645
Epoch 17: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 189ms/step - loss: 0.0892 - accuracy: 0.9645 - val_loss: 0.0794 - ...
   val_accuracy: 0.9700
Epoch 18/50
96/96 [==============================] - ETA: 0s - loss: 0.1484 - accuracy: 0.9488
Epoch 18: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 187ms/step - loss: 0.1484 - accuracy: 0.9488 - val_loss: 0.2840 - ...
   val_accuracy: 0.9074
Epoch 19/50
96/96 [==============================] - ETA: 0s - loss: 0.1680 - accuracy: 0.9433
Epoch 19: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 186ms/step - loss: 0.1680 - accuracy: 0.9433 - val_loss: 0.1519 - ...
   val_accuracy: 0.9478
Epoch 20/50
96/96 [==============================] - ETA: 0s - loss: 0.1127 - accuracy: 0.9641
Epoch 20: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 185ms/step - loss: 0.1127 - accuracy: 0.9641 - val_loss: 0.1194 - ...
   val_accuracy: 0.9544
Epoch 21/50
96/96 [==============================] - ETA: 0s - loss: 0.1076 - accuracy: 0.9602
Epoch 21: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 188ms/step - loss: 0.1076 - accuracy: 0.9602 - val_loss: 0.0896 - ...
   val_accuracy: 0.9583
Epoch 22/50
96/96 [==============================] - ETA: 0s - loss: 0.0910 - accuracy: 0.9658
Epoch 22: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 190ms/step - loss: 0.0910 - accuracy: 0.9658 - val_loss: 0.0937 - ...
   val_accuracy: 0.9713
Epoch 23/50
96/96 [==============================] - ETA: 0s - loss: 0.1225 - accuracy: 0.9579
Epoch 23: val_accuracy did not improve from 0.97523
```

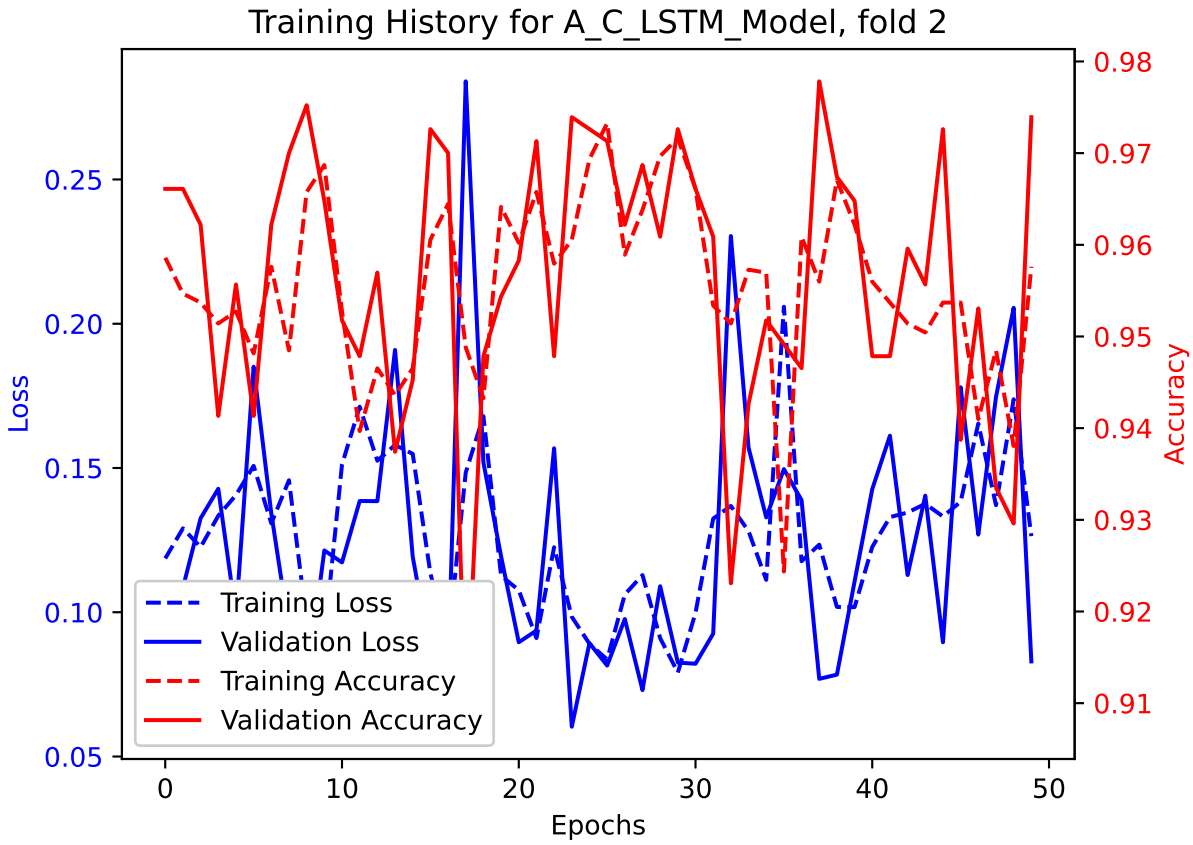
```
96/96 [==============================] - 18s 188ms/step - loss: 0.1225 - accuracy: 0.9579 - val_loss: 0.1568 - ...
   val_accuracy: 0.9478
Epoch 24/50
96/96 [==============================] - ETA: 0s - loss: 0.0984 - accuracy: 0.9605
Epoch 24: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 186ms/step - loss: 0.0984 - accuracy: 0.9605 - val_loss: 0.0603 - ...
   val_accuracy: 0.9739
Epoch 25/50
96/96 [==============================] - ETA: 0s - loss: 0.0891 - accuracy: 0.9694
Epoch 25: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 187ms/step - loss: 0.0891 - accuracy: 0.9694 - val_loss: 0.0893 - ...
   val_accuracy: 0.9726
Epoch 26/50
96/96 [==============================] - ETA: 0s - loss: 0.0836 - accuracy: 0.9733
Epoch 26: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 188ms/step - loss: 0.0836 - accuracy: 0.9733 - val_loss: 0.0815 - ...
   val_accuracy: 0.9713
Epoch 27/50
96/96 [==============================] - ETA: 0s - loss: 0.1062 - accuracy: 0.9589
Epoch 27: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 188ms/step - loss: 0.1062 - accuracy: 0.9589 - val_loss: 0.0977 - ...
   val_accuracy: 0.9622
Epoch 28/50
96/96 [==============================] - ETA: 0s - loss: 0.1129 - accuracy: 0.9638
Epoch 28: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 187ms/step - loss: 0.1129 - accuracy: 0.9638 - val_loss: 0.0730 - ...
   val_accuracy: 0.9687
Epoch 29/50
96/96 [==============================] - ETA: 0s - loss: 0.0910 - accuracy: 0.9697
Epoch 29: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 184ms/step - loss: 0.0910 - accuracy: 0.9697 - val_loss: 0.1090 - ...
   val_accuracy: 0.9609
Epoch 30/50
96/96 [==============================] - ETA: 0s - loss: 0.0791 - accuracy: 0.9716
Epoch 30: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 188ms/step - loss: 0.0791 - accuracy: 0.9716 - val_loss: 0.0826 - ...
   val_accuracy: 0.9726
Epoch 31/50
96/96 [==============================] - ETA: 0s - loss: 0.0998 - accuracy: 0.9661
```

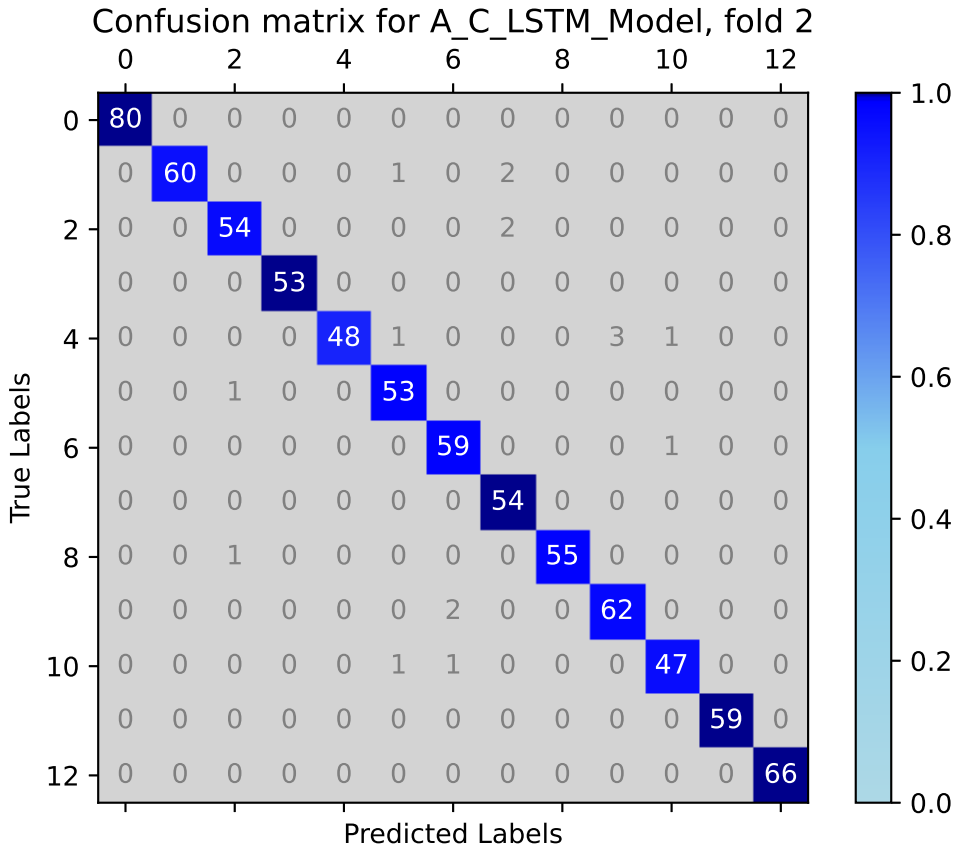
```
Epoch 31: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 185ms/step - loss: 0.0998 - accuracy: 0.9661 - val_loss: 0.0822 - ...
   val_accuracy: 0.9661
Epoch 32/50
96/96 [==============================] - ETA: 0s - loss: 0.1325 - accuracy: 0.9534
Epoch 32: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 186ms/step - loss: 0.1325 - accuracy: 0.9534 - val_loss: 0.0926 - ...
   val_accuracy: 0.9609
Epoch 33/50
96/96 [==============================] - ETA: 0s - loss: 0.1368 - accuracy: 0.9514
Epoch 33: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 188ms/step - loss: 0.1368 - accuracy: 0.9514 - val_loss: 0.2304 - ...
   val_accuracy: 0.9231
Epoch 34/50
96/96 [==============================] - ETA: 0s - loss: 0.1281 - accuracy: 0.9573
Epoch 34: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 186ms/step - loss: 0.1281 - accuracy: 0.9573 - val_loss: 0.1570 - ...
   val_accuracy: 0.9426
Epoch 35/50
96/96 [==============================] - ETA: 0s - loss: 0.1112 - accuracy: 0.9570
Epoch 35: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 188ms/step - loss: 0.1112 - accuracy: 0.9570 - val_loss: 0.1329 - ...
   val_accuracy: 0.9518
Epoch 36/50
96/96 [==============================] - ETA: 0s - loss: 0.2059 - accuracy: 0.9244
Epoch 36: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 185ms/step - loss: 0.2059 - accuracy: 0.9244 - val_loss: 0.1496 - ...
   val_accuracy: 0.9492
Epoch 37/50
96/96 [==============================] - ETA: 0s - loss: 0.1177 - accuracy: 0.9609
Epoch 37: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 188ms/step - loss: 0.1177 - accuracy: 0.9609 - val_loss: 0.1389 - ...
   val_accuracy: 0.9465
Epoch 38/50
96/96 [==============================] - ETA: 0s - loss: 0.1234 - accuracy: 0.9560
Epoch 38: val_accuracy improved from 0.97523 to 0.97784, saving model to ...
   /kaggle/working/A_C_LSTM_Model_Fold_2_Checkpoint.h5
96/96 [==============================] - 18s 185ms/step - loss: 0.1234 - accuracy: 0.9560 - val_loss: 0.0769 - ...
   val_accuracy: 0.9778
```

```
Epoch 39/50
96/96 [==============================] - ETA: 0s - loss: 0.1018 - accuracy: 0.9671
Epoch 39: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 190ms/step - loss: 0.1018 - accuracy: 0.9671 - val_loss: 0.0784 - ...
   val_accuracy: 0.9674
Epoch 40/50
96/96 [==============================] - ETA: 0s - loss: 0.1018 - accuracy: 0.9622
Epoch 40: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 188ms/step - loss: 0.1018 - accuracy: 0.9622 - val_loss: 0.1107 - ...
   val_accuracy: 0.9648
Epoch 41/50
96/96 [==============================] - ETA: 0s - loss: 0.1227 - accuracy: 0.9560
Epoch 41: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 186ms/step - loss: 0.1227 - accuracy: 0.9560 - val_loss: 0.1427 - ...
   val_accuracy: 0.9478
Epoch 42/50
96/96 [==============================] - ETA: 0s - loss: 0.1330 - accuracy: 0.9537
Epoch 42: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 188ms/step - loss: 0.1330 - accuracy: 0.9537 - val_loss: 0.1612 - ...
   val_accuracy: 0.9478
Epoch 43/50
96/96 [==============================] - ETA: 0s - loss: 0.1346 - accuracy: 0.9514
Epoch 43: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 184ms/step - loss: 0.1346 - accuracy: 0.9514 - val_loss: 0.1129 - ...
   val_accuracy: 0.9596
Epoch 44/50
96/96 [==============================] - ETA: 0s - loss: 0.1376 - accuracy: 0.9504
Epoch 44: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 189ms/step - loss: 0.1376 - accuracy: 0.9504 - val_loss: 0.1404 - ...
   val_accuracy: 0.9557
Epoch 45/50
96/96 [==============================] - ETA: 0s - loss: 0.1332 - accuracy: 0.9537
Epoch 45: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 186ms/step - loss: 0.1332 - accuracy: 0.9537 - val_loss: 0.0896 - ...
   val_accuracy: 0.9726
Epoch 46/50
96/96 [==============================] - ETA: 0s - loss: 0.1383 - accuracy: 0.9537
Epoch 46: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 187ms/step - loss: 0.1383 - accuracy: 0.9537 - val_loss: 0.1780 - ...
```

```
   val_accuracy: 0.9387
Epoch 47/50
96/96 [==============================] - ETA: 0s - loss: 0.1654 - accuracy: 0.9410
Epoch 47: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 190ms/step - loss: 0.1654 - accuracy: 0.9410 - val_loss: 0.1269 - ...
   val_accuracy: 0.9531
Epoch 48/50
96/96 [==============================] - ETA: 0s - loss: 0.1371 - accuracy: 0.9485
Epoch 48: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 186ms/step - loss: 0.1371 - accuracy: 0.9485 - val_loss: 0.1746 - ...
   val_accuracy: 0.9335
Epoch 49/50
96/96 [==============================] - ETA: 0s - loss: 0.1738 - accuracy: 0.9381
Epoch 49: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 187ms/step - loss: 0.1738 - accuracy: 0.9381 - val_loss: 0.2055 - ...
   val_accuracy: 0.9296
Epoch 50/50
96/96 [==============================] - ETA: 0s - loss: 0.1264 - accuracy: 0.9576
Epoch 50: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 186ms/step - loss: 0.1264 - accuracy: 0.9576 - val_loss: 0.0830 - ...
   val_accuracy: 0.9739
```

```
24/24 [==============================] - 1s 18ms/step - loss: 0.0769 - accuracy: 0.9778
Test Loss A_C_LSTM_Model, fold 2: 0.07691764831542969, Test Accuracy A_C_LSTM_Model, fold 2: 0.9778357148170471
24/24 [==============================] - 1s 18ms/step
F1 score for A_C_LSTM_Model, fold 2: 0.9766877451617154
24/24 [==============================] - 0s 17ms/step
```


Training History for A_C_LSTM_Model, fold 2
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
0.25
0.20
0.15
0.10
0.05
0.98
0.97
0.96
0.95
0.94
0.93
0.92
0.91
0
10
20
30
40
50


Confusion matrix for A_C_LSTM_Model, fold 2
True Labels
Predicted Labels
0 2 4 6 8 10 12
1.0
0.8
0.6
0.4
0.2
0.0

```
Epoch 1/50
96/96 [==============================] - ETA: 0s - loss: 0.0984 - accuracy: 0.9667
Epoch 1: val_accuracy improved from -inf to 0.97523, saving model to /kaggle/working/A_C_LSTM_Model_Fold_3_Checkpoint.h5
96/96 [==============================] - 25s 195ms/step - loss: 0.0984 - accuracy: 0.9667 - val_loss: 0.0627 - ...
   val_accuracy: 0.9752
Epoch 2/50



96/96 [==============================] - ETA: 0s - loss: 0.1291 - accuracy: 0.9602
Epoch 2: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 189ms/step - loss: 0.1291 - accuracy: 0.9602 - val_loss: 0.0971 - ...
   val_accuracy: 0.9635
Epoch 3/50
96/96 [==============================] - ETA: 0s - loss: 0.1271 - accuracy: 0.9586
Epoch 3: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 189ms/step - loss: 0.1271 - accuracy: 0.9586 - val_loss: 0.0889 - ...
   val_accuracy: 0.9661
Epoch 4/50
96/96 [==============================] - ETA: 0s - loss: 0.0921 - accuracy: 0.9697
Epoch 4: val_accuracy did not improve from 0.97523
96/96 [==============================] - 18s 190ms/step - loss: 0.0921 - accuracy: 0.9697 - val_loss: 0.0749 - ...
   val_accuracy: 0.9739
Epoch 5/50
96/96 [==============================] - ETA: 0s - loss: 0.1112 - accuracy: 0.9641
Epoch 5: val_accuracy improved from 0.97523 to 0.97784, saving model to /kaggle/working/A_C_LSTM_Model_Fold_3_Checkpoint.h5
96/96 [==============================] - 18s 187ms/step - loss: 0.1112 - accuracy: 0.9641 - val_loss: 0.0711 - ...
   val_accuracy: 0.9778
Epoch 6/50
96/96 [==============================] - ETA: 0s - loss: 0.2671 - accuracy: 0.9139
Epoch 6: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 190ms/step - loss: 0.2671 - accuracy: 0.9139 - val_loss: 0.2210 - ...
   val_accuracy: 0.9296
Epoch 7/50
96/96 [==============================] - ETA: 0s - loss: 0.2084 - accuracy: 0.9266
Epoch 7: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 188ms/step - loss: 0.2084 - accuracy: 0.9266 - val_loss: 0.1511 - ...
   val_accuracy: 0.9505
```

```
Epoch 8/50
96/96 [==============================] - ETA: 0s - loss: 0.1672 - accuracy: 0.9413
Epoch 8: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 187ms/step - loss: 0.1672 - accuracy: 0.9413 - val_loss: 0.1506 - ...
   val_accuracy: 0.9557
Epoch 9/50
96/96 [==============================] - ETA: 0s - loss: 0.1775 - accuracy: 0.9397
Epoch 9: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 186ms/step - loss: 0.1775 - accuracy: 0.9397 - val_loss: 0.0720 - ...
   val_accuracy: 0.9752
Epoch 10/50
96/96 [==============================] - ETA: 0s - loss: 0.1146 - accuracy: 0.9605
Epoch 10: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 187ms/step - loss: 0.1146 - accuracy: 0.9605 - val_loss: 0.0735 - ...
   val_accuracy: 0.9726
Epoch 11/50
96/96 [==============================] - ETA: 0s - loss: 0.1022 - accuracy: 0.9651
Epoch 11: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 189ms/step - loss: 0.1022 - accuracy: 0.9651 - val_loss: 0.0986 - ...
   val_accuracy: 0.9609
Epoch 12/50
96/96 [==============================] - ETA: 0s - loss: 0.0867 - accuracy: 0.9707
Epoch 12: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 187ms/step - loss: 0.0867 - accuracy: 0.9707 - val_loss: 0.0828 - ...
   val_accuracy: 0.9726
Epoch 13/50
96/96 [==============================] - ETA: 0s - loss: 0.1708 - accuracy: 0.9410
Epoch 13: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 188ms/step - loss: 0.1708 - accuracy: 0.9410 - val_loss: 0.1550 - ...
   val_accuracy: 0.9439
Epoch 14/50
96/96 [==============================] - ETA: 0s - loss: 0.1683 - accuracy: 0.9429
Epoch 14: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 188ms/step - loss: 0.1683 - accuracy: 0.9429 - val_loss: 0.1216 - ...
   val_accuracy: 0.9609
Epoch 15/50
96/96 [==============================] - ETA: 0s - loss: 0.1234 - accuracy: 0.9566
Epoch 15: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 187ms/step - loss: 0.1234 - accuracy: 0.9566 - val_loss: 0.1216 - ...
```

```
   val_accuracy: 0.9583
Epoch 16/50
96/96 [==============================] - ETA: 0s - loss: 0.1009 - accuracy: 0.9671
Epoch 16: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 189ms/step - loss: 0.1009 - accuracy: 0.9671 - val_loss: 0.0844 - ...
   val_accuracy: 0.9713
Epoch 17/50
96/96 [==============================] - ETA: 0s - loss: 0.2082 - accuracy: 0.9306
Epoch 17: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 190ms/step - loss: 0.2082 - accuracy: 0.9306 - val_loss: 0.1134 - ...
   val_accuracy: 0.9544
Epoch 18/50
96/96 [==============================] - ETA: 0s - loss: 0.1273 - accuracy: 0.9530
Epoch 18: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 185ms/step - loss: 0.1273 - accuracy: 0.9530 - val_loss: 0.2166 - ...
   val_accuracy: 0.9244
Epoch 19/50
96/96 [==============================] - ETA: 0s - loss: 0.1606 - accuracy: 0.9400
Epoch 19: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 186ms/step - loss: 0.1606 - accuracy: 0.9400 - val_loss: 0.1824 - ...
   val_accuracy: 0.9439
Epoch 20/50
96/96 [==============================] - ETA: 0s - loss: 0.1241 - accuracy: 0.9508
Epoch 20: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 189ms/step - loss: 0.1241 - accuracy: 0.9508 - val_loss: 0.1435 - ...
   val_accuracy: 0.9505
Epoch 21/50
96/96 [==============================] - ETA: 0s - loss: 0.0932 - accuracy: 0.9680
Epoch 21: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 185ms/step - loss: 0.0932 - accuracy: 0.9680 - val_loss: 0.0843 - ...
   val_accuracy: 0.9661
Epoch 22/50
96/96 [==============================] - ETA: 0s - loss: 0.1539 - accuracy: 0.9469
Epoch 22: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 185ms/step - loss: 0.1539 - accuracy: 0.9469 - val_loss: 0.1399 - ...
   val_accuracy: 0.9518
Epoch 23/50
96/96 [==============================] - ETA: 0s - loss: 0.1305 - accuracy: 0.9573
Epoch 23: val_accuracy did not improve from 0.97784
```

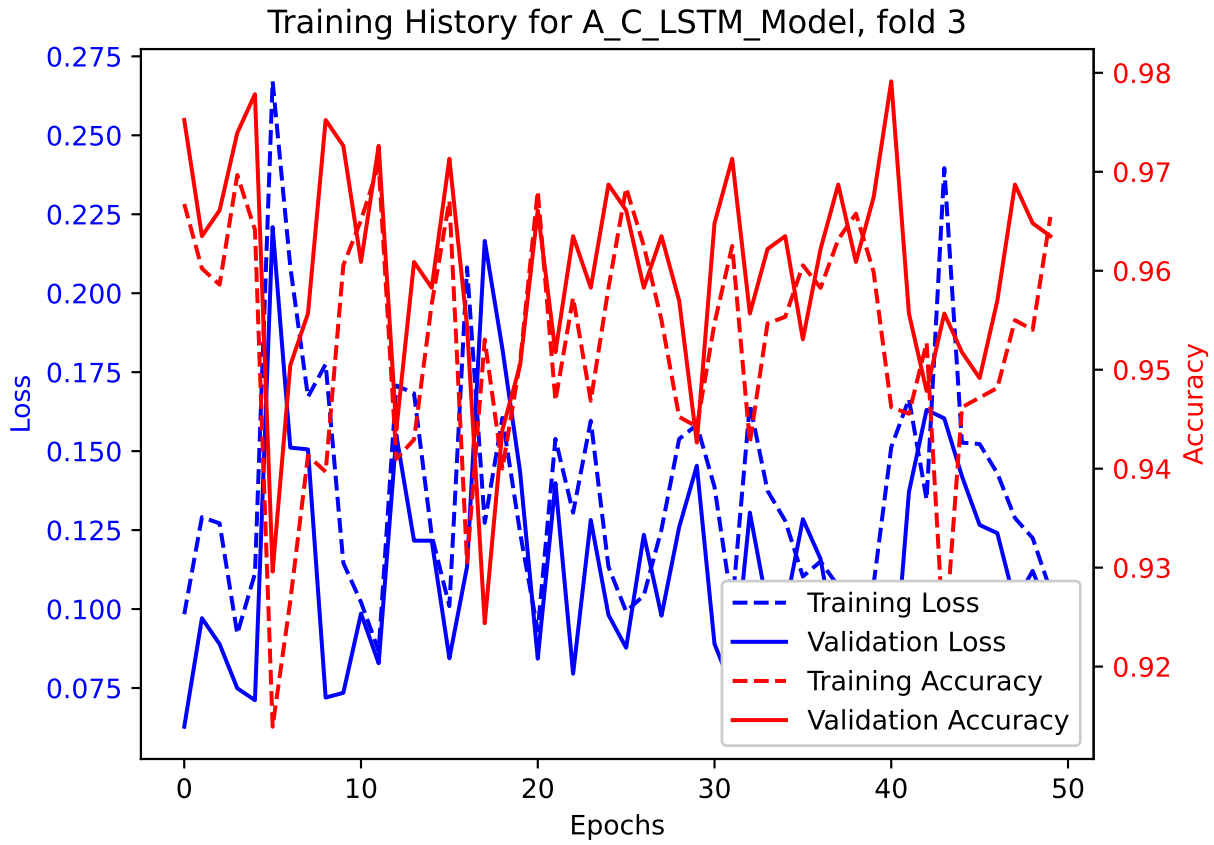
```
96/96 [==============================] - 18s 185ms/step - loss: 0.1305 - accuracy: 0.9573 - val_loss: 0.0795 - ...
   val_accuracy: 0.9635
Epoch 24/50
96/96 [==============================] - ETA: 0s - loss: 0.1596 - accuracy: 0.9469
Epoch 24: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 187ms/step - loss: 0.1596 - accuracy: 0.9469 - val_loss: 0.1282 - ...
   val_accuracy: 0.9583
Epoch 25/50
96/96 [==============================] - ETA: 0s - loss: 0.1134 - accuracy: 0.9583
Epoch 25: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 186ms/step - loss: 0.1134 - accuracy: 0.9583 - val_loss: 0.0981 - ...
   val_accuracy: 0.9687
Epoch 26/50
96/96 [==============================] - ETA: 0s - loss: 0.0992 - accuracy: 0.9684
Epoch 26: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 186ms/step - loss: 0.0992 - accuracy: 0.9684 - val_loss: 0.0877 - ...
   val_accuracy: 0.9661
Epoch 27/50
96/96 [==============================] - ETA: 0s - loss: 0.1043 - accuracy: 0.9625
Epoch 27: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 185ms/step - loss: 0.1043 - accuracy: 0.9625 - val_loss: 0.1235 - ...
   val_accuracy: 0.9583
Epoch 28/50
96/96 [==============================] - ETA: 0s - loss: 0.1252 - accuracy: 0.9550
Epoch 28: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 183ms/step - loss: 0.1252 - accuracy: 0.9550 - val_loss: 0.0979 - ...
   val_accuracy: 0.9635
Epoch 29/50
96/96 [==============================] - ETA: 0s - loss: 0.1539 - accuracy: 0.9452
Epoch 29: val_accuracy did not improve from 0.97784
96/96 [==============================] - 18s 188ms/step - loss: 0.1539 - accuracy: 0.9452 - val_loss: 0.1259 - ...
   val_accuracy: 0.9570
Epoch 30/50
96/96 [==============================] - ETA: 0s - loss: 0.1583 - accuracy: 0.9442
Epoch 30: val_accuracy did not improve from 0.97784
96/96 [==============================] - 19s 197ms/step - loss: 0.1583 - accuracy: 0.9442 - val_loss: 0.1454 - ...
   val_accuracy: 0.9426
Epoch 31/50
96/96 [==============================] - ETA: 0s - loss: 0.1387 - accuracy: 0.9547
```

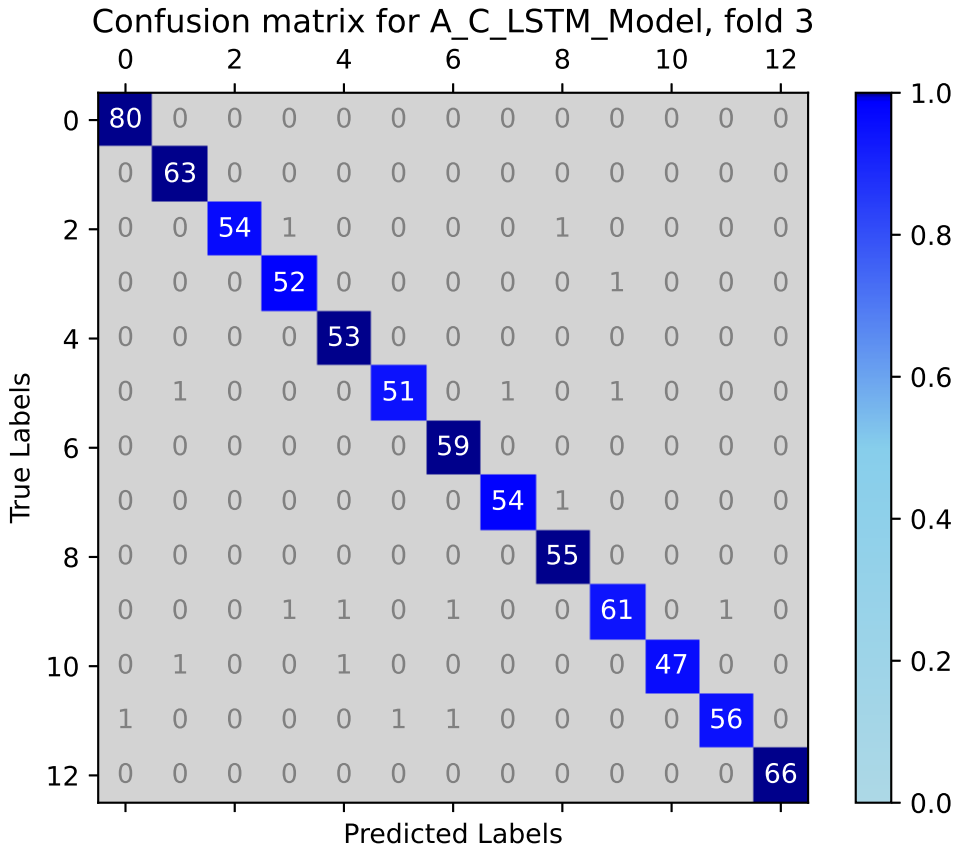
```
Epoch 31: val_accuracy did not improve from 0.97784
96/96 [==============================] - 20s 208ms/step - loss: 0.1387 - accuracy: 0.9547 - val_loss: 0.0890 - ...
   val_accuracy: 0.9648
Epoch 32/50
96/96 [==============================] - ETA: 0s - loss: 0.1008 - accuracy: 0.9625
Epoch 32: val_accuracy did not improve from 0.97784
96/96 [==============================] - 21s 220ms/step - loss: 0.1008 - accuracy: 0.9625 - val_loss: 0.0750 - ...
   val_accuracy: 0.9713
Epoch 33/50
96/96 [==============================] - ETA: 0s - loss: 0.1649 - accuracy: 0.9426
Epoch 33: val_accuracy did not improve from 0.97784
96/96 [==============================] - 21s 220ms/step - loss: 0.1649 - accuracy: 0.9426 - val_loss: 0.1305 - ...
   val_accuracy: 0.9557
Epoch 34/50
96/96 [==============================] - ETA: 0s - loss: 0.1375 - accuracy: 0.9547
Epoch 34: val_accuracy did not improve from 0.97784
96/96 [==============================] - 21s 221ms/step - loss: 0.1375 - accuracy: 0.9547 - val_loss: 0.1006 - ...
   val_accuracy: 0.9622
Epoch 35/50
96/96 [==============================] - ETA: 0s - loss: 0.1280 - accuracy: 0.9553
Epoch 35: val_accuracy did not improve from 0.97784
96/96 [==============================] - 21s 218ms/step - loss: 0.1280 - accuracy: 0.9553 - val_loss: 0.0999 - ...
   val_accuracy: 0.9635
Epoch 36/50
96/96 [==============================] - ETA: 0s - loss: 0.1103 - accuracy: 0.9605
Epoch 36: val_accuracy did not improve from 0.97784
96/96 [==============================] - 21s 215ms/step - loss: 0.1103 - accuracy: 0.9605 - val_loss: 0.1285 - ...
   val_accuracy: 0.9531
Epoch 37/50
96/96 [==============================] - ETA: 0s - loss: 0.1151 - accuracy: 0.9583
Epoch 37: val_accuracy did not improve from 0.97784
96/96 [==============================] - 20s 210ms/step - loss: 0.1151 - accuracy: 0.9583 - val_loss: 0.1159 - ...
   val_accuracy: 0.9622
Epoch 38/50
96/96 [==============================] - ETA: 0s - loss: 0.1072 - accuracy: 0.9632
Epoch 38: val_accuracy did not improve from 0.97784
96/96 [==============================] - 20s 210ms/step - loss: 0.1072 - accuracy: 0.9632 - val_loss: 0.0842 - ...
   val_accuracy: 0.9687
Epoch 39/50
```

```
96/96 [==============================] - ETA: 0s - loss: 0.0989 - accuracy: 0.9658
Epoch 39: val_accuracy did not improve from 0.97784
96/96 [==============================] - 20s 207ms/step - loss: 0.0989 - accuracy: 0.9658 - val_loss: 0.1076 - ...
   val_accuracy: 0.9609
Epoch 40/50
96/96 [==============================] - ETA: 0s - loss: 0.1078 - accuracy: 0.9599
Epoch 40: val_accuracy did not improve from 0.97784
96/96 [==============================] - 20s 209ms/step - loss: 0.1078 - accuracy: 0.9599 - val_loss: 0.0875 - ...
   val_accuracy: 0.9674
Epoch 41/50
96/96 [==============================] - ETA: 0s - loss: 0.1512 - accuracy: 0.9462
Epoch 41: val_accuracy improved from 0.97784 to 0.97914, saving model to ...
   /kaggle/working/A_C_LSTM_Model_Fold_3_Checkpoint.h5
96/96 [==============================] - 19s 201ms/step - loss: 0.1512 - accuracy: 0.9462 - val_loss: 0.0674 - ...
   val_accuracy: 0.9791
Epoch 42/50
96/96 [==============================] - ETA: 0s - loss: 0.1665 - accuracy: 0.9455
Epoch 42: val_accuracy did not improve from 0.97914
96/96 [==============================] - 20s 204ms/step - loss: 0.1665 - accuracy: 0.9455 - val_loss: 0.1371 - ...
   val_accuracy: 0.9557
Epoch 43/50
96/96 [==============================] - ETA: 0s - loss: 0.1346 - accuracy: 0.9527
Epoch 43: val_accuracy did not improve from 0.97914
96/96 [==============================] - 20s 212ms/step - loss: 0.1346 - accuracy: 0.9527 - val_loss: 0.1631 - ...
   val_accuracy: 0.9478
Epoch 44/50
96/96 [==============================] - ETA: 0s - loss: 0.2396 - accuracy: 0.9217
Epoch 44: val_accuracy did not improve from 0.97914
96/96 [==============================] - 20s 205ms/step - loss: 0.2396 - accuracy: 0.9217 - val_loss: 0.1604 - ...
   val_accuracy: 0.9557
Epoch 45/50
96/96 [==============================] - ETA: 0s - loss: 0.1527 - accuracy: 0.9462
Epoch 45: val_accuracy did not improve from 0.97914
96/96 [==============================] - 20s 204ms/step - loss: 0.1527 - accuracy: 0.9462 - val_loss: 0.1420 - ...
   val_accuracy: 0.9518
Epoch 46/50
96/96 [==============================] - ETA: 0s - loss: 0.1523 - accuracy: 0.9472
Epoch 46: val_accuracy did not improve from 0.97914
96/96 [==============================] - 20s 209ms/step - loss: 0.1523 - accuracy: 0.9472 - val_loss: 0.1266 - ...
```

```
   val_accuracy: 0.9492
Epoch 47/50
96/96 [==============================] - ETA: 0s - loss: 0.1430 - accuracy: 0.9482
Epoch 47: val_accuracy did not improve from 0.97914
96/96 [==============================] - 20s 205ms/step - loss: 0.1430 - accuracy: 0.9482 - val_loss: 0.1240 - ...
   val_accuracy: 0.9570
Epoch 48/50
96/96 [==============================] - ETA: 0s - loss: 0.1288 - accuracy: 0.9550
Epoch 48: val_accuracy did not improve from 0.97914
96/96 [==============================] - 21s 219ms/step - loss: 0.1288 - accuracy: 0.9550 - val_loss: 0.1021 - ...
   val_accuracy: 0.9687
Epoch 49/50
96/96 [==============================] - ETA: 0s - loss: 0.1225 - accuracy: 0.9540
Epoch 49: val_accuracy did not improve from 0.97914
96/96 [==============================] - 21s 216ms/step - loss: 0.1225 - accuracy: 0.9540 - val_loss: 0.1121 - ...
   val_accuracy: 0.9648
Epoch 50/50
96/96 [==============================] - ETA: 0s - loss: 0.1060 - accuracy: 0.9654
Epoch 50: val_accuracy did not improve from 0.97914
96/96 [==============================] - 20s 208ms/step - loss: 0.1060 - accuracy: 0.9654 - val_loss: 0.0978 - ...
   val_accuracy: 0.9635
```

```
24/24 [==============================] - 1s 20ms/step - loss: 0.0674 - accuracy: 0.9791
Test Loss A_C_LSTM_Model, fold 3: 0.06735911220312119, Test Accuracy A_C_LSTM_Model, fold 3: 0.979139506816864
24/24 [==============================] - 1s 18ms/step
F1 score for A_C_LSTM_Model, fold 3: 0.9785225793293548
24/24 [==============================] - 0s 18ms/step
```


Training History for A_C_LSTM_Model, fold 3
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy


Confusion matrix for A_C_LSTM_Model, fold 3
True Labels
Predicted Labels

```
Epoch 1/50
96/96 [==============================] - ETA: 0s - loss: 0.1189 - accuracy: 0.9570
Epoch 1: val_accuracy improved from -inf to 0.97128, saving model to /kaggle/working/A_C_LSTM_Model_Fold_4_Checkpoint.h5
96/96 [==============================] - 28s 217ms/step - loss: 0.1189 - accuracy: 0.9570 - val_loss: 0.0807 - ...
    val_accuracy: 0.9713
Epoch 2/50

/opt/conda/lib/python3.10/site-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an ...
    HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras ...
    format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(

96/96 [==============================] - ETA: 0s - loss: 0.1318 - accuracy: 0.9544
Epoch 2: val_accuracy did not improve from 0.97128
96/96 [==============================] - 20s 207ms/step - loss: 0.1318 - accuracy: 0.9544 - val_loss: 0.0924 - ...
    val_accuracy: 0.9713
Epoch 3/50
96/96 [==============================] - ETA: 0s - loss: 0.1023 - accuracy: 0.9635
Epoch 3: val_accuracy improved from 0.97128 to 0.97389, saving model to /kaggle/working/A_C_LSTM_Model_Fold_4_Checkpoint.h5
96/96 [==============================] - 20s 206ms/step - loss: 0.1023 - accuracy: 0.9635 - val_loss: 0.0791 - ...
    val_accuracy: 0.9739
Epoch 4/50
96/96 [==============================] - ETA: 0s - loss: 0.1310 - accuracy: 0.9514
Epoch 4: val_accuracy did not improve from 0.97389
96/96 [==============================] - 19s 200ms/step - loss: 0.1310 - accuracy: 0.9514 - val_loss: 0.1022 - ...
    val_accuracy: 0.9608
Epoch 5/50
96/96 [==============================] - ETA: 0s - loss: 0.1543 - accuracy: 0.9465
Epoch 5: val_accuracy did not improve from 0.97389
96/96 [==============================] - 19s 201ms/step - loss: 0.1543 - accuracy: 0.9465 - val_loss: 0.1046 - ...
    val_accuracy: 0.9569
Epoch 6/50
96/96 [==============================] - ETA: 0s - loss: 0.1974 - accuracy: 0.9358
Epoch 6: val_accuracy did not improve from 0.97389
96/96 [==============================] - 19s 200ms/step - loss: 0.1974 - accuracy: 0.9358 - val_loss: 0.2046 - ...
    val_accuracy: 0.9282
Epoch 7/50
96/96 [==============================] - ETA: 0s - loss: 0.1570 - accuracy: 0.9462
```

```
Epoch 7: val_accuracy did not improve from 0.97389
96/96 [==============================] - 19s 203ms/step - loss: 0.1570 - accuracy: 0.9462 - val_loss: 0.1339 - ...
   val_accuracy: 0.9504
Epoch 8/50
96/96 [==============================] - ETA: 0s - loss: 0.1271 - accuracy: 0.9580
Epoch 8: val_accuracy did not improve from 0.97389
96/96 [==============================] - 19s 202ms/step - loss: 0.1271 - accuracy: 0.9580 - val_loss: 0.1031 - ...
   val_accuracy: 0.9530
Epoch 9/50
96/96 [==============================] - ETA: 0s - loss: 0.1020 - accuracy: 0.9641
Epoch 9: val_accuracy did not improve from 0.97389
96/96 [==============================] - 19s 195ms/step - loss: 0.1020 - accuracy: 0.9641 - val_loss: 0.0866 - ...
   val_accuracy: 0.9661
Epoch 10/50
96/96 [==============================] - ETA: 0s - loss: 0.0977 - accuracy: 0.9632
Epoch 10: val_accuracy did not improve from 0.97389
96/96 [==============================] - 19s 198ms/step - loss: 0.0977 - accuracy: 0.9632 - val_loss: 0.1004 - ...
   val_accuracy: 0.9569
Epoch 11/50
96/96 [==============================] - ETA: 0s - loss: 0.1186 - accuracy: 0.9566
Epoch 11: val_accuracy did not improve from 0.97389
96/96 [==============================] - 20s 204ms/step - loss: 0.1186 - accuracy: 0.9566 - val_loss: 0.0977 - ...
   val_accuracy: 0.9543
Epoch 12/50
96/96 [==============================] - ETA: 0s - loss: 0.0987 - accuracy: 0.9638
Epoch 12: val_accuracy improved from 0.97389 to 0.97520, saving model to ...
   /kaggle/working/A_C_LSTM_Model_Fold_4_Checkpoint.h5
96/96 [==============================] - 20s 204ms/step - loss: 0.0987 - accuracy: 0.9638 - val_loss: 0.1019 - ...
   val_accuracy: 0.9752
Epoch 13/50
96/96 [==============================] - ETA: 0s - loss: 0.1058 - accuracy: 0.9609
Epoch 13: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 202ms/step - loss: 0.1058 - accuracy: 0.9609 - val_loss: 0.1023 - ...
   val_accuracy: 0.9608
Epoch 14/50
96/96 [==============================] - ETA: 0s - loss: 0.1477 - accuracy: 0.9456
Epoch 14: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 201ms/step - loss: 0.1477 - accuracy: 0.9456 - val_loss: 0.1002 - ...
   val_accuracy: 0.9582
```

```
Epoch 15/50
96/96 [==============================] - ETA: 0s - loss: 0.1378 - accuracy: 0.9511
Epoch 15: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 199ms/step - loss: 0.1378 - accuracy: 0.9511 - val_loss: 0.1425 - ...
   val_accuracy: 0.9439
Epoch 16/50
96/96 [==============================] - ETA: 0s - loss: 0.1091 - accuracy: 0.9586
Epoch 16: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 199ms/step - loss: 0.1091 - accuracy: 0.9586 - val_loss: 0.1441 - ...
   val_accuracy: 0.9439
Epoch 17/50
96/96 [==============================] - ETA: 0s - loss: 0.1070 - accuracy: 0.9632
Epoch 17: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 197ms/step - loss: 0.1070 - accuracy: 0.9632 - val_loss: 0.1304 - ...
   val_accuracy: 0.9465
Epoch 18/50
96/96 [==============================] - ETA: 0s - loss: 0.1026 - accuracy: 0.9658
Epoch 18: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 198ms/step - loss: 0.1026 - accuracy: 0.9658 - val_loss: 0.0842 - ...
   val_accuracy: 0.9648
Epoch 19/50
96/96 [==============================] - ETA: 0s - loss: 0.1374 - accuracy: 0.9524
Epoch 19: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 198ms/step - loss: 0.1374 - accuracy: 0.9524 - val_loss: 0.1469 - ...
   val_accuracy: 0.9413
Epoch 20/50
96/96 [==============================] - ETA: 0s - loss: 0.1530 - accuracy: 0.9430
Epoch 20: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 197ms/step - loss: 0.1530 - accuracy: 0.9430 - val_loss: 0.1570 - ...
   val_accuracy: 0.9556
Epoch 21/50
96/96 [==============================] - ETA: 0s - loss: 0.1370 - accuracy: 0.9488
Epoch 21: val_accuracy did not improve from 0.97520
96/96 [==============================] - 18s 191ms/step - loss: 0.1370 - accuracy: 0.9488 - val_loss: 0.1105 - ...
   val_accuracy: 0.9674
Epoch 22/50
96/96 [==============================] - ETA: 0s - loss: 0.1287 - accuracy: 0.9573
Epoch 22: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 193ms/step - loss: 0.1287 - accuracy: 0.9573 - val_loss: 0.1489 - ...
```

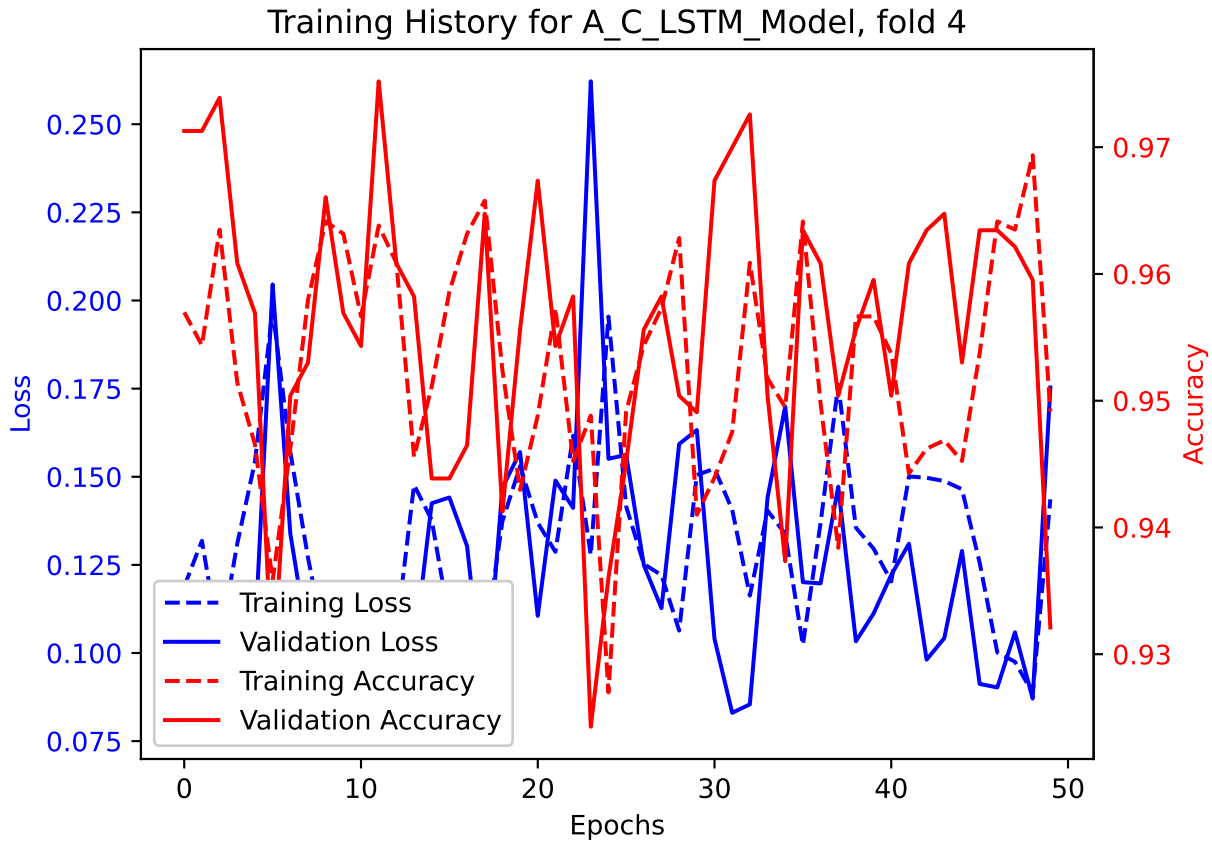
```
   val_accuracy: 0.9543
Epoch 23/50
96/96 [==============================] - ETA: 0s - loss: 0.1617 - accuracy: 0.9452
Epoch 23: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 194ms/step - loss: 0.1617 - accuracy: 0.9452 - val_loss: 0.1412 - ...
   val_accuracy: 0.9582
Epoch 24/50
96/96 [==============================] - ETA: 0s - loss: 0.1278 - accuracy: 0.9488
Epoch 24: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 199ms/step - loss: 0.1278 - accuracy: 0.9488 - val_loss: 0.2622 - ...
   val_accuracy: 0.9243
Epoch 25/50
96/96 [==============================] - ETA: 0s - loss: 0.1954 - accuracy: 0.9270
Epoch 25: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 198ms/step - loss: 0.1954 - accuracy: 0.9270 - val_loss: 0.1551 - ...
   val_accuracy: 0.9360
Epoch 26/50
96/96 [==============================] - ETA: 0s - loss: 0.1416 - accuracy: 0.9492
Epoch 26: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 195ms/step - loss: 0.1416 - accuracy: 0.9492 - val_loss: 0.1562 - ...
   val_accuracy: 0.9452
Epoch 27/50
96/96 [==============================] - ETA: 0s - loss: 0.1252 - accuracy: 0.9544
Epoch 27: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 198ms/step - loss: 0.1252 - accuracy: 0.9544 - val_loss: 0.1250 - ...
   val_accuracy: 0.9556
Epoch 28/50
96/96 [==============================] - ETA: 0s - loss: 0.1221 - accuracy: 0.9573
Epoch 28: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 198ms/step - loss: 0.1221 - accuracy: 0.9573 - val_loss: 0.1127 - ...
   val_accuracy: 0.9582
Epoch 29/50
96/96 [==============================] - ETA: 0s - loss: 0.1064 - accuracy: 0.9628
Epoch 29: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 194ms/step - loss: 0.1064 - accuracy: 0.9628 - val_loss: 0.1593 - ...
   val_accuracy: 0.9504
Epoch 30/50
96/96 [==============================] - ETA: 0s - loss: 0.1505 - accuracy: 0.9410
Epoch 30: val_accuracy did not improve from 0.97520
```

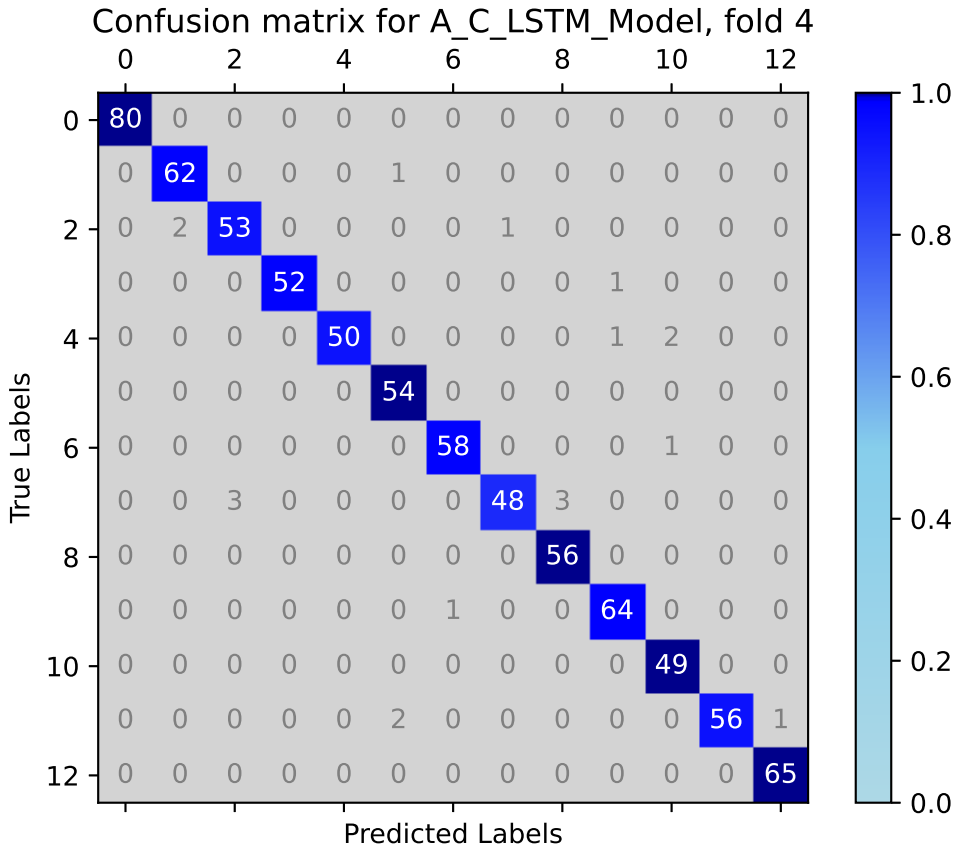
```
96/96 [==============================] - 19s 195ms/step - loss: 0.1505 - accuracy: 0.9410 - val_loss: 0.1633 - ...
    val_accuracy: 0.9491
Epoch 31/50
96/96 [==============================] - ETA: 0s - loss: 0.1523 - accuracy: 0.9439
Epoch 31: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 194ms/step - loss: 0.1523 - accuracy: 0.9439 - val_loss: 0.1040 - ...
    val_accuracy: 0.9674
Epoch 32/50
96/96 [==============================] - ETA: 0s - loss: 0.1406 - accuracy: 0.9475
Epoch 32: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 199ms/step - loss: 0.1406 - accuracy: 0.9475 - val_loss: 0.0830 - ...
    val_accuracy: 0.9700
Epoch 33/50
96/96 [==============================] - ETA: 0s - loss: 0.1163 - accuracy: 0.9609
Epoch 33: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 196ms/step - loss: 0.1163 - accuracy: 0.9609 - val_loss: 0.0854 - ...
    val_accuracy: 0.9726
Epoch 34/50
96/96 [==============================] - ETA: 0s - loss: 0.1404 - accuracy: 0.9518
Epoch 34: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 194ms/step - loss: 0.1404 - accuracy: 0.9518 - val_loss: 0.1440 - ...
    val_accuracy: 0.9504
Epoch 35/50
96/96 [==============================] - ETA: 0s - loss: 0.1338 - accuracy: 0.9495
Epoch 35: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 197ms/step - loss: 0.1338 - accuracy: 0.9495 - val_loss: 0.1699 - ...
    val_accuracy: 0.9373
Epoch 36/50
96/96 [==============================] - ETA: 0s - loss: 0.1023 - accuracy: 0.9641
Epoch 36: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 196ms/step - loss: 0.1023 - accuracy: 0.9641 - val_loss: 0.1201 - ...
    val_accuracy: 0.9634
Epoch 37/50
96/96 [==============================] - ETA: 0s - loss: 0.1368 - accuracy: 0.9498
Epoch 37: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 196ms/step - loss: 0.1368 - accuracy: 0.9498 - val_loss: 0.1197 - ...
    val_accuracy: 0.9608
Epoch 38/50
96/96 [==============================] - ETA: 0s - loss: 0.1761 - accuracy: 0.9384
```

```
Epoch 38: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 196ms/step - loss: 0.1761 - accuracy: 0.9384 - val_loss: 0.1472 - ...
   val_accuracy: 0.9504
Epoch 39/50
96/96 [==============================] - ETA: 0s - loss: 0.1355 - accuracy: 0.9566
Epoch 39: val_accuracy did not improve from 0.97520
96/96 [==============================] - 18s 193ms/step - loss: 0.1355 - accuracy: 0.9566 - val_loss: 0.1033 - ...
   val_accuracy: 0.9556
Epoch 40/50
96/96 [==============================] - ETA: 0s - loss: 0.1296 - accuracy: 0.9566
Epoch 40: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 198ms/step - loss: 0.1296 - accuracy: 0.9566 - val_loss: 0.1112 - ...
   val_accuracy: 0.9595
Epoch 41/50
96/96 [==============================] - ETA: 0s - loss: 0.1203 - accuracy: 0.9537
Epoch 41: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 196ms/step - loss: 0.1203 - accuracy: 0.9537 - val_loss: 0.1222 - ...
   val_accuracy: 0.9504
Epoch 42/50
96/96 [==============================] - ETA: 0s - loss: 0.1501 - accuracy: 0.9443
Epoch 42: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 196ms/step - loss: 0.1501 - accuracy: 0.9443 - val_loss: 0.1310 - ...
   val_accuracy: 0.9608
Epoch 43/50
96/96 [==============================] - ETA: 0s - loss: 0.1497 - accuracy: 0.9462
Epoch 43: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 199ms/step - loss: 0.1497 - accuracy: 0.9462 - val_loss: 0.0981 - ...
   val_accuracy: 0.9634
Epoch 44/50
96/96 [==============================] - ETA: 0s - loss: 0.1488 - accuracy: 0.9469
Epoch 44: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 195ms/step - loss: 0.1488 - accuracy: 0.9469 - val_loss: 0.1042 - ...
   val_accuracy: 0.9648
Epoch 45/50
96/96 [==============================] - ETA: 0s - loss: 0.1465 - accuracy: 0.9452
Epoch 45: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 197ms/step - loss: 0.1465 - accuracy: 0.9452 - val_loss: 0.1290 - ...
   val_accuracy: 0.9530
Epoch 46/50
```

```
96/96 [==============================] - ETA: 0s - loss: 0.1256 - accuracy: 0.9537
Epoch 46: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 196ms/step - loss: 0.1256 - accuracy: 0.9537 - val_loss: 0.0912 - ...
   val_accuracy: 0.9634
Epoch 47/50
96/96 [==============================] - ETA: 0s - loss: 0.1001 - accuracy: 0.9641
Epoch 47: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 194ms/step - loss: 0.1001 - accuracy: 0.9641 - val_loss: 0.0902 - ...
   val_accuracy: 0.9634
Epoch 48/50
96/96 [==============================] - ETA: 0s - loss: 0.0975 - accuracy: 0.9635
Epoch 48: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 200ms/step - loss: 0.0975 - accuracy: 0.9635 - val_loss: 0.1059 - ...
   val_accuracy: 0.9621
Epoch 49/50
96/96 [==============================] - ETA: 0s - loss: 0.0886 - accuracy: 0.9694
Epoch 49: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 195ms/step - loss: 0.0886 - accuracy: 0.9694 - val_loss: 0.0871 - ...
   val_accuracy: 0.9595
Epoch 50/50
96/96 [==============================] - ETA: 0s - loss: 0.1436 - accuracy: 0.9492
Epoch 50: val_accuracy did not improve from 0.97520
96/96 [==============================] - 19s 196ms/step - loss: 0.1436 - accuracy: 0.9492 - val_loss: 0.1753 - ...
   val_accuracy: 0.9321
```

```
24/24 [==============================] - 1s 18ms/step - loss: 0.1019 - accuracy: 0.9752
Test Loss A_C_LSTM_Model, fold 4: 0.10190760344266891, Test Accuracy A_C_LSTM_Model, fold 4: 0.9751958250999451
24/24 [==============================] - 1s 17ms/step
F1 score for A_C_LSTM_Model, fold 4: 0.9738313786126882
24/24 [==============================] - 0s 18ms/step
```


Training History for A_C_LSTM_Model, fold 4
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
0.250
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.97
0.96
0.95
0.94
0.93
0
10
20
30
40
50


Confusion matrix for A_C_LSTM_Model, fold 4
True Labels
Predicted Labels
0 2 4 6 8 10 12
0 80 0 0 0 0 0 0 0 0 0 0 0 0
1 0 62 0 0 0 1 0 0 0 0 0 0 0
2 0 2 53 0 0 0 0 1 0 0 0 0 0
3 0 0 0 52 0 0 0 0 0 1 0 0 0
4 0 0 0 0 50 0 0 0 0 1 2 0 0
5 0 0 0 0 0 54 0 0 0 0 0 0 0
6 0 0 0 0 0 0 58 0 0 0 1 0 0
7 0 0 3 0 0 0 0 48 3 0 0 0 0
8 0 0 0 0 0 0 0 0 56 0 0 0 0
9 0 0 0 0 0 0 1 0 0 64 0 0 0
10 0 0 0 0 0 0 0 0 0 0 49 0 0
11 0 0 0 0 0 2 0 0 0 0 0 56 1
12 0 0 0 0 0 0 0 0 0 0 0 0 65
1.0
0.8
0.6
0.4
0.2
0.0

RESULTS for A C LSTM Model

```
Average F1 Score for A_C_LSTM_Model: 0.9774748267649562
F1 scores for all folds for A_C_LSTM_Model: [0.9783559218526776, 0.9799765088683448, 0.9766877451617154, 0.9785225793293548, 0.97383137861
```

TCN Model

```
Epoch 1/50
96/96 [==============================] - ETA: 0s - loss: 0.0417 - accuracy: 0.9850
Epoch 1: val_accuracy improved from -inf to 0.97653, saving model to /kaggle/working/TCN_Model_Fold_0_Checkpoint.h5
96/96 [==============================] - 22s 40ms/step - loss: 0.0417 - accuracy: 0.9850 - val_loss: 0.0631 - ...
    val_accuracy: 0.9765
Epoch 2/50




/opt/conda/lib/python3.10/site-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an ...
    HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras ...
    format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(




94/96 [============================>.] - ETA: 0s - loss: 0.0131 - accuracy: 0.9970
Epoch 2: val_accuracy improved from 0.97653 to 1.00000, saving model to /kaggle/working/TCN_Model_Fold_0_Checkpoint.h5
```

```
96/96 [==============================] - 2s 25ms/step - loss: 0.0129 - accuracy: 0.9971 - val_loss: 0.0035 - ...
   val_accuracy: 1.0000
Epoch 3/50
94/96 [============================>.] - ETA: 0s - loss: 0.0019 - accuracy: 0.9997
Epoch 3: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 0.0018 - accuracy: 0.9997 - val_loss: 0.0016 - ...
   val_accuracy: 1.0000
Epoch 4/50
96/96 [==============================] - ETA: 0s - loss: 7.1036e-04 - accuracy: 1.0000
Epoch 4: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 7.1036e-04 - accuracy: 1.0000 - val_loss: 0.0015 - ...
   val_accuracy: 1.0000
Epoch 5/50
94/96 [============================>.] - ETA: 0s - loss: 5.2228e-04 - accuracy: 1.0000
Epoch 5: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 5.1551e-04 - accuracy: 1.0000 - val_loss: 0.0011 - ...
   val_accuracy: 1.0000
Epoch 6/50
94/96 [============================>.] - ETA: 0s - loss: 3.9168e-04 - accuracy: 1.0000
Epoch 6: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 3.9442e-04 - accuracy: 1.0000 - val_loss: 0.0011 - ...
   val_accuracy: 1.0000
Epoch 7/50
96/96 [==============================] - ETA: 0s - loss: 3.1362e-04 - accuracy: 1.0000
Epoch 7: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 3.1362e-04 - accuracy: 1.0000 - val_loss: 9.9233e-04 - ...
   val_accuracy: 1.0000
Epoch 8/50
94/96 [============================>.] - ETA: 0s - loss: 2.6209e-04 - accuracy: 1.0000
Epoch 8: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 2.6318e-04 - accuracy: 1.0000 - val_loss: 9.3743e-04 - ...
   val_accuracy: 1.0000
Epoch 9/50
95/96 [============================>.] - ETA: 0s - loss: 2.2398e-04 - accuracy: 1.0000
Epoch 9: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 2.2222e-04 - accuracy: 1.0000 - val_loss: 8.7691e-04 - ...
   val_accuracy: 1.0000
Epoch 10/50
94/96 [============================>.] - ETA: 0s - loss: 1.9219e-04 - accuracy: 1.0000
```

```
Epoch 10: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 1.8957e-04 - accuracy: 1.0000 - val_loss: 8.3908e-04 - ...
   val_accuracy: 1.0000
Epoch 11/50
94/96 [============================>.] - ETA: 0s - loss: 1.6824e-04 - accuracy: 1.0000
Epoch 11: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 1.6653e-04 - accuracy: 1.0000 - val_loss: 8.2186e-04 - ...
   val_accuracy: 1.0000
Epoch 12/50
94/96 [============================>.] - ETA: 0s - loss: 1.4848e-04 - accuracy: 1.0000
Epoch 12: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.4630e-04 - accuracy: 1.0000 - val_loss: 7.7871e-04 - ...
   val_accuracy: 1.0000
Epoch 13/50
96/96 [==============================] - ETA: 0s - loss: 1.2976e-04 - accuracy: 1.0000
Epoch 13: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.2976e-04 - accuracy: 1.0000 - val_loss: 7.3625e-04 - ...
   val_accuracy: 1.0000
Epoch 14/50
94/96 [============================>.] - ETA: 0s - loss: 1.1637e-04 - accuracy: 1.0000
Epoch 14: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 1.1509e-04 - accuracy: 1.0000 - val_loss: 6.9993e-04 - ...
   val_accuracy: 1.0000
Epoch 15/50
94/96 [============================>.] - ETA: 0s - loss: 1.0266e-04 - accuracy: 1.0000
Epoch 15: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 24ms/step - loss: 1.0287e-04 - accuracy: 1.0000 - val_loss: 7.1690e-04 - ...
   val_accuracy: 1.0000
Epoch 16/50
94/96 [============================>.] - ETA: 0s - loss: 9.2409e-05 - accuracy: 1.0000
Epoch 16: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 9.1623e-05 - accuracy: 1.0000 - val_loss: 6.8611e-04 - ...
   val_accuracy: 1.0000
Epoch 17/50
94/96 [============================>.] - ETA: 0s - loss: 8.3563e-05 - accuracy: 1.0000
Epoch 17: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 8.2290e-05 - accuracy: 1.0000 - val_loss: 6.5373e-04 - ...
   val_accuracy: 1.0000
Epoch 18/50
```

```
96/96 [==============================] - ETA: 0s - loss: 7.5610e-05 - accuracy: 1.0000
Epoch 18: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 7.5610e-05 - accuracy: 1.0000 - val_loss: 6.6043e-04 - ...
   val_accuracy: 1.0000
Epoch 19/50
94/96 [============================>.] - ETA: 0s - loss: 6.9073e-05 - accuracy: 1.0000
Epoch 19: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 6.8215e-05 - accuracy: 1.0000 - val_loss: 6.1732e-04 - ...
   val_accuracy: 1.0000
Epoch 20/50
94/96 [============================>.] - ETA: 0s - loss: 6.3580e-05 - accuracy: 1.0000
Epoch 20: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 6.2620e-05 - accuracy: 1.0000 - val_loss: 6.2595e-04 - ...
   val_accuracy: 1.0000
Epoch 21/50
96/96 [==============================] - ETA: 0s - loss: 5.7047e-05 - accuracy: 1.0000
Epoch 21: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 5.7047e-05 - accuracy: 1.0000 - val_loss: 6.0241e-04 - ...
   val_accuracy: 1.0000
Epoch 22/50
94/96 [============================>.] - ETA: 0s - loss: 5.0980e-05 - accuracy: 1.0000
Epoch 22: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 5.1799e-05 - accuracy: 1.0000 - val_loss: 5.9105e-04 - ...
   val_accuracy: 1.0000
Epoch 23/50
96/96 [==============================] - ETA: 0s - loss: 4.7826e-05 - accuracy: 1.0000
Epoch 23: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 4.7826e-05 - accuracy: 1.0000 - val_loss: 6.0322e-04 - ...
   val_accuracy: 1.0000
Epoch 24/50
94/96 [============================>.] - ETA: 0s - loss: 4.4342e-05 - accuracy: 1.0000
Epoch 24: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 4.3986e-05 - accuracy: 1.0000 - val_loss: 5.7600e-04 - ...
   val_accuracy: 1.0000
Epoch 25/50
94/96 [============================>.] - ETA: 0s - loss: 4.0752e-05 - accuracy: 1.0000
Epoch 25: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 4.0366e-05 - accuracy: 1.0000 - val_loss: 5.9744e-04 - ...
   val_accuracy: 1.0000
```

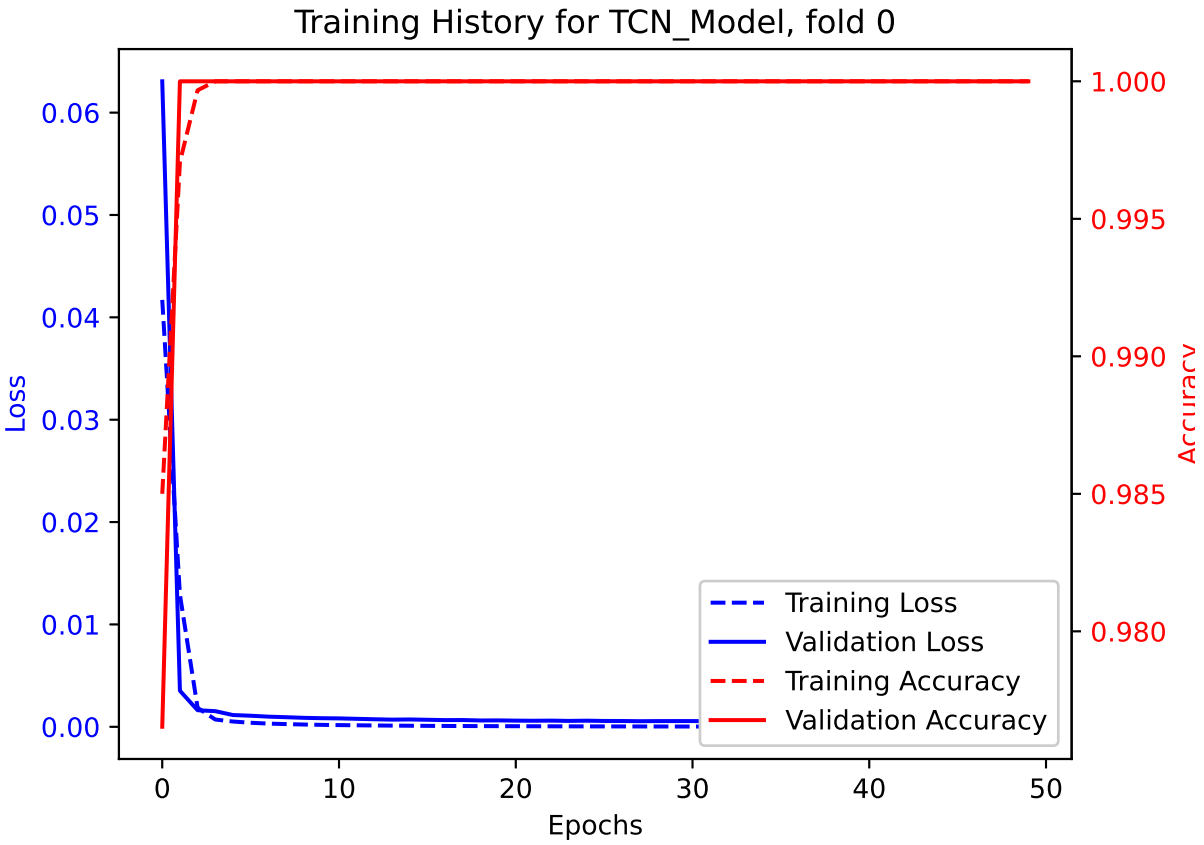
```
Epoch 26/50
94/96 [============================>.] - ETA: 0s - loss: 3.7969e-05 - accuracy: 1.0000
Epoch 26: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 3.7665e-05 - accuracy: 1.0000 - val_loss: 5.7158e-04 - ...
   val_accuracy: 1.0000
Epoch 27/50
95/96 [============================>.] - ETA: 0s - loss: 3.5031e-05 - accuracy: 1.0000
Epoch 27: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 3.4777e-05 - accuracy: 1.0000 - val_loss: 5.6710e-04 - ...
   val_accuracy: 1.0000
Epoch 28/50
94/96 [============================>.] - ETA: 0s - loss: 3.1884e-05 - accuracy: 1.0000
Epoch 28: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 3.2423e-05 - accuracy: 1.0000 - val_loss: 5.4738e-04 - ...
   val_accuracy: 1.0000
Epoch 29/50
95/96 [============================>.] - ETA: 0s - loss: 2.9554e-05 - accuracy: 1.0000
Epoch 29: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 24ms/step - loss: 2.9982e-05 - accuracy: 1.0000 - val_loss: 5.5808e-04 - ...
   val_accuracy: 1.0000
Epoch 30/50
95/96 [============================>.] - ETA: 0s - loss: 2.7702e-05 - accuracy: 1.0000
Epoch 30: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 2.7645e-05 - accuracy: 1.0000 - val_loss: 5.5990e-04 - ...
   val_accuracy: 1.0000
Epoch 31/50
94/96 [============================>.] - ETA: 0s - loss: 2.5897e-05 - accuracy: 1.0000
Epoch 31: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 2.5715e-05 - accuracy: 1.0000 - val_loss: 5.5884e-04 - ...
   val_accuracy: 1.0000
Epoch 32/50
96/96 [==============================] - ETA: 0s - loss: 2.4222e-05 - accuracy: 1.0000
Epoch 32: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 2.4222e-05 - accuracy: 1.0000 - val_loss: 5.4851e-04 - ...
   val_accuracy: 1.0000
Epoch 33/50
94/96 [============================>.] - ETA: 0s - loss: 2.2324e-05 - accuracy: 1.0000
Epoch 33: val_accuracy did not improve from 1.00000
```

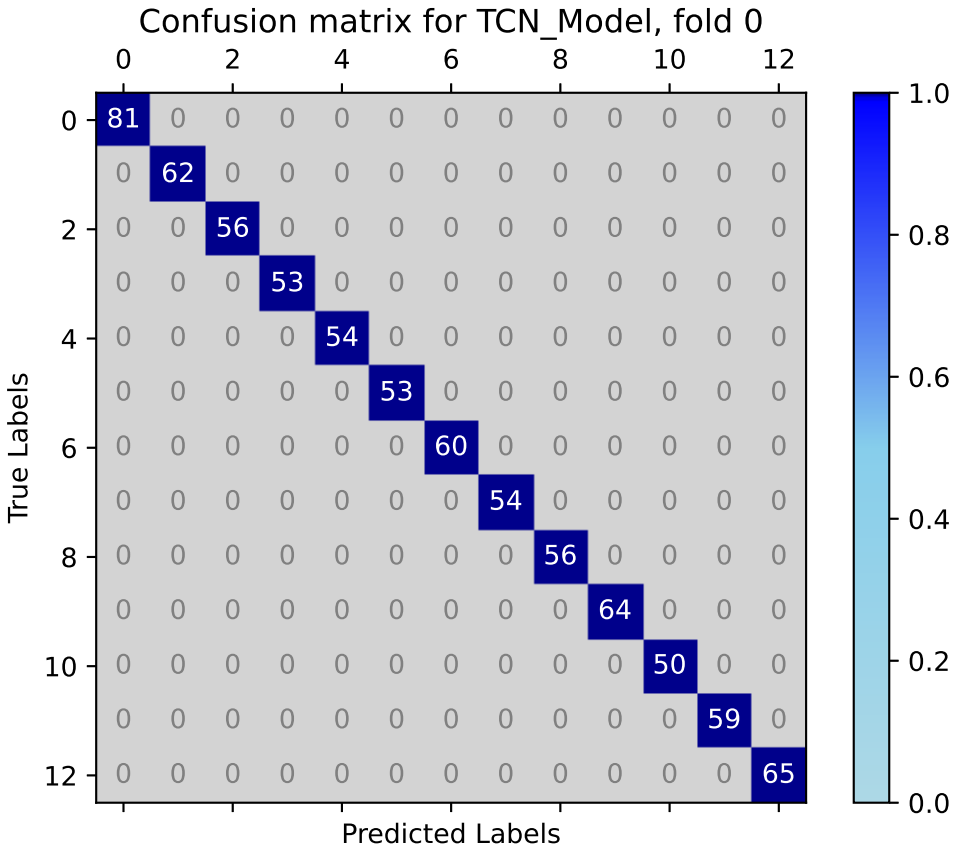
```
96/96 [==============================] - 2s 22ms/step - loss: 2.2413e-05 - accuracy: 1.0000 - val_loss: 5.3125e-04 - ...
   val_accuracy: 1.0000
Epoch 34/50
94/96 [============================>.] - ETA: 0s - loss: 2.0549e-05 - accuracy: 1.0000
Epoch 34: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 2.0751e-05 - accuracy: 1.0000 - val_loss: 5.4124e-04 - ...
   val_accuracy: 1.0000
Epoch 35/50
94/96 [============================>.] - ETA: 0s - loss: 1.9514e-05 - accuracy: 1.0000
Epoch 35: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.9430e-05 - accuracy: 1.0000 - val_loss: 5.4785e-04 - ...
   val_accuracy: 1.0000
Epoch 36/50
96/96 [==============================] - ETA: 0s - loss: 1.8057e-05 - accuracy: 1.0000
Epoch 36: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 1.8057e-05 - accuracy: 1.0000 - val_loss: 5.5123e-04 - ...
   val_accuracy: 1.0000
Epoch 37/50
94/96 [============================>.] - ETA: 0s - loss: 1.7015e-05 - accuracy: 1.0000
Epoch 37: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 1.6942e-05 - accuracy: 1.0000 - val_loss: 5.2688e-04 - ...
   val_accuracy: 1.0000
Epoch 38/50
96/96 [==============================] - ETA: 0s - loss: 1.5912e-05 - accuracy: 1.0000
Epoch 38: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 1.5912e-05 - accuracy: 1.0000 - val_loss: 5.2719e-04 - ...
   val_accuracy: 1.0000
Epoch 39/50
94/96 [============================>.] - ETA: 0s - loss: 1.4919e-05 - accuracy: 1.0000
Epoch 39: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.4854e-05 - accuracy: 1.0000 - val_loss: 5.3745e-04 - ...
   val_accuracy: 1.0000
Epoch 40/50
94/96 [============================>.] - ETA: 0s - loss: 1.4011e-05 - accuracy: 1.0000
Epoch 40: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 1.3918e-05 - accuracy: 1.0000 - val_loss: 5.4256e-04 - ...
   val_accuracy: 1.0000
Epoch 41/50
95/96 [============================>.] - ETA: 0s - loss: 1.3099e-05 - accuracy: 1.0000
```

```
Epoch 41: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.3053e-05 - accuracy: 1.0000 - val_loss: 5.5289e-04 - ...
   val_accuracy: 1.0000
Epoch 42/50
94/96 [============================>.] - ETA: 0s - loss: 1.2313e-05 - accuracy: 1.0000
Epoch 42: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 1.2255e-05 - accuracy: 1.0000 - val_loss: 5.5064e-04 - ...
   val_accuracy: 1.0000
Epoch 43/50
94/96 [============================>.] - ETA: 0s - loss: 1.1529e-05 - accuracy: 1.0000
Epoch 43: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.1434e-05 - accuracy: 1.0000 - val_loss: 5.5349e-04 - ...
   val_accuracy: 1.0000
Epoch 44/50
95/96 [============================>.] - ETA: 0s - loss: 1.0725e-05 - accuracy: 1.0000
Epoch 44: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 25ms/step - loss: 1.0780e-05 - accuracy: 1.0000 - val_loss: 5.4965e-04 - ...
   val_accuracy: 1.0000
Epoch 45/50
96/96 [==============================] - ETA: 0s - loss: 1.0125e-05 - accuracy: 1.0000
Epoch 45: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 1.0125e-05 - accuracy: 1.0000 - val_loss: 5.5254e-04 - ...
   val_accuracy: 1.0000
Epoch 46/50
95/96 [============================>.] - ETA: 0s - loss: 9.5352e-06 - accuracy: 1.0000
Epoch 46: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 9.5065e-06 - accuracy: 1.0000 - val_loss: 5.5900e-04 - ...
   val_accuracy: 1.0000
Epoch 47/50
94/96 [============================>.] - ETA: 0s - loss: 8.8168e-06 - accuracy: 1.0000
Epoch 47: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 8.9702e-06 - accuracy: 1.0000 - val_loss: 5.5612e-04 - ...
   val_accuracy: 1.0000
Epoch 48/50
94/96 [============================>.] - ETA: 0s - loss: 8.4033e-06 - accuracy: 1.0000
Epoch 48: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 8.4027e-06 - accuracy: 1.0000 - val_loss: 5.7278e-04 - ...
   val_accuracy: 1.0000
Epoch 49/50
```

```
94/96 [============================>.] - ETA: 0s - loss: 7.9598e-06 - accuracy: 1.0000
Epoch 49: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 7.9193e-06 - accuracy: 1.0000 - val_loss: 5.4910e-04 - ...
    val_accuracy: 1.0000
Epoch 50/50
96/96 [==============================] - ETA: 0s - loss: 7.4839e-06 - accuracy: 1.0000
Epoch 50: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 7.4839e-06 - accuracy: 1.0000 - val_loss: 5.4070e-04 - ...
    val_accuracy: 1.0000
```

```
24/24 [==============================] - 1s 6ms/step - loss: 0.0035 - accuracy: 1.0000
Test Loss TCN_Model, fold 0: 0.0035221746657043695, Test Accuracy TCN_Model, fold 0: 1.0
24/24 [==============================] - 2s 5ms/step
F1 score for TCN_Model, fold 0: 1.0
24/24 [==============================] - 0s 5ms/step
```


Training History for TCN_Model, fold 0
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
0.06
0.05
0.04
0.03
0.02
0.01
0.00
1.000
0.995
0.990
0.985
0.980
0
10
20
30
40
50


Confusion matrix for TCN_Model, fold 0
True Labels
Predicted Labels
0
2
4
6
8
10
12
1.0
0.8
0.6
0.4
0.2
0.0
81
62
56
53
54
53
60
54
56
64
50
59
65

```
Epoch 1/50
94/96 [============================>.] - ETA: 0s - loss: 0.0296 - accuracy: 0.9907
Epoch 1: val_accuracy improved from -inf to 0.99870, saving model to /kaggle/working/TCN_Model_Fold_1_Checkpoint.h5

/opt/conda/lib/python3.10/site-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an ...
    HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras ...
    format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(



96/96 [==============================] - 20s 35ms/step - loss: 0.0291 - accuracy: 0.9909 - val_loss: 0.0053 - ...
    val_accuracy: 0.9987
Epoch 2/50
96/96 [==============================] - ETA: 0s - loss: 0.0061 - accuracy: 0.9980
Epoch 2: val_accuracy did not improve from 0.99870
96/96 [==============================] - 2s 23ms/step - loss: 0.0061 - accuracy: 0.9980 - val_loss: 0.0241 - ...
    val_accuracy: 0.9922
Epoch 3/50
96/96 [==============================] - ETA: 0s - loss: 0.0497 - accuracy: 0.9847
Epoch 3: val_accuracy did not improve from 0.99870
96/96 [==============================] - 2s 23ms/step - loss: 0.0497 - accuracy: 0.9847 - val_loss: 0.1063 - ...
    val_accuracy: 0.9726
Epoch 4/50
94/96 [============================>.] - ETA: 0s - loss: 0.0706 - accuracy: 0.9801
Epoch 4: val_accuracy did not improve from 0.99870
96/96 [==============================] - 2s 23ms/step - loss: 0.0694 - accuracy: 0.9804 - val_loss: 0.0169 - ...
    val_accuracy: 0.9922
Epoch 5/50
94/96 [============================>.] - ETA: 0s - loss: 0.0316 - accuracy: 0.9894
Epoch 5: val_accuracy did not improve from 0.99870
96/96 [==============================] - 2s 23ms/step - loss: 0.0321 - accuracy: 0.9892 - val_loss: 0.0286 - ...
    val_accuracy: 0.9870
Epoch 6/50
95/96 [============================>.] - ETA: 0s - loss: 0.0141 - accuracy: 0.9954
Epoch 6: val_accuracy did not improve from 0.99870
96/96 [==============================] - 2s 23ms/step - loss: 0.0140 - accuracy: 0.9954 - val_loss: 0.0163 - ...
    val_accuracy: 0.9961
```

```
Epoch 7/50
95/96 [============================>.] - ETA: 0s - loss: 0.0098 - accuracy: 0.9974
Epoch 7: val_accuracy improved from 0.99870 to 1.00000, saving model to /kaggle/working/TCN_Model_Fold_1_Checkpoint.h5
96/96 [==============================] - 2s 26ms/step - loss: 0.0097 - accuracy: 0.9974 - val_loss: 0.0033 - ...
   val_accuracy: 1.0000
Epoch 8/50
94/96 [============================>.] - ETA: 0s - loss: 0.0011 - accuracy: 1.0000
Epoch 8: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0017 - ...
   val_accuracy: 1.0000
Epoch 9/50
94/96 [============================>.] - ETA: 0s - loss: 7.2268e-04 - accuracy: 1.0000
Epoch 9: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 7.1469e-04 - accuracy: 1.0000 - val_loss: 0.0014 - ...
   val_accuracy: 1.0000
Epoch 10/50
94/96 [============================>.] - ETA: 0s - loss: 5.7997e-04 - accuracy: 1.0000
Epoch 10: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 5.7148e-04 - accuracy: 1.0000 - val_loss: 0.0012 - ...
   val_accuracy: 1.0000
Epoch 11/50
96/96 [==============================] - ETA: 0s - loss: 4.7001e-04 - accuracy: 1.0000
Epoch 11: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 25ms/step - loss: 4.7001e-04 - accuracy: 1.0000 - val_loss: 0.0011 - ...
   val_accuracy: 1.0000
Epoch 12/50
94/96 [============================>.] - ETA: 0s - loss: 4.0021e-04 - accuracy: 1.0000
Epoch 12: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 3.9686e-04 - accuracy: 1.0000 - val_loss: 9.7906e-04 - ...
   val_accuracy: 1.0000
Epoch 13/50
94/96 [============================>.] - ETA: 0s - loss: 3.3933e-04 - accuracy: 1.0000
Epoch 13: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 3.4367e-04 - accuracy: 1.0000 - val_loss: 8.8015e-04 - ...
   val_accuracy: 1.0000
Epoch 14/50
94/96 [============================>.] - ETA: 0s - loss: 3.0014e-04 - accuracy: 1.0000
Epoch 14: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 2.9724e-04 - accuracy: 1.0000 - val_loss: 7.9602e-04 - ...
```

```
   val_accuracy: 1.0000
Epoch 15/50
96/96 [==============================] - ETA: 0s - loss: 2.6018e-04 - accuracy: 1.0000
Epoch 15: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 24ms/step - loss: 2.6018e-04 - accuracy: 1.0000 - val_loss: 7.3151e-04 - ...
   val_accuracy: 1.0000
Epoch 16/50
96/96 [==============================] - ETA: 0s - loss: 2.3179e-04 - accuracy: 1.0000
Epoch 16: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 2.3179e-04 - accuracy: 1.0000 - val_loss: 6.9322e-04 - ...
   val_accuracy: 1.0000
Epoch 17/50
94/96 [============================>.] - ETA: 0s - loss: 2.0823e-04 - accuracy: 1.0000
Epoch 17: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 2.0897e-04 - accuracy: 1.0000 - val_loss: 6.4517e-04 - ...
   val_accuracy: 1.0000
Epoch 18/50
94/96 [============================>.] - ETA: 0s - loss: 1.8831e-04 - accuracy: 1.0000
Epoch 18: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.8613e-04 - accuracy: 1.0000 - val_loss: 6.1448e-04 - ...
   val_accuracy: 1.0000
Epoch 19/50
94/96 [============================>.] - ETA: 0s - loss: 1.6402e-04 - accuracy: 1.0000
Epoch 19: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.6701e-04 - accuracy: 1.0000 - val_loss: 5.7246e-04 - ...
   val_accuracy: 1.0000
Epoch 20/50
94/96 [============================>.] - ETA: 0s - loss: 1.5144e-04 - accuracy: 1.0000
Epoch 20: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.5031e-04 - accuracy: 1.0000 - val_loss: 5.3326e-04 - ...
   val_accuracy: 1.0000
Epoch 21/50
94/96 [============================>.] - ETA: 0s - loss: 1.3802e-04 - accuracy: 1.0000
Epoch 21: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.3769e-04 - accuracy: 1.0000 - val_loss: 5.1737e-04 - ...
   val_accuracy: 1.0000
Epoch 22/50
94/96 [============================>.] - ETA: 0s - loss: 1.2582e-04 - accuracy: 1.0000
Epoch 22: val_accuracy did not improve from 1.00000
```

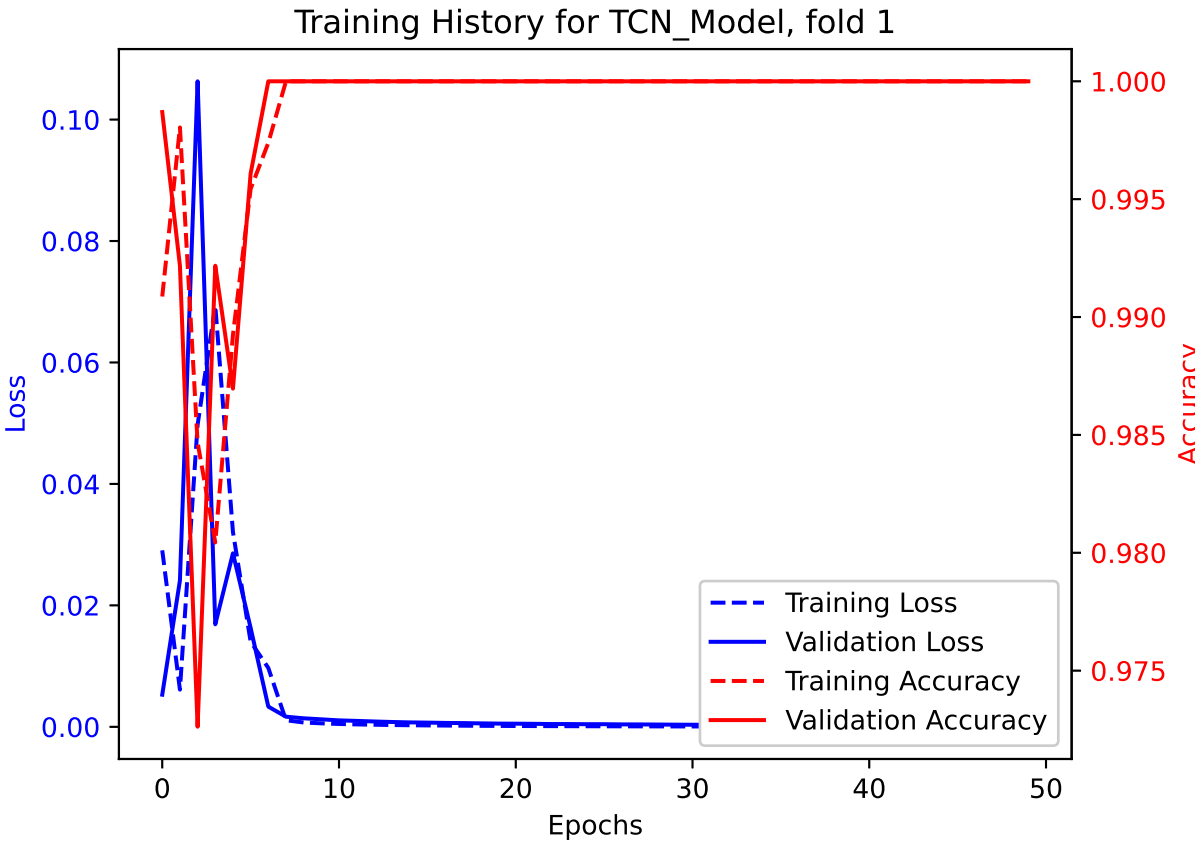
```
96/96 [==============================] - 2s 23ms/step - loss: 1.2522e-04 - accuracy: 1.0000 - val_loss: 4.9611e-04 - ...
    val_accuracy: 1.0000
Epoch 23/50
94/96 [============================>.] - ETA: 0s - loss: 1.1387e-04 - accuracy: 1.0000
Epoch 23: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.1431e-04 - accuracy: 1.0000 - val_loss: 4.5726e-04 - ...
    val_accuracy: 1.0000
Epoch 24/50
94/96 [============================>.] - ETA: 0s - loss: 1.0456e-04 - accuracy: 1.0000
Epoch 24: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.0504e-04 - accuracy: 1.0000 - val_loss: 4.5800e-04 - ...
    val_accuracy: 1.0000
Epoch 25/50
95/96 [============================>.] - ETA: 0s - loss: 9.6736e-05 - accuracy: 1.0000
Epoch 25: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 9.6428e-05 - accuracy: 1.0000 - val_loss: 4.3338e-04 - ...
    val_accuracy: 1.0000
Epoch 26/50
94/96 [============================>.] - ETA: 0s - loss: 8.9086e-05 - accuracy: 1.0000
Epoch 26: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 8.8490e-05 - accuracy: 1.0000 - val_loss: 4.3221e-04 - ...
    val_accuracy: 1.0000
Epoch 27/50
94/96 [============================>.] - ETA: 0s - loss: 8.0258e-05 - accuracy: 1.0000
Epoch 27: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 8.1858e-05 - accuracy: 1.0000 - val_loss: 4.0118e-04 - ...
    val_accuracy: 1.0000
Epoch 28/50
94/96 [============================>.] - ETA: 0s - loss: 7.3053e-05 - accuracy: 1.0000
Epoch 28: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 7.5754e-05 - accuracy: 1.0000 - val_loss: 3.9564e-04 - ...
    val_accuracy: 1.0000
Epoch 29/50
96/96 [==============================] - ETA: 0s - loss: 7.0887e-05 - accuracy: 1.0000
Epoch 29: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 24ms/step - loss: 7.0887e-05 - accuracy: 1.0000 - val_loss: 3.7772e-04 - ...
    val_accuracy: 1.0000
Epoch 30/50
94/96 [============================>.] - ETA: 0s - loss: 6.4647e-05 - accuracy: 1.0000
```

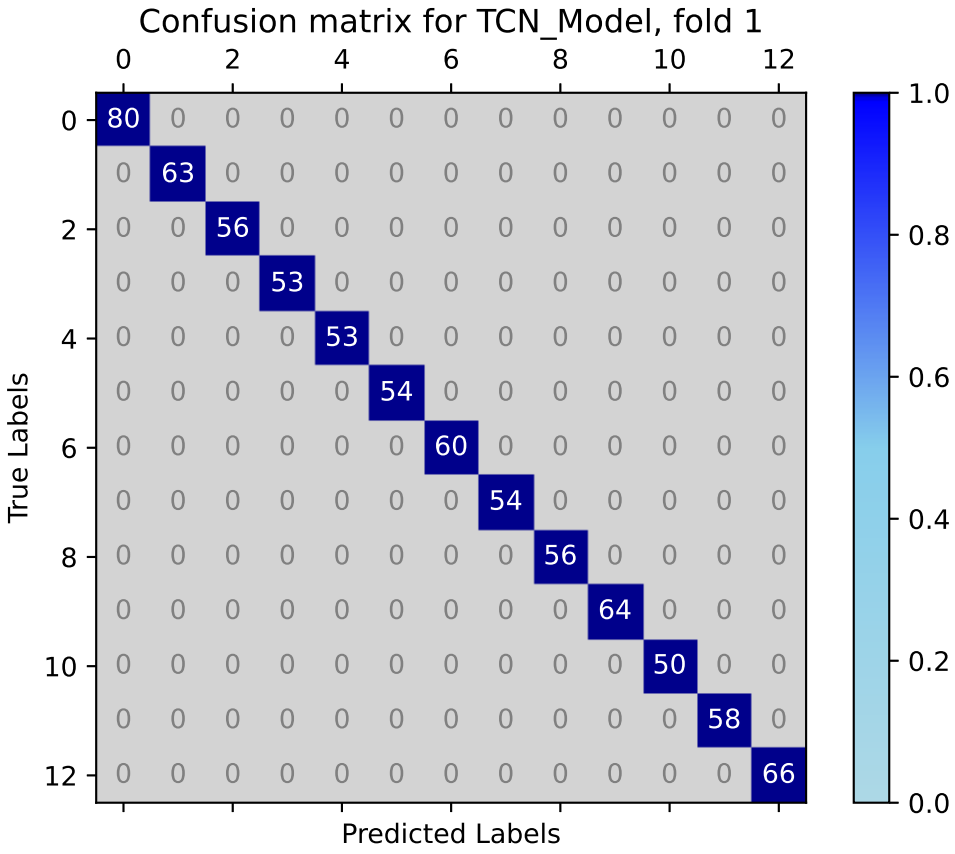
```
Epoch 30: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 6.4755e-05 - accuracy: 1.0000 - val_loss: 3.5835e-04 - ...
   val_accuracy: 1.0000
Epoch 31/50
94/96 [============================>.] - ETA: 0s - loss: 6.0292e-05 - accuracy: 1.0000
Epoch 31: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 6.0545e-05 - accuracy: 1.0000 - val_loss: 3.4358e-04 - ...
   val_accuracy: 1.0000
Epoch 32/50
94/96 [============================>.] - ETA: 0s - loss: 5.6579e-05 - accuracy: 1.0000
Epoch 32: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 5.6459e-05 - accuracy: 1.0000 - val_loss: 3.4494e-04 - ...
   val_accuracy: 1.0000
Epoch 33/50
94/96 [============================>.] - ETA: 0s - loss: 5.3034e-05 - accuracy: 1.0000
Epoch 33: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 5.2908e-05 - accuracy: 1.0000 - val_loss: 3.2310e-04 - ...
   val_accuracy: 1.0000
Epoch 34/50
95/96 [============================>.] - ETA: 0s - loss: 4.8910e-05 - accuracy: 1.0000
Epoch 34: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 4.8890e-05 - accuracy: 1.0000 - val_loss: 3.2270e-04 - ...
   val_accuracy: 1.0000
Epoch 35/50
95/96 [============================>.] - ETA: 0s - loss: 4.6081e-05 - accuracy: 1.0000
Epoch 35: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 4.5735e-05 - accuracy: 1.0000 - val_loss: 3.1771e-04 - ...
   val_accuracy: 1.0000
Epoch 36/50
94/96 [============================>.] - ETA: 0s - loss: 4.3097e-05 - accuracy: 1.0000
Epoch 36: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 4.2802e-05 - accuracy: 1.0000 - val_loss: 3.0055e-04 - ...
   val_accuracy: 1.0000
Epoch 37/50
94/96 [============================>.] - ETA: 0s - loss: 3.9835e-05 - accuracy: 1.0000
Epoch 37: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 4.0053e-05 - accuracy: 1.0000 - val_loss: 2.9361e-04 - ...
   val_accuracy: 1.0000
Epoch 38/50
```

```
94/96 [============================>.] - ETA: 0s - loss: 3.7264e-05 - accuracy: 1.0000
Epoch 38: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 3.7487e-05 - accuracy: 1.0000 - val_loss: 2.8917e-04 - ...
   val_accuracy: 1.0000
Epoch 39/50
96/96 [==============================] - ETA: 0s - loss: 3.5237e-05 - accuracy: 1.0000
Epoch 39: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 3.5237e-05 - accuracy: 1.0000 - val_loss: 2.7583e-04 - ...
   val_accuracy: 1.0000
Epoch 40/50
95/96 [============================>.] - ETA: 0s - loss: 3.3047e-05 - accuracy: 1.0000
Epoch 40: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 3.2985e-05 - accuracy: 1.0000 - val_loss: 2.7222e-04 - ...
   val_accuracy: 1.0000
Epoch 41/50
96/96 [==============================] - ETA: 0s - loss: 3.0872e-05 - accuracy: 1.0000
Epoch 41: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 3.0872e-05 - accuracy: 1.0000 - val_loss: 2.6519e-04 - ...
   val_accuracy: 1.0000
Epoch 42/50
96/96 [==============================] - ETA: 0s - loss: 2.8946e-05 - accuracy: 1.0000
Epoch 42: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 24ms/step - loss: 2.8946e-05 - accuracy: 1.0000 - val_loss: 2.6275e-04 - ...
   val_accuracy: 1.0000
Epoch 43/50
95/96 [============================>.] - ETA: 0s - loss: 2.7117e-05 - accuracy: 1.0000
Epoch 43: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 2.7232e-05 - accuracy: 1.0000 - val_loss: 2.4764e-04 - ...
   val_accuracy: 1.0000
Epoch 44/50
94/96 [============================>.] - ETA: 0s - loss: 2.5558e-05 - accuracy: 1.0000
Epoch 44: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 2.5533e-05 - accuracy: 1.0000 - val_loss: 2.3924e-04 - ...
   val_accuracy: 1.0000
Epoch 45/50
94/96 [============================>.] - ETA: 0s - loss: 2.3573e-05 - accuracy: 1.0000
Epoch 45: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 2.4086e-05 - accuracy: 1.0000 - val_loss: 2.3887e-04 - ...
   val_accuracy: 1.0000
```

```
Epoch 46/50
94/96 [============================>.] - ETA: 0s - loss: 2.2978e-05 - accuracy: 1.0000
Epoch 46: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 2.2657e-05 - accuracy: 1.0000 - val_loss: 2.3193e-04 - ...
   val_accuracy: 1.0000
Epoch 47/50
94/96 [============================>.] - ETA: 0s - loss: 2.1127e-05 - accuracy: 1.0000
Epoch 47: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 2.1222e-05 - accuracy: 1.0000 - val_loss: 2.3825e-04 - ...
   val_accuracy: 1.0000
Epoch 48/50
95/96 [============================>.] - ETA: 0s - loss: 2.0162e-05 - accuracy: 1.0000
Epoch 48: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 2.0041e-05 - accuracy: 1.0000 - val_loss: 2.3365e-04 - ...
   val_accuracy: 1.0000
Epoch 49/50
94/96 [============================>.] - ETA: 0s - loss: 1.9085e-05 - accuracy: 1.0000
Epoch 49: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.8832e-05 - accuracy: 1.0000 - val_loss: 2.1813e-04 - ...
   val_accuracy: 1.0000
Epoch 50/50
94/96 [============================>.] - ETA: 0s - loss: 1.7617e-05 - accuracy: 1.0000
Epoch 50: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.7759e-05 - accuracy: 1.0000 - val_loss: 2.2015e-04 - ...
   val_accuracy: 1.0000
```

```
24/24 [==============================] - 1s 6ms/step - loss: 0.0033 - accuracy: 1.0000
Test Loss TCN_Model, fold 1: 0.0033277326729148626, Test Accuracy TCN_Model, fold 1: 1.0
24/24 [==============================] - 1s 5ms/step
F1 score for TCN_Model, fold 1: 1.0
24/24 [==============================] - 0s 5ms/step
```


Training History for TCN_Model, fold 1
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
0.00
0.02
0.04
0.06
0.08
0.10
0.975
0.980
0.985
0.990
0.995
1.000
0
10
20
30
40
50


Confusion matrix for TCN_Model, fold 1
True Labels
Predicted Labels
0
2
4
6
8
10
12
80
63
56
53
53
54
60
54
56
64
50
58
66
1.0
0.8
0.6
0.4
0.2
0.0

```
Epoch 1/50
96/96 [==============================] - ETA: 0s - loss: 0.0454 - accuracy: 0.9870
Epoch 1: val_accuracy improved from -inf to 0.98566, saving model to /kaggle/working/TCN_Model_Fold_2_Checkpoint.h5



/opt/conda/lib/python3.10/site-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an ...
   HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras ...
   format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(



96/96 [==============================] - 21s 35ms/step - loss: 0.0454 - accuracy: 0.9870 - val_loss: 0.0342 - ...
   val_accuracy: 0.9857
Epoch 2/50
94/96 [============================>.] - ETA: 0s - loss: 0.0234 - accuracy: 0.9910
Epoch 2: val_accuracy improved from 0.98566 to 0.99087, saving model to /kaggle/working/TCN_Model_Fold_2_Checkpoint.h5
96/96 [==============================] - 2s 25ms/step - loss: 0.0233 - accuracy: 0.9909 - val_loss: 0.0252 - ...
   val_accuracy: 0.9909
Epoch 3/50
94/96 [============================>.] - ETA: 0s - loss: 0.0208 - accuracy: 0.9937
Epoch 3: val_accuracy improved from 0.99087 to 0.99870, saving model to /kaggle/working/TCN_Model_Fold_2_Checkpoint.h5
96/96 [==============================] - 2s 25ms/step - loss: 0.0204 - accuracy: 0.9938 - val_loss: 0.0157 - ...
   val_accuracy: 0.9987
Epoch 4/50
94/96 [============================>.] - ETA: 0s - loss: 0.0034 - accuracy: 0.9997
Epoch 4: val_accuracy did not improve from 0.99870
96/96 [==============================] - 2s 23ms/step - loss: 0.0035 - accuracy: 0.9997 - val_loss: 0.0048 - ...
   val_accuracy: 0.9987
Epoch 5/50
96/96 [==============================] - ETA: 0s - loss: 0.0034 - accuracy: 0.9993
Epoch 5: val_accuracy did not improve from 0.99870
96/96 [==============================] - 2s 23ms/step - loss: 0.0034 - accuracy: 0.9993 - val_loss: 0.0033 - ...
   val_accuracy: 0.9987
Epoch 6/50
```

```
94/96 [============================>.] - ETA: 0s - loss: 0.0011 - accuracy: 1.0000
Epoch 6: val_accuracy improved from 0.99870 to 1.00000, saving model to /kaggle/working/TCN_Model_Fold_2_Checkpoint.h5
96/96 [==============================] - 2s 25ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0027 - ...
   val_accuracy: 1.0000
Epoch 7/50
94/96 [============================>.] - ETA: 0s - loss: 5.7109e-04 - accuracy: 1.0000
Epoch 7: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 5.7040e-04 - accuracy: 1.0000 - val_loss: 0.0015 - ...
   val_accuracy: 1.0000
Epoch 8/50
94/96 [============================>.] - ETA: 0s - loss: 2.8384e-04 - accuracy: 1.0000
Epoch 8: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 2.7889e-04 - accuracy: 1.0000 - val_loss: 0.0012 - ...
   val_accuracy: 1.0000
Epoch 9/50
95/96 [============================>.] - ETA: 0s - loss: 2.2709e-04 - accuracy: 1.0000
Epoch 9: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 25ms/step - loss: 2.2857e-04 - accuracy: 1.0000 - val_loss: 0.0011 - ...
   val_accuracy: 1.0000
Epoch 10/50
94/96 [============================>.] - ETA: 0s - loss: 1.9540e-04 - accuracy: 1.0000
Epoch 10: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.9444e-04 - accuracy: 1.0000 - val_loss: 0.0010 - ...
   val_accuracy: 1.0000
Epoch 11/50
94/96 [============================>.] - ETA: 0s - loss: 1.6686e-04 - accuracy: 1.0000
Epoch 11: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.6723e-04 - accuracy: 1.0000 - val_loss: 9.6719e-04 - ...
   val_accuracy: 1.0000
Epoch 12/50
94/96 [============================>.] - ETA: 0s - loss: 1.4774e-04 - accuracy: 1.0000
Epoch 12: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.4720e-04 - accuracy: 1.0000 - val_loss: 8.8144e-04 - ...
   val_accuracy: 1.0000
Epoch 13/50
94/96 [============================>.] - ETA: 0s - loss: 1.2926e-04 - accuracy: 1.0000
Epoch 13: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.2950e-04 - accuracy: 1.0000 - val_loss: 8.1669e-04 - ...
   val_accuracy: 1.0000
```

```
Epoch 14/50
96/96 [==============================] - ETA: 0s - loss: 1.1555e-04 - accuracy: 1.0000
Epoch 14: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.1555e-04 - accuracy: 1.0000 - val_loss: 7.8627e-04 - ...
   val_accuracy: 1.0000
Epoch 15/50
94/96 [===========================>.] - ETA: 0s - loss: 1.0403e-04 - accuracy: 1.0000
Epoch 15: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.0394e-04 - accuracy: 1.0000 - val_loss: 7.1506e-04 - ...
   val_accuracy: 1.0000
Epoch 16/50
94/96 [===========================>.] - ETA: 0s - loss: 9.4301e-05 - accuracy: 1.0000
Epoch 16: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 9.3423e-05 - accuracy: 1.0000 - val_loss: 6.9700e-04 - ...
   val_accuracy: 1.0000
Epoch 17/50
94/96 [===========================>.] - ETA: 0s - loss: 8.4265e-05 - accuracy: 1.0000
Epoch 17: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 8.4245e-05 - accuracy: 1.0000 - val_loss: 6.6805e-04 - ...
   val_accuracy: 1.0000
Epoch 18/50
95/96 [============================>.] - ETA: 0s - loss: 7.6909e-05 - accuracy: 1.0000
Epoch 18: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 7.6424e-05 - accuracy: 1.0000 - val_loss: 6.2471e-04 - ...
   val_accuracy: 1.0000
Epoch 19/50
94/96 [===========================>.] - ETA: 0s - loss: 7.0536e-05 - accuracy: 1.0000
Epoch 19: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 6.9691e-05 - accuracy: 1.0000 - val_loss: 5.8638e-04 - ...
   val_accuracy: 1.0000
Epoch 20/50
94/96 [===========================>.] - ETA: 0s - loss: 6.3338e-05 - accuracy: 1.0000
Epoch 20: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 6.3694e-05 - accuracy: 1.0000 - val_loss: 5.6360e-04 - ...
   val_accuracy: 1.0000
Epoch 21/50
94/96 [===========================>.] - ETA: 0s - loss: 5.8536e-05 - accuracy: 1.0000
Epoch 21: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 5.7939e-05 - accuracy: 1.0000 - val_loss: 5.4835e-04 - ...
```

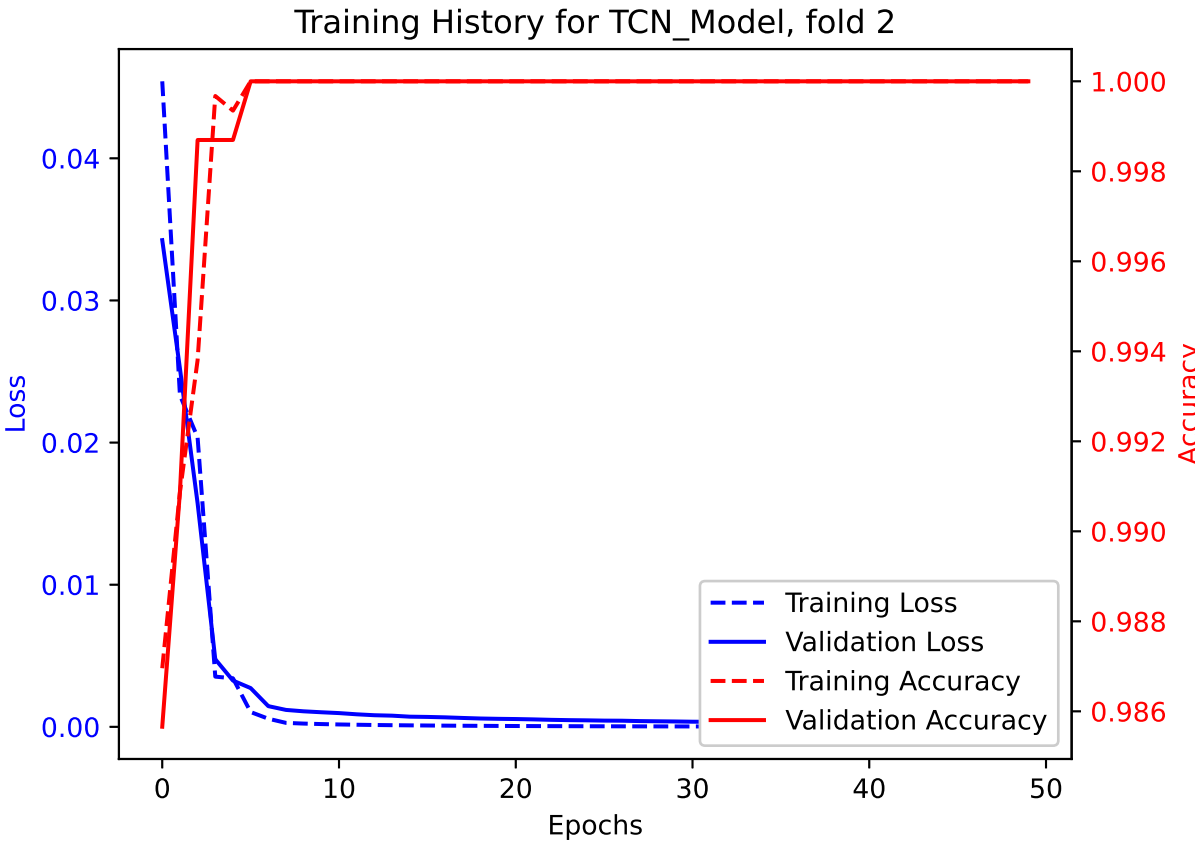
```
   val_accuracy: 1.0000
Epoch 22/50
94/96 [============================>.] - ETA: 0s - loss: 5.3247e-05 - accuracy: 1.0000
Epoch 22: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 5.3280e-05 - accuracy: 1.0000 - val_loss: 5.2407e-04 - ...
   val_accuracy: 1.0000
Epoch 23/50
96/96 [==============================] - ETA: 0s - loss: 4.9240e-05 - accuracy: 1.0000
Epoch 23: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 4.9240e-05 - accuracy: 1.0000 - val_loss: 4.9255e-04 - ...
   val_accuracy: 1.0000
Epoch 24/50
94/96 [============================>.] - ETA: 0s - loss: 4.6101e-05 - accuracy: 1.0000
Epoch 24: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 4.5395e-05 - accuracy: 1.0000 - val_loss: 4.7118e-04 - ...
   val_accuracy: 1.0000
Epoch 25/50
94/96 [============================>.] - ETA: 0s - loss: 4.2028e-05 - accuracy: 1.0000
Epoch 25: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 4.1531e-05 - accuracy: 1.0000 - val_loss: 4.5744e-04 - ...
   val_accuracy: 1.0000
Epoch 26/50
96/96 [==============================] - ETA: 0s - loss: 3.8550e-05 - accuracy: 1.0000
Epoch 26: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 3.8550e-05 - accuracy: 1.0000 - val_loss: 4.3780e-04 - ...
   val_accuracy: 1.0000
Epoch 27/50
94/96 [============================>.] - ETA: 0s - loss: 3.4546e-05 - accuracy: 1.0000
Epoch 27: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 3.5521e-05 - accuracy: 1.0000 - val_loss: 4.3094e-04 - ...
   val_accuracy: 1.0000
Epoch 28/50
96/96 [==============================] - ETA: 0s - loss: 3.2882e-05 - accuracy: 1.0000
Epoch 28: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 3.2882e-05 - accuracy: 1.0000 - val_loss: 4.0512e-04 - ...
   val_accuracy: 1.0000
Epoch 29/50
94/96 [============================>.] - ETA: 0s - loss: 3.0835e-05 - accuracy: 1.0000
Epoch 29: val_accuracy did not improve from 1.00000
```

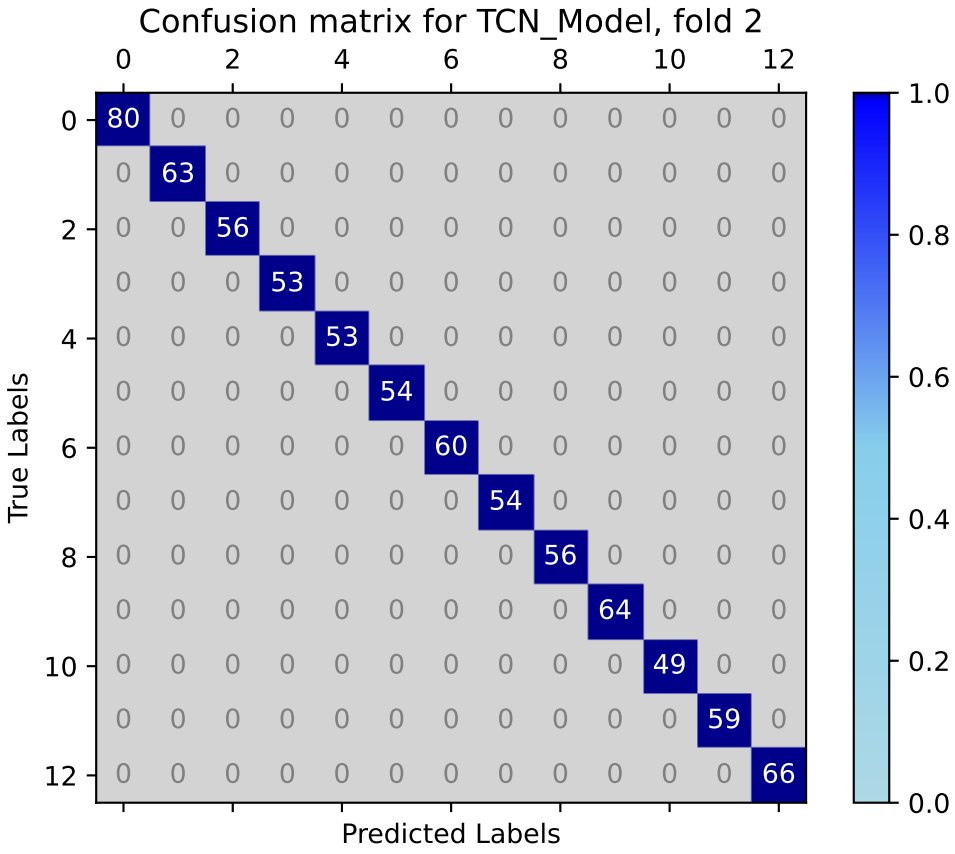
```
96/96 [==============================] - 2s 23ms/step - loss: 3.0670e-05 - accuracy: 1.0000 - val_loss: 3.9050e-04 - ...
    val_accuracy: 1.0000
Epoch 30/50
96/96 [==============================] - ETA: 0s - loss: 2.8566e-05 - accuracy: 1.0000
Epoch 30: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 2.8566e-05 - accuracy: 1.0000 - val_loss: 3.7810e-04 - ...
    val_accuracy: 1.0000
Epoch 31/50
94/96 [============================>.] - ETA: 0s - loss: 2.6879e-05 - accuracy: 1.0000
Epoch 31: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 2.6546e-05 - accuracy: 1.0000 - val_loss: 3.6026e-04 - ...
    val_accuracy: 1.0000
Epoch 32/50
94/96 [============================>.] - ETA: 0s - loss: 2.4845e-05 - accuracy: 1.0000
Epoch 32: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 2.4711e-05 - accuracy: 1.0000 - val_loss: 3.5530e-04 - ...
    val_accuracy: 1.0000
Epoch 33/50
94/96 [============================>.] - ETA: 0s - loss: 2.3184e-05 - accuracy: 1.0000
Epoch 33: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 2.3074e-05 - accuracy: 1.0000 - val_loss: 3.3322e-04 - ...
    val_accuracy: 1.0000
Epoch 34/50
95/96 [============================>.] - ETA: 0s - loss: 2.1495e-05 - accuracy: 1.0000
Epoch 34: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 2.1519e-05 - accuracy: 1.0000 - val_loss: 3.3578e-04 - ...
    val_accuracy: 1.0000
Epoch 35/50
95/96 [============================>.] - ETA: 0s - loss: 2.0192e-05 - accuracy: 1.0000
Epoch 35: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 2.0170e-05 - accuracy: 1.0000 - val_loss: 3.1608e-04 - ...
    val_accuracy: 1.0000
Epoch 36/50
94/96 [============================>.] - ETA: 0s - loss: 1.8913e-05 - accuracy: 1.0000
Epoch 36: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.8831e-05 - accuracy: 1.0000 - val_loss: 3.0321e-04 - ...
    val_accuracy: 1.0000
Epoch 37/50
96/96 [==============================] - ETA: 0s - loss: 1.7655e-05 - accuracy: 1.0000
```

```
Epoch 37: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.7655e-05 - accuracy: 1.0000 - val_loss: 2.9961e-04 - ...
   val_accuracy: 1.0000
Epoch 38/50
94/96 [============================>.] - ETA: 0s - loss: 1.6690e-05 - accuracy: 1.0000
Epoch 38: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.6538e-05 - accuracy: 1.0000 - val_loss: 2.8891e-04 - ...
   val_accuracy: 1.0000
Epoch 39/50
94/96 [============================>.] - ETA: 0s - loss: 1.5570e-05 - accuracy: 1.0000
Epoch 39: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.5581e-05 - accuracy: 1.0000 - val_loss: 2.8768e-04 - ...
   val_accuracy: 1.0000
Epoch 40/50
94/96 [============================>.] - ETA: 0s - loss: 1.4549e-05 - accuracy: 1.0000
Epoch 40: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.4613e-05 - accuracy: 1.0000 - val_loss: 2.6945e-04 - ...
   val_accuracy: 1.0000
Epoch 41/50
94/96 [============================>.] - ETA: 0s - loss: 1.3875e-05 - accuracy: 1.0000
Epoch 41: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.3709e-05 - accuracy: 1.0000 - val_loss: 2.6656e-04 - ...
   val_accuracy: 1.0000
Epoch 42/50
96/96 [==============================] - ETA: 0s - loss: 1.2915e-05 - accuracy: 1.0000
Epoch 42: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.2915e-05 - accuracy: 1.0000 - val_loss: 2.5642e-04 - ...
   val_accuracy: 1.0000
Epoch 43/50
94/96 [============================>.] - ETA: 0s - loss: 1.2103e-05 - accuracy: 1.0000
Epoch 43: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 1.2121e-05 - accuracy: 1.0000 - val_loss: 2.5585e-04 - ...
   val_accuracy: 1.0000
Epoch 44/50
94/96 [============================>.] - ETA: 0s - loss: 1.1354e-05 - accuracy: 1.0000
Epoch 44: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.1528e-05 - accuracy: 1.0000 - val_loss: 2.4512e-04 - ...
   val_accuracy: 1.0000
Epoch 45/50
```

```
94/96 [============================>.] - ETA: 0s - loss: 1.0714e-05 - accuracy: 1.0000
Epoch 45: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.0763e-05 - accuracy: 1.0000 - val_loss: 2.4108e-04 - ...
   val_accuracy: 1.0000
Epoch 46/50
94/96 [============================>.] - ETA: 0s - loss: 1.0040e-05 - accuracy: 1.0000
Epoch 46: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.0095e-05 - accuracy: 1.0000 - val_loss: 2.4169e-04 - ...
   val_accuracy: 1.0000
Epoch 47/50
96/96 [==============================] - ETA: 0s - loss: 9.5057e-06 - accuracy: 1.0000
Epoch 47: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 9.5057e-06 - accuracy: 1.0000 - val_loss: 2.3362e-04 - ...
   val_accuracy: 1.0000
Epoch 48/50
95/96 [============================>.] - ETA: 0s - loss: 8.8765e-06 - accuracy: 1.0000
Epoch 48: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 8.9784e-06 - accuracy: 1.0000 - val_loss: 2.3502e-04 - ...
   val_accuracy: 1.0000
Epoch 49/50
94/96 [============================>.] - ETA: 0s - loss: 8.2490e-06 - accuracy: 1.0000
Epoch 49: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 8.4575e-06 - accuracy: 1.0000 - val_loss: 2.2714e-04 - ...
   val_accuracy: 1.0000
Epoch 50/50
94/96 [============================>.] - ETA: 0s - loss: 8.0136e-06 - accuracy: 1.0000
Epoch 50: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 7.9855e-06 - accuracy: 1.0000 - val_loss: 2.1521e-04 - ...
   val_accuracy: 1.0000
```

```
24/24 [==============================] - 1s 6ms/step - loss: 0.0027 - accuracy: 1.0000
Test Loss TCN_Model, fold 2: 0.0027266021352261305, Test Accuracy TCN_Model, fold 2: 1.0
24/24 [==============================] - 1s 5ms/step
F1 score for TCN_Model, fold 2: 1.0
24/24 [==============================] - 0s 5ms/step
```


Training History for TCN_Model, fold 2
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
0.00
0.01
0.02
0.03
0.04
0.986
0.988
0.990
0.992
0.994
0.996
0.998
1.000
0
10
20
30
40
50


Confusion matrix for TCN_Model, fold 2
True Labels
Predicted Labels
0
2
4
6
8
10
12
1.0
0.8
0.6
0.4
0.2
0.0

```
Epoch 1/50
94/96 [============================>.] - ETA: 0s - loss: 0.0364 - accuracy: 0.9914
Epoch 1: val_accuracy improved from -inf to 0.99218, saving model to /kaggle/working/TCN_Model_Fold_3_Checkpoint.h5


/opt/conda/lib/python3.10/site-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an ...
    HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras ...
    format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(

96/96 [==============================] - 21s 34ms/step - loss: 0.0378 - accuracy: 0.9909 - val_loss: 0.0163 - ...
    val_accuracy: 0.9922
Epoch 2/50
94/96 [============================>.] - ETA: 0s - loss: 0.0278 - accuracy: 0.9914
Epoch 2: val_accuracy did not improve from 0.99218
96/96 [==============================] - 2s 21ms/step - loss: 0.0273 - accuracy: 0.9915 - val_loss: 0.0154 - ...
    val_accuracy: 0.9909
Epoch 3/50
94/96 [============================>.] - ETA: 0s - loss: 0.0528 - accuracy: 0.9827
Epoch 3: val_accuracy improved from 0.99218 to 0.99348, saving model to /kaggle/working/TCN_Model_Fold_3_Checkpoint.h5
96/96 [==============================] - 2s 24ms/step - loss: 0.0522 - accuracy: 0.9827 - val_loss: 0.0201 - ...
    val_accuracy: 0.9935
Epoch 4/50
96/96 [==============================] - ETA: 0s - loss: 0.0177 - accuracy: 0.9941
Epoch 4: val_accuracy did not improve from 0.99348
96/96 [==============================] - 2s 22ms/step - loss: 0.0177 - accuracy: 0.9941 - val_loss: 0.0282 - ...
    val_accuracy: 0.9909
Epoch 5/50
94/96 [============================>.] - ETA: 0s - loss: 0.0314 - accuracy: 0.9920
Epoch 5: val_accuracy improved from 0.99348 to 0.99870, saving model to /kaggle/working/TCN_Model_Fold_3_Checkpoint.h5
96/96 [==============================] - 2s 25ms/step - loss: 0.0309 - accuracy: 0.9922 - val_loss: 0.0092 - ...
    val_accuracy: 0.9987
Epoch 6/50
94/96 [============================>.] - ETA: 0s - loss: 0.0033 - accuracy: 0.9997
Epoch 6: val_accuracy did not improve from 0.99870
96/96 [==============================] - 2s 22ms/step - loss: 0.0033 - accuracy: 0.9997 - val_loss: 0.0024 - ...
    val_accuracy: 0.9987
Epoch 7/50
```

```
94/96 [============================>.] - ETA: 0s - loss: 0.0012 - accuracy: 1.0000
Epoch 7: val_accuracy improved from 0.99870 to 1.00000, saving model to /kaggle/working/TCN_Model_Fold_3_Checkpoint.h5
96/96 [==============================] - 2s 26ms/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.0020 - ...
   val_accuracy: 1.0000
Epoch 8/50
94/96 [============================>.] - ETA: 0s - loss: 6.3927e-04 - accuracy: 1.0000
Epoch 8: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 24ms/step - loss: 6.3312e-04 - accuracy: 1.0000 - val_loss: 0.0013 - ...
   val_accuracy: 1.0000
Epoch 9/50
95/96 [============================>.] - ETA: 0s - loss: 4.4152e-04 - accuracy: 1.0000
Epoch 9: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 4.4014e-04 - accuracy: 1.0000 - val_loss: 0.0010 - ...
   val_accuracy: 1.0000
Epoch 10/50
94/96 [============================>.] - ETA: 0s - loss: 3.5216e-04 - accuracy: 1.0000
Epoch 10: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 3.4977e-04 - accuracy: 1.0000 - val_loss: 9.1120e-04 - ...
   val_accuracy: 1.0000
Epoch 11/50
94/96 [============================>.] - ETA: 0s - loss: 2.9426e-04 - accuracy: 1.0000
Epoch 11: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 2.9149e-04 - accuracy: 1.0000 - val_loss: 7.8714e-04 - ...
   val_accuracy: 1.0000
Epoch 12/50
94/96 [============================>.] - ETA: 0s - loss: 2.4891e-04 - accuracy: 1.0000
Epoch 12: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 2.4817e-04 - accuracy: 1.0000 - val_loss: 7.2418e-04 - ...
   val_accuracy: 1.0000
Epoch 13/50
96/96 [==============================] - ETA: 0s - loss: 2.1542e-04 - accuracy: 1.0000
Epoch 13: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 2.1542e-04 - accuracy: 1.0000 - val_loss: 7.0652e-04 - ...
   val_accuracy: 1.0000
Epoch 14/50
94/96 [============================>.] - ETA: 0s - loss: 1.8874e-04 - accuracy: 1.0000
Epoch 14: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.8629e-04 - accuracy: 1.0000 - val_loss: 6.0809e-04 - ...
   val_accuracy: 1.0000
```

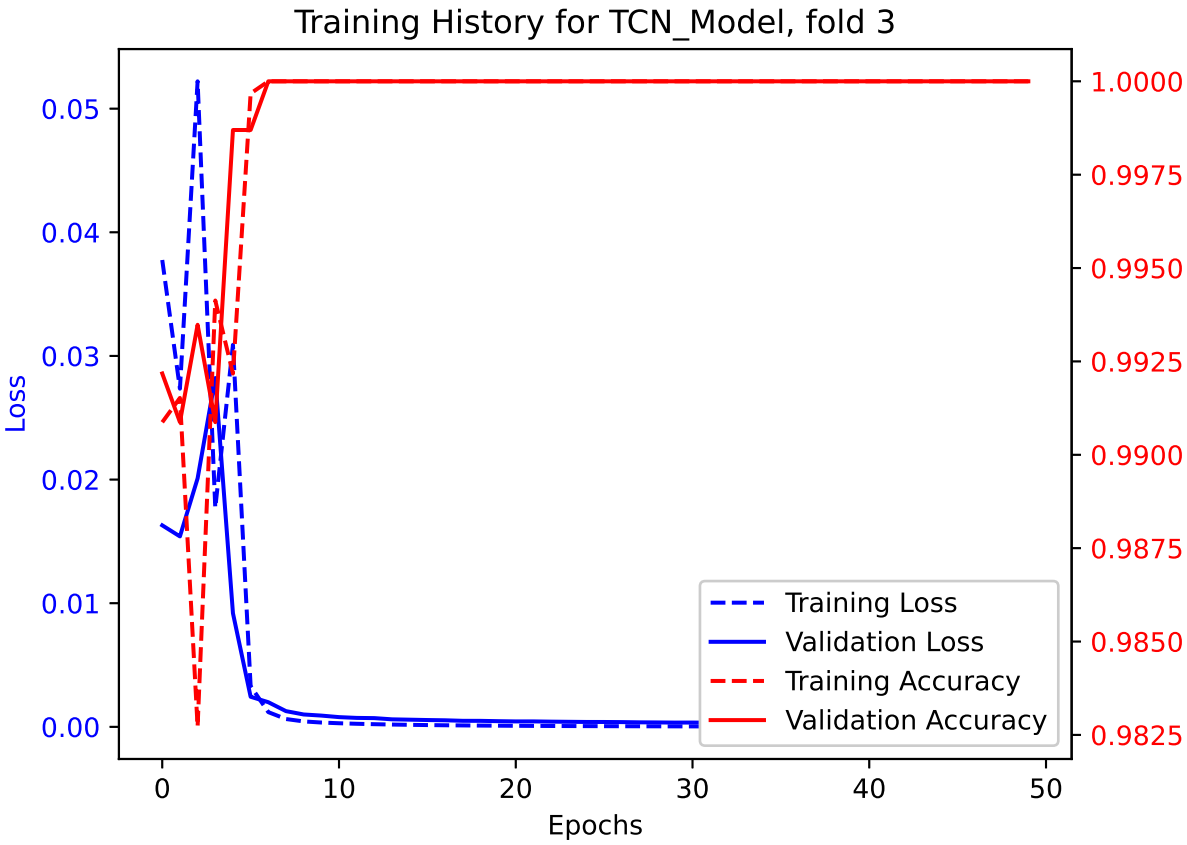
```
Epoch 15/50
94/96 [============================>.] - ETA: 0s - loss: 1.5976e-04 - accuracy: 1.0000
Epoch 15: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.6357e-04 - accuracy: 1.0000 - val_loss: 5.8036e-04 - ...
   val_accuracy: 1.0000
Epoch 16/50
94/96 [============================>.] - ETA: 0s - loss: 1.4545e-04 - accuracy: 1.0000
Epoch 16: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.4436e-04 - accuracy: 1.0000 - val_loss: 5.5169e-04 - ...
   val_accuracy: 1.0000
Epoch 17/50
94/96 [============================>.] - ETA: 0s - loss: 1.2829e-04 - accuracy: 1.0000
Epoch 17: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.2827e-04 - accuracy: 1.0000 - val_loss: 5.3097e-04 - ...
   val_accuracy: 1.0000
Epoch 18/50
96/96 [==============================] - ETA: 0s - loss: 1.1557e-04 - accuracy: 1.0000
Epoch 18: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 1.1557e-04 - accuracy: 1.0000 - val_loss: 4.8855e-04 - ...
   val_accuracy: 1.0000
Epoch 19/50
94/96 [============================>.] - ETA: 0s - loss: 1.0283e-04 - accuracy: 1.0000
Epoch 19: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.0342e-04 - accuracy: 1.0000 - val_loss: 4.8586e-04 - ...
   val_accuracy: 1.0000
Epoch 20/50
94/96 [============================>.] - ETA: 0s - loss: 9.4054e-05 - accuracy: 1.0000
Epoch 20: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 9.4048e-05 - accuracy: 1.0000 - val_loss: 4.5994e-04 - ...
   val_accuracy: 1.0000
Epoch 21/50
94/96 [============================>.] - ETA: 0s - loss: 8.5357e-05 - accuracy: 1.0000
Epoch 21: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 8.4924e-05 - accuracy: 1.0000 - val_loss: 4.4007e-04 - ...
   val_accuracy: 1.0000
Epoch 22/50
94/96 [============================>.] - ETA: 0s - loss: 7.8173e-05 - accuracy: 1.0000
Epoch 22: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 7.6885e-05 - accuracy: 1.0000 - val_loss: 4.4150e-04 - ...
```

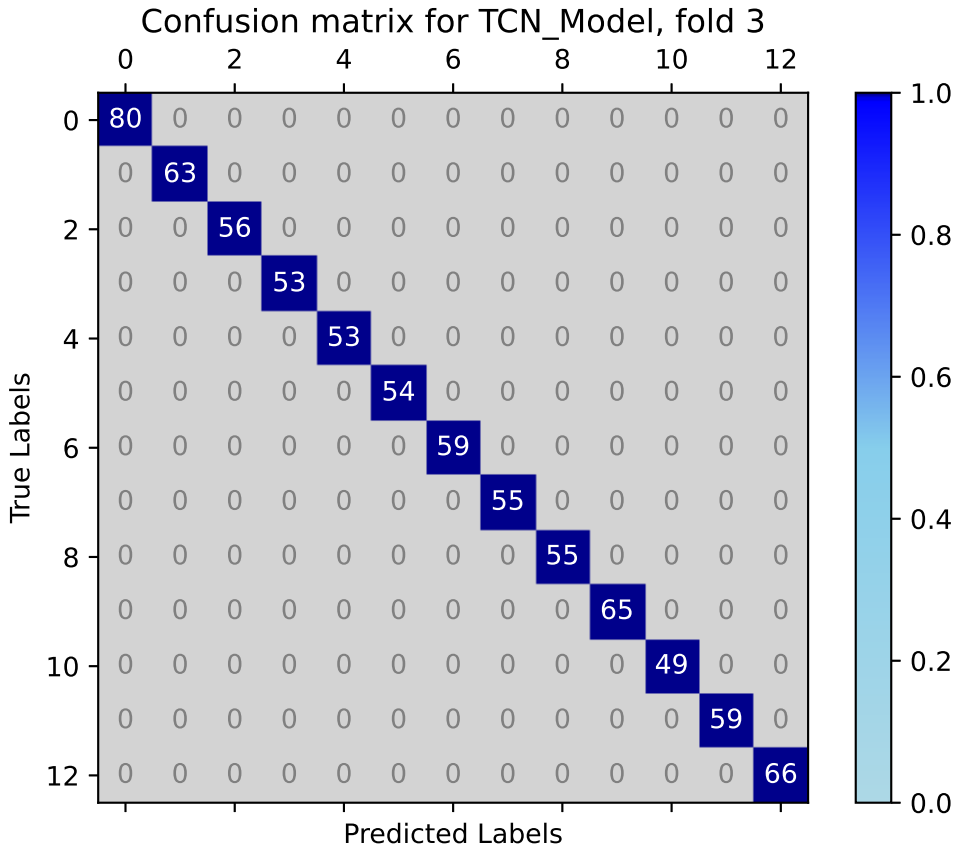
```
   val_accuracy: 1.0000
Epoch 23/50
96/96 [==============================] - ETA: 0s - loss: 7.0188e-05 - accuracy: 1.0000
Epoch 23: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 7.0188e-05 - accuracy: 1.0000 - val_loss: 4.2152e-04 - ...
   val_accuracy: 1.0000
Epoch 24/50
96/96 [==============================] - ETA: 0s - loss: 6.4382e-05 - accuracy: 1.0000
Epoch 24: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 6.4382e-05 - accuracy: 1.0000 - val_loss: 4.0853e-04 - ...
   val_accuracy: 1.0000
Epoch 25/50
94/96 [============================>.] - ETA: 0s - loss: 5.8491e-05 - accuracy: 1.0000
Epoch 25: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 5.9006e-05 - accuracy: 1.0000 - val_loss: 3.9552e-04 - ...
   val_accuracy: 1.0000
Epoch 26/50
94/96 [============================>.] - ETA: 0s - loss: 5.5246e-05 - accuracy: 1.0000
Epoch 26: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 5.4677e-05 - accuracy: 1.0000 - val_loss: 3.9376e-04 - ...
   val_accuracy: 1.0000
Epoch 27/50
94/96 [============================>.] - ETA: 0s - loss: 5.1279e-05 - accuracy: 1.0000
Epoch 27: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 5.0647e-05 - accuracy: 1.0000 - val_loss: 3.8424e-04 - ...
   val_accuracy: 1.0000
Epoch 28/50
94/96 [============================>.] - ETA: 0s - loss: 4.6153e-05 - accuracy: 1.0000
Epoch 28: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 4.6381e-05 - accuracy: 1.0000 - val_loss: 3.6566e-04 - ...
   val_accuracy: 1.0000
Epoch 29/50
94/96 [============================>.] - ETA: 0s - loss: 4.3372e-05 - accuracy: 1.0000
Epoch 29: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 4.3119e-05 - accuracy: 1.0000 - val_loss: 3.5742e-04 - ...
   val_accuracy: 1.0000
Epoch 30/50
94/96 [============================>.] - ETA: 0s - loss: 4.0172e-05 - accuracy: 1.0000
Epoch 30: val_accuracy did not improve from 1.00000
```

```
96/96 [==============================] - 2s 21ms/step - loss: 3.9998e-05 - accuracy: 1.0000 - val_loss: 3.4522e-04 - ...
    val_accuracy: 1.0000
Epoch 31/50
94/96 [============================>.] - ETA: 0s - loss: 3.6719e-05 - accuracy: 1.0000
Epoch 31: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 3.7144e-05 - accuracy: 1.0000 - val_loss: 3.4817e-04 - ...
    val_accuracy: 1.0000
Epoch 32/50
96/96 [==============================] - ETA: 0s - loss: 3.4578e-05 - accuracy: 1.0000
Epoch 32: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 3.4578e-05 - accuracy: 1.0000 - val_loss: 3.3524e-04 - ...
    val_accuracy: 1.0000
Epoch 33/50
94/96 [============================>.] - ETA: 0s - loss: 3.2547e-05 - accuracy: 1.0000
Epoch 33: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 3.2205e-05 - accuracy: 1.0000 - val_loss: 3.2733e-04 - ...
    val_accuracy: 1.0000
Epoch 34/50
94/96 [============================>.] - ETA: 0s - loss: 3.0594e-05 - accuracy: 1.0000
Epoch 34: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 3.0221e-05 - accuracy: 1.0000 - val_loss: 3.1504e-04 - ...
    val_accuracy: 1.0000
Epoch 35/50
94/96 [============================>.] - ETA: 0s - loss: 2.7157e-05 - accuracy: 1.0000
Epoch 35: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 2.8098e-05 - accuracy: 1.0000 - val_loss: 3.0965e-04 - ...
    val_accuracy: 1.0000
Epoch 36/50
95/96 [============================>.] - ETA: 0s - loss: 2.6418e-05 - accuracy: 1.0000
Epoch 36: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 2.6216e-05 - accuracy: 1.0000 - val_loss: 2.9671e-04 - ...
    val_accuracy: 1.0000
Epoch 37/50
94/96 [============================>.] - ETA: 0s - loss: 2.4447e-05 - accuracy: 1.0000
Epoch 37: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 2.4465e-05 - accuracy: 1.0000 - val_loss: 3.1127e-04 - ...
    val_accuracy: 1.0000
Epoch 38/50
94/96 [============================>.] - ETA: 0s - loss: 2.3003e-05 - accuracy: 1.0000
```

```
Epoch 38: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 2.2902e-05 - accuracy: 1.0000 - val_loss: 3.0858e-04 - ...
   val_accuracy: 1.0000
Epoch 39/50
94/96 [============================>.] - ETA: 0s - loss: 2.1371e-05 - accuracy: 1.0000
Epoch 39: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 2.1540e-05 - accuracy: 1.0000 - val_loss: 2.9455e-04 - ...
   val_accuracy: 1.0000
Epoch 40/50
94/96 [============================>.] - ETA: 0s - loss: 2.0270e-05 - accuracy: 1.0000
Epoch 40: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 2.0322e-05 - accuracy: 1.0000 - val_loss: 2.9765e-04 - ...
   val_accuracy: 1.0000
Epoch 41/50
94/96 [============================>.] - ETA: 0s - loss: 1.9011e-05 - accuracy: 1.0000
Epoch 41: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.8888e-05 - accuracy: 1.0000 - val_loss: 2.8050e-04 - ...
   val_accuracy: 1.0000
Epoch 42/50
96/96 [==============================] - ETA: 0s - loss: 1.7774e-05 - accuracy: 1.0000
Epoch 42: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 1.7774e-05 - accuracy: 1.0000 - val_loss: 2.7031e-04 - ...
   val_accuracy: 1.0000
Epoch 43/50
94/96 [============================>.] - ETA: 0s - loss: 1.6804e-05 - accuracy: 1.0000
Epoch 43: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.6659e-05 - accuracy: 1.0000 - val_loss: 2.8150e-04 - ...
   val_accuracy: 1.0000
Epoch 44/50
94/96 [============================>.] - ETA: 0s - loss: 1.5712e-05 - accuracy: 1.0000
Epoch 44: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.5660e-05 - accuracy: 1.0000 - val_loss: 2.7275e-04 - ...
   val_accuracy: 1.0000
Epoch 45/50
94/96 [============================>.] - ETA: 0s - loss: 1.4922e-05 - accuracy: 1.0000
Epoch 45: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.4706e-05 - accuracy: 1.0000 - val_loss: 2.7587e-04 - ...
   val_accuracy: 1.0000
Epoch 46/50
```

```
94/96 [============================>.] - ETA: 0s - loss: 1.3910e-05 - accuracy: 1.0000
Epoch 46: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.3836e-05 - accuracy: 1.0000 - val_loss: 2.6173e-04 - ...
   val_accuracy: 1.0000
Epoch 47/50
96/96 [==============================] - ETA: 0s - loss: 1.3051e-05 - accuracy: 1.0000
Epoch 47: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.3051e-05 - accuracy: 1.0000 - val_loss: 2.6550e-04 - ...
   val_accuracy: 1.0000
Epoch 48/50
94/96 [============================>.] - ETA: 0s - loss: 1.2394e-05 - accuracy: 1.0000
Epoch 48: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.2302e-05 - accuracy: 1.0000 - val_loss: 2.6586e-04 - ...
   val_accuracy: 1.0000
Epoch 49/50
94/96 [============================>.] - ETA: 0s - loss: 1.1639e-05 - accuracy: 1.0000
Epoch 49: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 1.1563e-05 - accuracy: 1.0000 - val_loss: 2.5378e-04 - ...
   val_accuracy: 1.0000
Epoch 50/50
94/96 [============================>.] - ETA: 0s - loss: 1.0398e-05 - accuracy: 1.0000
Epoch 50: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.0845e-05 - accuracy: 1.0000 - val_loss: 2.6097e-04 - ...
   val_accuracy: 1.0000
```


Training History for TCN_Model, fold 3
Loss
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
0.05
0.04
0.03
0.02
0.01
0.00
1.0000
0.9975
0.9950
0.9925
0.9900
0.9875
0.9850
0.9825
0
10
20
30
40
50


Confusion matrix for TCN_Model, fold 3
True Labels
Predicted Labels
0
2
4
6
8
10
12
1.0
0.8
0.6
0.4
0.2
0.0
80
63
56
53
53
54
59
55
55
65
49
59
66

```
Epoch 1/50
96/96 [==============================] - ETA: 0s - loss: 0.0313 - accuracy: 0.9892
Epoch 1: val_accuracy improved from -inf to 0.99869, saving model to /kaggle/working/TCN_Model_Fold_4_Checkpoint.h5
96/96 [==============================] - 20s 38ms/step - loss: 0.0313 - accuracy: 0.9892 - val_loss: 0.0060 - ...
    val_accuracy: 0.9987
Epoch 2/50

/opt/conda/lib/python3.10/site-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an ...
    HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras ...
    format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(

94/96 [============================>.] - ETA: 0s - loss: 0.0412 - accuracy: 0.9877
Epoch 2: val_accuracy did not improve from 0.99869
96/96 [==============================] - 2s 22ms/step - loss: 0.0404 - accuracy: 0.9879 - val_loss: 0.0203 - ...
    val_accuracy: 0.9935
Epoch 3/50
94/96 [============================>.] - ETA: 0s - loss: 0.0137 - accuracy: 0.9960
Epoch 3: val_accuracy did not improve from 0.99869
96/96 [==============================] - 2s 21ms/step - loss: 0.0135 - accuracy: 0.9961 - val_loss: 0.0329 - ...
    val_accuracy: 0.9909
Epoch 4/50
95/96 [============================>.] - ETA: 0s - loss: 0.0055 - accuracy: 0.9987
Epoch 4: val_accuracy improved from 0.99869 to 1.00000, saving model to /kaggle/working/TCN_Model_Fold_4_Checkpoint.h5
96/96 [==============================] - 2s 25ms/step - loss: 0.0054 - accuracy: 0.9987 - val_loss: 0.0016 - ...
    val_accuracy: 1.0000
Epoch 5/50
95/96 [============================>.] - ETA: 0s - loss: 7.6712e-04 - accuracy: 1.0000
Epoch 5: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 7.6091e-04 - accuracy: 1.0000 - val_loss: 7.6142e-04 - ...
    val_accuracy: 1.0000
Epoch 6/50
94/96 [============================>.] - ETA: 0s - loss: 3.5520e-04 - accuracy: 1.0000
Epoch 6: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 3.6065e-04 - accuracy: 1.0000 - val_loss: 6.3451e-04 - ...
    val_accuracy: 1.0000
Epoch 7/50
94/96 [============================>.] - ETA: 0s - loss: 2.7985e-04 - accuracy: 1.0000
```

```
Epoch 7: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 2.8092e-04 - accuracy: 1.0000 - val_loss: 5.6099e-04 - ...
   val_accuracy: 1.0000
Epoch 8/50
94/96 [============================>.] - ETA: 0s - loss: 2.3220e-04 - accuracy: 1.0000
Epoch 8: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 2.3624e-04 - accuracy: 1.0000 - val_loss: 5.1023e-04 - ...
   val_accuracy: 1.0000
Epoch 9/50
95/96 [============================>.] - ETA: 0s - loss: 2.0174e-04 - accuracy: 1.0000
Epoch 9: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 2.0061e-04 - accuracy: 1.0000 - val_loss: 4.6077e-04 - ...
   val_accuracy: 1.0000
Epoch 10/50
94/96 [============================>.] - ETA: 0s - loss: 1.7378e-04 - accuracy: 1.0000
Epoch 10: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 1.7172e-04 - accuracy: 1.0000 - val_loss: 4.1827e-04 - ...
   val_accuracy: 1.0000
Epoch 11/50
94/96 [============================>.] - ETA: 0s - loss: 1.4862e-04 - accuracy: 1.0000
Epoch 11: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.5059e-04 - accuracy: 1.0000 - val_loss: 4.0267e-04 - ...
   val_accuracy: 1.0000
Epoch 12/50
94/96 [============================>.] - ETA: 0s - loss: 1.3495e-04 - accuracy: 1.0000
Epoch 12: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.3270e-04 - accuracy: 1.0000 - val_loss: 3.6745e-04 - ...
   val_accuracy: 1.0000
Epoch 13/50
94/96 [============================>.] - ETA: 0s - loss: 1.1751e-04 - accuracy: 1.0000
Epoch 13: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.1783e-04 - accuracy: 1.0000 - val_loss: 3.4786e-04 - ...
   val_accuracy: 1.0000
Epoch 14/50
96/96 [==============================] - ETA: 0s - loss: 1.0506e-04 - accuracy: 1.0000
Epoch 14: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.0506e-04 - accuracy: 1.0000 - val_loss: 3.2219e-04 - ...
   val_accuracy: 1.0000
Epoch 15/50
```

```
94/96 [============================>.] - ETA: 0s - loss: 9.5430e-05 - accuracy: 1.0000
Epoch 15: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 9.4257e-05 - accuracy: 1.0000 - val_loss: 2.9993e-04 - ...
   val_accuracy: 1.0000
Epoch 16/50
94/96 [============================>.] - ETA: 0s - loss: 8.4319e-05 - accuracy: 1.0000
Epoch 16: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 8.5236e-05 - accuracy: 1.0000 - val_loss: 2.8524e-04 - ...
   val_accuracy: 1.0000
Epoch 17/50
94/96 [============================>.] - ETA: 0s - loss: 7.6862e-05 - accuracy: 1.0000
Epoch 17: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 7.7006e-05 - accuracy: 1.0000 - val_loss: 2.6801e-04 - ...
   val_accuracy: 1.0000
Epoch 18/50
94/96 [============================>.] - ETA: 0s - loss: 7.0063e-05 - accuracy: 1.0000
Epoch 18: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 7.0155e-05 - accuracy: 1.0000 - val_loss: 2.5507e-04 - ...
   val_accuracy: 1.0000
Epoch 19/50
96/96 [==============================] - ETA: 0s - loss: 6.3682e-05 - accuracy: 1.0000
Epoch 19: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 24ms/step - loss: 6.3682e-05 - accuracy: 1.0000 - val_loss: 2.4237e-04 - ...
   val_accuracy: 1.0000
Epoch 20/50
94/96 [============================>.] - ETA: 0s - loss: 5.7283e-05 - accuracy: 1.0000
Epoch 20: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 5.8458e-05 - accuracy: 1.0000 - val_loss: 2.2899e-04 - ...
   val_accuracy: 1.0000
Epoch 21/50
94/96 [============================>.] - ETA: 0s - loss: 5.3490e-05 - accuracy: 1.0000
Epoch 21: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 5.3664e-05 - accuracy: 1.0000 - val_loss: 2.1283e-04 - ...
   val_accuracy: 1.0000
Epoch 22/50
94/96 [============================>.] - ETA: 0s - loss: 4.9152e-05 - accuracy: 1.0000
Epoch 22: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 4.9387e-05 - accuracy: 1.0000 - val_loss: 2.0564e-04 - ...
   val_accuracy: 1.0000
```

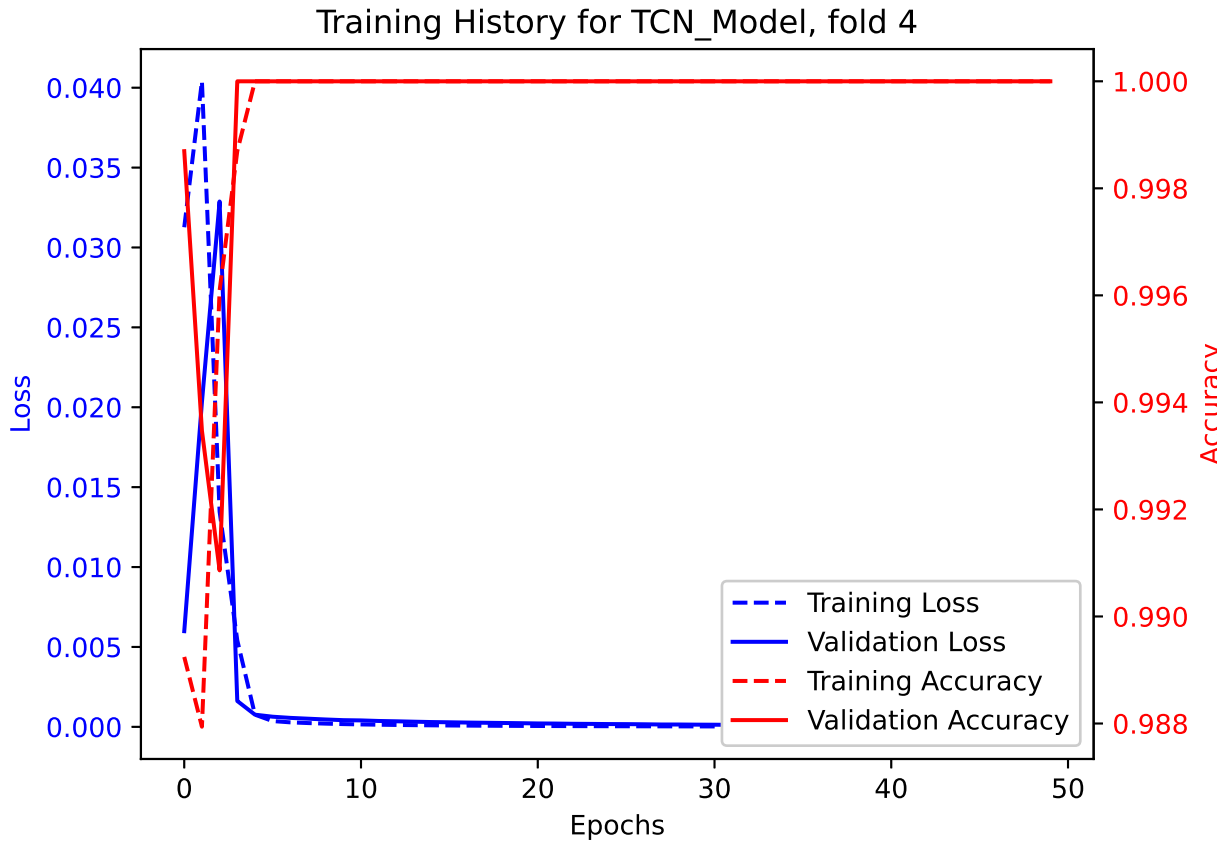
```
Epoch 23/50
94/96 [============================>.] - ETA: 0s - loss: 4.4365e-05 - accuracy: 1.0000
Epoch 23: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 4.5328e-05 - accuracy: 1.0000 - val_loss: 1.9159e-04 - ...
   val_accuracy: 1.0000
Epoch 24/50
96/96 [==============================] - ETA: 0s - loss: 4.1938e-05 - accuracy: 1.0000
Epoch 24: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 4.1938e-05 - accuracy: 1.0000 - val_loss: 1.8405e-04 - ...
   val_accuracy: 1.0000
Epoch 25/50
94/96 [============================>.] - ETA: 0s - loss: 3.8818e-05 - accuracy: 1.0000
Epoch 25: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 3.8688e-05 - accuracy: 1.0000 - val_loss: 1.7415e-04 - ...
   val_accuracy: 1.0000
Epoch 26/50
94/96 [============================>.] - ETA: 0s - loss: 3.6126e-05 - accuracy: 1.0000
Epoch 26: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 3.5876e-05 - accuracy: 1.0000 - val_loss: 1.6975e-04 - ...
   val_accuracy: 1.0000
Epoch 27/50
94/96 [============================>.] - ETA: 0s - loss: 3.3657e-05 - accuracy: 1.0000
Epoch 27: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 3.3320e-05 - accuracy: 1.0000 - val_loss: 1.6037e-04 - ...
   val_accuracy: 1.0000
Epoch 28/50
94/96 [============================>.] - ETA: 0s - loss: 3.0381e-05 - accuracy: 1.0000
Epoch 28: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 3.1048e-05 - accuracy: 1.0000 - val_loss: 1.4685e-04 - ...
   val_accuracy: 1.0000
Epoch 29/50
96/96 [==============================] - ETA: 0s - loss: 2.8672e-05 - accuracy: 1.0000
Epoch 29: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 2.8672e-05 - accuracy: 1.0000 - val_loss: 1.4570e-04 - ...
   val_accuracy: 1.0000
Epoch 30/50
94/96 [============================>.] - ETA: 0s - loss: 2.7072e-05 - accuracy: 1.0000
Epoch 30: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 2.6734e-05 - accuracy: 1.0000 - val_loss: 1.3735e-04 - ...
```

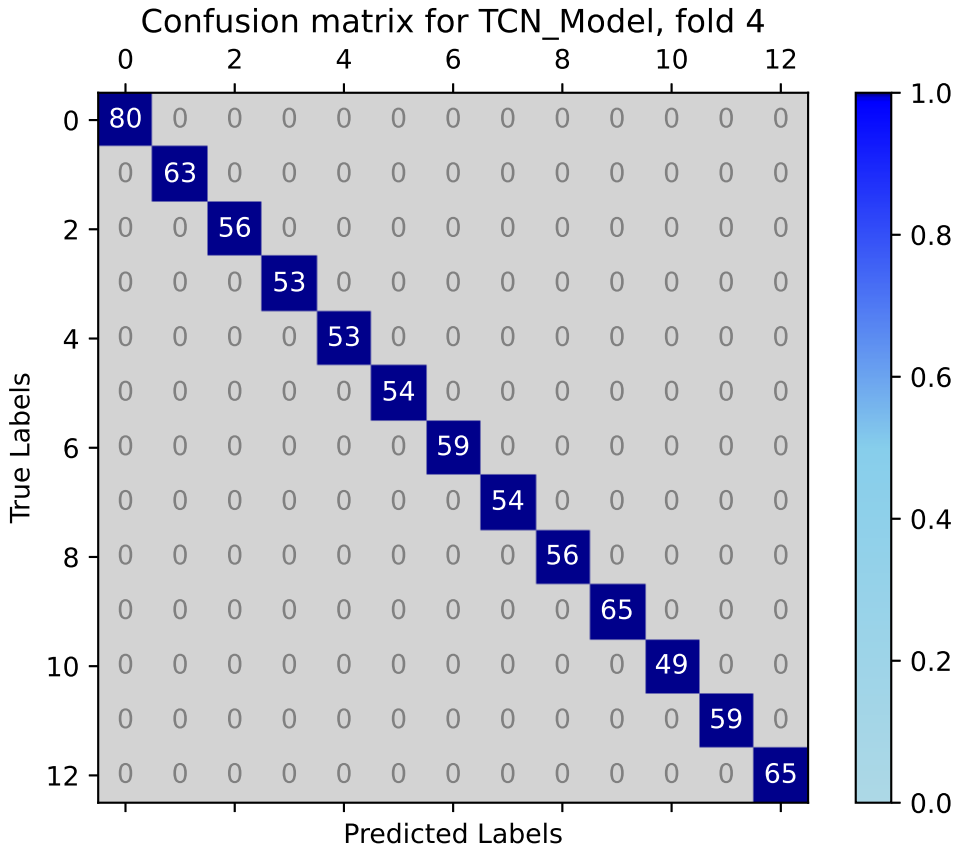
```
   val_accuracy: 1.0000
Epoch 31/50
94/96 [============================>.] - ETA: 0s - loss: 2.5077e-05 - accuracy: 1.0000
Epoch 31: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 2.4935e-05 - accuracy: 1.0000 - val_loss: 1.3303e-04 - ...
   val_accuracy: 1.0000
Epoch 32/50
94/96 [============================>.] - ETA: 0s - loss: 2.3225e-05 - accuracy: 1.0000
Epoch 32: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 2.3234e-05 - accuracy: 1.0000 - val_loss: 1.2317e-04 - ...
   val_accuracy: 1.0000
Epoch 33/50
96/96 [==============================] - ETA: 0s - loss: 2.1735e-05 - accuracy: 1.0000
Epoch 33: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 2.1735e-05 - accuracy: 1.0000 - val_loss: 1.1746e-04 - ...
   val_accuracy: 1.0000
Epoch 34/50
94/96 [============================>.] - ETA: 0s - loss: 2.0343e-05 - accuracy: 1.0000
Epoch 34: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 2.0352e-05 - accuracy: 1.0000 - val_loss: 1.1440e-04 - ...
   val_accuracy: 1.0000
Epoch 35/50
94/96 [============================>.] - ETA: 0s - loss: 1.8887e-05 - accuracy: 1.0000
Epoch 35: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 1.9046e-05 - accuracy: 1.0000 - val_loss: 1.0561e-04 - ...
   val_accuracy: 1.0000
Epoch 36/50
94/96 [============================>.] - ETA: 0s - loss: 1.7750e-05 - accuracy: 1.0000
Epoch 36: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.7879e-05 - accuracy: 1.0000 - val_loss: 1.0484e-04 - ...
   val_accuracy: 1.0000
Epoch 37/50
94/96 [============================>.] - ETA: 0s - loss: 1.6902e-05 - accuracy: 1.0000
Epoch 37: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.6743e-05 - accuracy: 1.0000 - val_loss: 9.5555e-05 - ...
   val_accuracy: 1.0000
Epoch 38/50
96/96 [==============================] - ETA: 0s - loss: 1.5708e-05 - accuracy: 1.0000
Epoch 38: val_accuracy did not improve from 1.00000
```

```
96/96 [==============================] - 2s 21ms/step - loss: 1.5708e-05 - accuracy: 1.0000 - val_loss: 9.3722e-05 - ...
    val_accuracy: 1.0000
Epoch 39/50
94/96 [============================>.] - ETA: 0s - loss: 1.4922e-05 - accuracy: 1.0000
Epoch 39: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.4726e-05 - accuracy: 1.0000 - val_loss: 8.9993e-05 - ...
    val_accuracy: 1.0000
Epoch 40/50
94/96 [============================>.] - ETA: 0s - loss: 1.3873e-05 - accuracy: 1.0000
Epoch 40: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.3824e-05 - accuracy: 1.0000 - val_loss: 8.8320e-05 - ...
    val_accuracy: 1.0000
Epoch 41/50
94/96 [============================>.] - ETA: 0s - loss: 1.3130e-05 - accuracy: 1.0000
Epoch 41: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.3043e-05 - accuracy: 1.0000 - val_loss: 8.4535e-05 - ...
    val_accuracy: 1.0000
Epoch 42/50
94/96 [============================>.] - ETA: 0s - loss: 1.2297e-05 - accuracy: 1.0000
Epoch 42: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.2238e-05 - accuracy: 1.0000 - val_loss: 8.0065e-05 - ...
    val_accuracy: 1.0000
Epoch 43/50
96/96 [==============================] - ETA: 0s - loss: 1.1537e-05 - accuracy: 1.0000
Epoch 43: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.1537e-05 - accuracy: 1.0000 - val_loss: 7.6455e-05 - ...
    val_accuracy: 1.0000
Epoch 44/50
94/96 [============================>.] - ETA: 0s - loss: 1.0991e-05 - accuracy: 1.0000
Epoch 44: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.0832e-05 - accuracy: 1.0000 - val_loss: 7.3137e-05 - ...
    val_accuracy: 1.0000
Epoch 45/50
94/96 [============================>.] - ETA: 0s - loss: 1.0044e-05 - accuracy: 1.0000
Epoch 45: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 1.0230e-05 - accuracy: 1.0000 - val_loss: 7.2747e-05 - ...
    val_accuracy: 1.0000
Epoch 46/50
94/96 [============================>.] - ETA: 0s - loss: 9.5610e-06 - accuracy: 1.0000
```

```
Epoch 46: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 22ms/step - loss: 9.6569e-06 - accuracy: 1.0000 - val_loss: 6.9268e-05 - ...
    val_accuracy: 1.0000
Epoch 47/50
94/96 [============================>.] - ETA: 0s - loss: 9.2419e-06 - accuracy: 1.0000
Epoch 47: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 9.0885e-06 - accuracy: 1.0000 - val_loss: 6.8214e-05 - ...
    val_accuracy: 1.0000
Epoch 48/50
96/96 [==============================] - ETA: 0s - loss: 8.5844e-06 - accuracy: 1.0000
Epoch 48: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 8.5844e-06 - accuracy: 1.0000 - val_loss: 6.4375e-05 - ...
    val_accuracy: 1.0000
Epoch 49/50
94/96 [============================>.] - ETA: 0s - loss: 8.0817e-06 - accuracy: 1.0000
Epoch 49: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 21ms/step - loss: 8.0933e-06 - accuracy: 1.0000 - val_loss: 6.2279e-05 - ...
    val_accuracy: 1.0000
Epoch 50/50
94/96 [============================>.] - ETA: 0s - loss: 7.7657e-06 - accuracy: 1.0000
Epoch 50: val_accuracy did not improve from 1.00000
96/96 [==============================] - 2s 23ms/step - loss: 7.6699e-06 - accuracy: 1.0000 - val_loss: 6.1317e-05 - ...
    val_accuracy: 1.0000
```

```
24/24 [==============================] - 1s 6ms/step - loss: 0.0016 - accuracy: 1.0000
Test Loss TCN_Model, fold 4: 0.0016189317684620619, Test Accuracy TCN_Model, fold 4: 1.0
24/24 [==============================] - 1s 5ms/step
F1 score for TCN_Model, fold 4: 1.0
24/24 [==============================] - 0s 5ms/step
```


Training History for TCN_Model, fold 4
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
1.000
0.998
0.996
0.994
0.992
0.990
0.988
0
10
20
30
40
50


Confusion matrix for TCN_Model, fold 4
True Labels
Predicted Labels
0
2
4
6
8
10
12
1.0
0.8
0.6
0.4
0.2
0.0
80
63
56
53
53
54
59
54
56
65
49
59
65

RESULTS for TCN Model

```
RESULTS for TCN_Model
Average F1 Score for TCN_Model: 1.0
F1 scores for all folds for TCN_Model: [1.0, 1.0, 1.0, 1.0, 1.0]
```

CNN model

```
Epoch 1/50
96/96 [==============================] - ETA: 0s - loss: 0.5738 - accuracy: 0.9423
Epoch 1: val_accuracy improved from -inf to 0.97914, saving model to /kaggle/working/CNN_Model_Fold_0_Checkpoint.h5
96/96 [==============================] - 7s 17ms/step - loss: 0.5738 - accuracy: 0.9423 - val_loss: 0.4400 - ...
   val_accuracy: 0.9791
Epoch 2/50
 7/96 [=>............................] - ETA: 0s - loss: 0.5001 - accuracy: 0.9598


91/96 [===========================>..] - ETA: 0s - loss: 0.5024 - accuracy: 0.9560
Epoch 2: val_accuracy improved from 0.97914 to 0.98696, saving model to /kaggle/working/CNN_Model_Fold_0_Checkpoint.h5
96/96 [==============================] - 1s 10ms/step - loss: 0.4986 - accuracy: 0.9570 - val_loss: 0.4020 - ...
   val_accuracy: 0.9870
Epoch 3/50
96/96 [==============================] - ETA: 0s - loss: 0.4743 - accuracy: 0.9527
Epoch 3: val_accuracy did not improve from 0.98696
96/96 [==============================] - 1s 10ms/step - loss: 0.4743 - accuracy: 0.9527 - val_loss: 0.3941 - ...
   val_accuracy: 0.9831
Epoch 4/50
95/96 [============================>.] - ETA: 0s - loss: 0.4577 - accuracy: 0.9556
Epoch 4: val_accuracy did not improve from 0.98696
96/96 [==============================] - 1s 10ms/step - loss: 0.4588 - accuracy: 0.9553 - val_loss: 0.3703 - ...
   val_accuracy: 0.9857
Epoch 5/50
91/96 [===========================>..] - ETA: 0s - loss: 0.4229 - accuracy: 0.9605
Epoch 5: val_accuracy improved from 0.98696 to 0.98957, saving model to /kaggle/working/CNN_Model_Fold_0_Checkpoint.h5
96/96 [==============================] - 1s 10ms/step - loss: 0.4247 - accuracy: 0.9605 - val_loss: 0.3525 - ...
   val_accuracy: 0.9896
```

```
Epoch 6/50
91/96 [==========================>..] - ETA: 0s - loss: 0.4004 - accuracy: 0.9677
Epoch 6: val_accuracy did not improve from 0.98957
96/96 [==============================] - 1s 10ms/step - loss: 0.3969 - accuracy: 0.9687 - val_loss: 0.3415 - ...
   val_accuracy: 0.9857
Epoch 7/50
91/96 [==========================>..] - ETA: 0s - loss: 0.3931 - accuracy: 0.9629
Epoch 7: val_accuracy did not improve from 0.98957
96/96 [==============================] - 1s 10ms/step - loss: 0.3900 - accuracy: 0.9641 - val_loss: 0.3291 - ...
   val_accuracy: 0.9831
Epoch 8/50
91/96 [==========================>..] - ETA: 0s - loss: 0.3643 - accuracy: 0.9701
Epoch 8: val_accuracy did not improve from 0.98957
96/96 [==============================] - 1s 10ms/step - loss: 0.3664 - accuracy: 0.9697 - val_loss: 0.3183 - ...
   val_accuracy: 0.9844
Epoch 9/50
91/96 [==========================>..] - ETA: 0s - loss: 0.3640 - accuracy: 0.9691
Epoch 9: val_accuracy did not improve from 0.98957
96/96 [==============================] - 1s 10ms/step - loss: 0.3623 - accuracy: 0.9694 - val_loss: 0.3080 - ...
   val_accuracy: 0.9857
Epoch 10/50
96/96 [==============================] - ETA: 0s - loss: 0.3625 - accuracy: 0.9661
Epoch 10: val_accuracy did not improve from 0.98957
96/96 [==============================] - 1s 10ms/step - loss: 0.3625 - accuracy: 0.9661 - val_loss: 0.3024 - ...
   val_accuracy: 0.9844
Epoch 11/50
91/96 [==========================>..] - ETA: 0s - loss: 0.3434 - accuracy: 0.9705
Epoch 11: val_accuracy did not improve from 0.98957
96/96 [==============================] - 1s 10ms/step - loss: 0.3479 - accuracy: 0.9694 - val_loss: 0.2978 - ...
   val_accuracy: 0.9883
Epoch 12/50
91/96 [==========================>..] - ETA: 0s - loss: 0.3349 - accuracy: 0.9742
Epoch 12: val_accuracy did not improve from 0.98957
96/96 [==============================] - 1s 10ms/step - loss: 0.3337 - accuracy: 0.9746 - val_loss: 0.2930 - ...
   val_accuracy: 0.9870
Epoch 13/50
91/96 [==========================>..] - ETA: 0s - loss: 0.3192 - accuracy: 0.9742
Epoch 13: val_accuracy did not improve from 0.98957
```

```
96/96 [==============================] - 1s 10ms/step - loss: 0.3192 - accuracy: 0.9746 - val_loss: 0.2879 - ...
   val_accuracy: 0.9883
Epoch 14/50
91/96 [===========================>..] - ETA: 0s - loss: 0.3222 - accuracy: 0.9725
Epoch 14: val_accuracy did not improve from 0.98957
96/96 [==============================] - 1s 10ms/step - loss: 0.3214 - accuracy: 0.9733 - val_loss: 0.2842 - ...
   val_accuracy: 0.9857
Epoch 15/50
94/96 [============================>.] - ETA: 0s - loss: 0.3101 - accuracy: 0.9781
Epoch 15: val_accuracy did not improve from 0.98957
96/96 [==============================] - 1s 10ms/step - loss: 0.3124 - accuracy: 0.9775 - val_loss: 0.2789 - ...
   val_accuracy: 0.9870
Epoch 16/50
91/96 [===========================>..] - ETA: 0s - loss: 0.3489 - accuracy: 0.9633
Epoch 16: val_accuracy did not improve from 0.98957
96/96 [==============================] - 1s 10ms/step - loss: 0.3476 - accuracy: 0.9638 - val_loss: 0.2872 - ...
   val_accuracy: 0.9844
Epoch 17/50
91/96 [===========================>..] - ETA: 0s - loss: 0.3094 - accuracy: 0.9773
Epoch 17: val_accuracy did not improve from 0.98957
96/96 [==============================] - 1s 10ms/step - loss: 0.3130 - accuracy: 0.9755 - val_loss: 0.2779 - ...
   val_accuracy: 0.9870
Epoch 18/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2992 - accuracy: 0.9753
Epoch 18: val_accuracy did not improve from 0.98957
96/96 [==============================] - 1s 10ms/step - loss: 0.2986 - accuracy: 0.9755 - val_loss: 0.2680 - ...
   val_accuracy: 0.9870
Epoch 19/50
91/96 [===========================>..] - ETA: 0s - loss: 0.3042 - accuracy: 0.9705
Epoch 19: val_accuracy did not improve from 0.98957
96/96 [==============================] - 1s 10ms/step - loss: 0.3023 - accuracy: 0.9713 - val_loss: 0.2672 - ...
   val_accuracy: 0.9857
Epoch 20/50
94/96 [============================>.] - ETA: 0s - loss: 0.2971 - accuracy: 0.9724
Epoch 20: val_accuracy did not improve from 0.98957
96/96 [==============================] - 1s 10ms/step - loss: 0.2962 - accuracy: 0.9729 - val_loss: 0.2628 - ...
   val_accuracy: 0.9870
Epoch 21/50
95/96 [============================>.] - ETA: 0s - loss: 0.3036 - accuracy: 0.9720
```

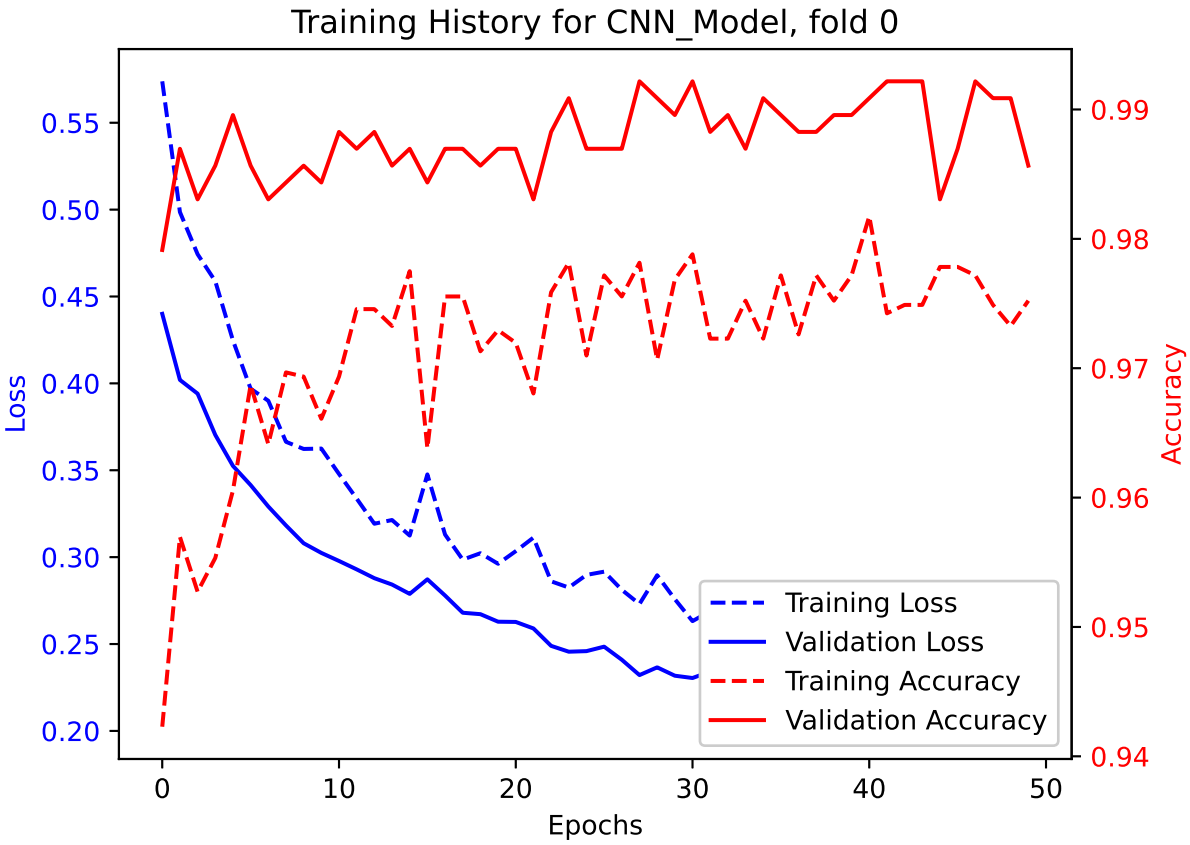
```
Epoch 21: val_accuracy did not improve from 0.98957
96/96 [==============================] - 1s 11ms/step - loss: 0.3035 - accuracy: 0.9720 - val_loss: 0.2627 - ...
   val_accuracy: 0.9870
Epoch 22/50
91/96 [===========================>..] - ETA: 0s - loss: 0.3115 - accuracy: 0.9681
Epoch 22: val_accuracy did not improve from 0.98957
96/96 [==============================] - 1s 10ms/step - loss: 0.3112 - accuracy: 0.9680 - val_loss: 0.2590 - ...
   val_accuracy: 0.9831
Epoch 23/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2857 - accuracy: 0.9756
Epoch 23: val_accuracy did not improve from 0.98957
96/96 [==============================] - 1s 10ms/step - loss: 0.2861 - accuracy: 0.9759 - val_loss: 0.2489 - ...
   val_accuracy: 0.9883
Epoch 24/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2835 - accuracy: 0.9784
Epoch 24: val_accuracy improved from 0.98957 to 0.99087, saving model to /kaggle/working/CNN_Model_Fold_0_Checkpoint.h5
96/96 [==============================] - 1s 10ms/step - loss: 0.2825 - accuracy: 0.9782 - val_loss: 0.2456 - ...
   val_accuracy: 0.9909
Epoch 25/50
95/96 [============================>.] - ETA: 0s - loss: 0.2903 - accuracy: 0.9707
Epoch 25: val_accuracy did not improve from 0.99087
96/96 [==============================] - 1s 10ms/step - loss: 0.2898 - accuracy: 0.9710 - val_loss: 0.2459 - ...
   val_accuracy: 0.9870
Epoch 26/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2915 - accuracy: 0.9777
Epoch 26: val_accuracy did not improve from 0.99087
96/96 [==============================] - 1s 10ms/step - loss: 0.2916 - accuracy: 0.9772 - val_loss: 0.2485 - ...
   val_accuracy: 0.9870
Epoch 27/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2824 - accuracy: 0.9753
Epoch 27: val_accuracy did not improve from 0.99087
96/96 [==============================] - 1s 10ms/step - loss: 0.2812 - accuracy: 0.9755 - val_loss: 0.2410 - ...
   val_accuracy: 0.9870
Epoch 28/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2748 - accuracy: 0.9773
Epoch 28: val_accuracy improved from 0.99087 to 0.99218, saving model to /kaggle/working/CNN_Model_Fold_0_Checkpoint.h5
96/96 [==============================] - 1s 10ms/step - loss: 0.2730 - accuracy: 0.9782 - val_loss: 0.2321 - ...
   val_accuracy: 0.9922
Epoch 29/50
```

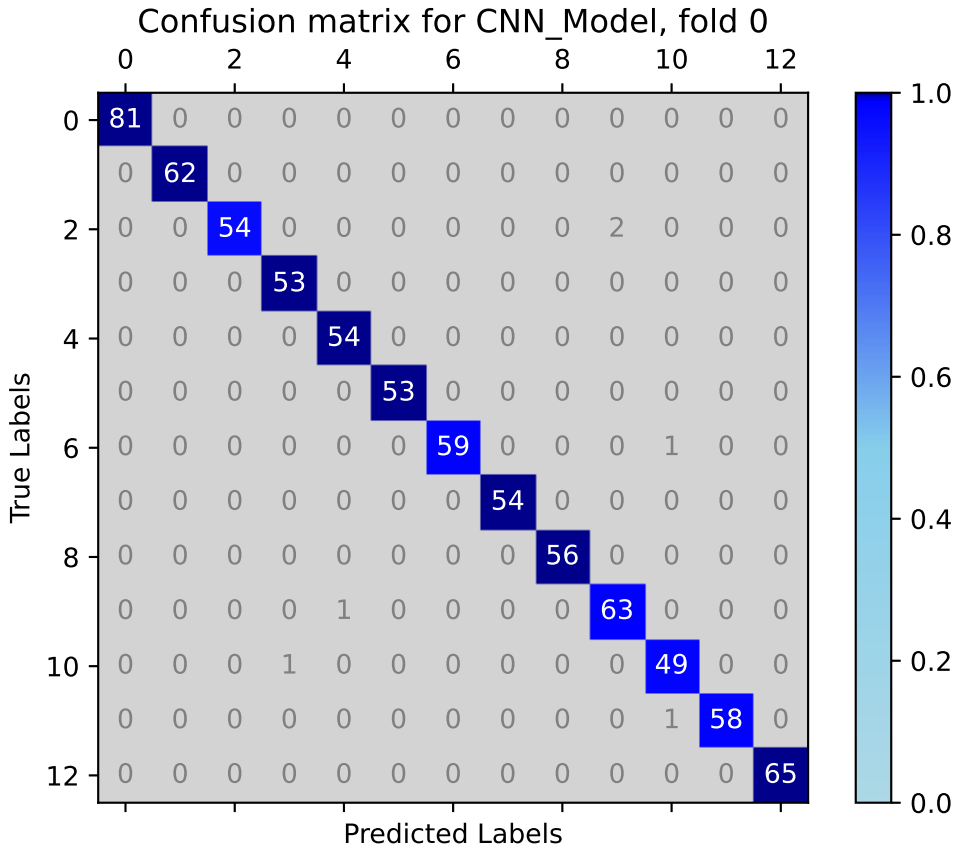
```
91/96 [==========================>..] - ETA: 0s - loss: 0.2901 - accuracy: 0.9705
Epoch 29: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2896 - accuracy: 0.9707 - val_loss: 0.2366 - ...
   val_accuracy: 0.9909
Epoch 30/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2758 - accuracy: 0.9766
Epoch 30: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2758 - accuracy: 0.9769 - val_loss: 0.2318 - ...
   val_accuracy: 0.9896
Epoch 31/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2605 - accuracy: 0.9797
Epoch 31: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2632 - accuracy: 0.9788 - val_loss: 0.2304 - ...
   val_accuracy: 0.9922
Epoch 32/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2710 - accuracy: 0.9712
Epoch 32: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2689 - accuracy: 0.9723 - val_loss: 0.2341 - ...
   val_accuracy: 0.9883
Epoch 33/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2705 - accuracy: 0.9722
Epoch 33: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2694 - accuracy: 0.9723 - val_loss: 0.2275 - ...
   val_accuracy: 0.9896
Epoch 34/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2609 - accuracy: 0.9766
Epoch 34: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2647 - accuracy: 0.9752 - val_loss: 0.2286 - ...
   val_accuracy: 0.9870
Epoch 35/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2708 - accuracy: 0.9718
Epoch 35: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2703 - accuracy: 0.9723 - val_loss: 0.2272 - ...
   val_accuracy: 0.9909
Epoch 36/50
96/96 [==============================] - ETA: 0s - loss: 0.2589 - accuracy: 0.9772
Epoch 36: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2589 - accuracy: 0.9772 - val_loss: 0.2228 - ...
   val_accuracy: 0.9896
```

```
Epoch 37/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2590 - accuracy: 0.9729
Epoch 37: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2592 - accuracy: 0.9726 - val_loss: 0.2260 - ...
   val_accuracy: 0.9883
Epoch 38/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2585 - accuracy: 0.9770
Epoch 38: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2590 - accuracy: 0.9772 - val_loss: 0.2209 - ...
   val_accuracy: 0.9883
Epoch 39/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2567 - accuracy: 0.9749
Epoch 39: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2563 - accuracy: 0.9752 - val_loss: 0.2204 - ...
   val_accuracy: 0.9896
Epoch 40/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2447 - accuracy: 0.9777
Epoch 40: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2454 - accuracy: 0.9772 - val_loss: 0.2186 - ...
   val_accuracy: 0.9896
Epoch 41/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2346 - accuracy: 0.9815
Epoch 41: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2363 - accuracy: 0.9817 - val_loss: 0.2177 - ...
   val_accuracy: 0.9909
Epoch 42/50
96/96 [==============================] - ETA: 0s - loss: 0.2605 - accuracy: 0.9742
Epoch 42: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2605 - accuracy: 0.9742 - val_loss: 0.2107 - ...
   val_accuracy: 0.9922
Epoch 43/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2534 - accuracy: 0.9753
Epoch 43: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2551 - accuracy: 0.9749 - val_loss: 0.2101 - ...
   val_accuracy: 0.9922
Epoch 44/50
96/96 [==============================] - ETA: 0s - loss: 0.2445 - accuracy: 0.9749
Epoch 44: val_accuracy did not improve from 0.99218
```

```
96/96 [==============================] - 1s 10ms/step - loss: 0.2445 - accuracy: 0.9749 - val_loss: 0.2109 - ...
   val_accuracy: 0.9922
Epoch 45/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2511 - accuracy: 0.9777
Epoch 45: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2509 - accuracy: 0.9778 - val_loss: 0.2181 - ...
   val_accuracy: 0.9831
Epoch 46/50
96/96 [==============================] - ETA: 0s - loss: 0.2366 - accuracy: 0.9778
Epoch 46: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2366 - accuracy: 0.9778 - val_loss: 0.2140 - ...
   val_accuracy: 0.9870
Epoch 47/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2464 - accuracy: 0.9766
Epoch 47: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2459 - accuracy: 0.9772 - val_loss: 0.2024 - ...
   val_accuracy: 0.9922
Epoch 48/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2488 - accuracy: 0.9749
Epoch 48: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2475 - accuracy: 0.9749 - val_loss: 0.2053 - ...
   val_accuracy: 0.9909
Epoch 49/50
96/96 [==============================] - ETA: 0s - loss: 0.2518 - accuracy: 0.9733
Epoch 49: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2518 - accuracy: 0.9733 - val_loss: 0.2073 - ...
   val_accuracy: 0.9909
Epoch 50/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2410 - accuracy: 0.9760
Epoch 50: val_accuracy did not improve from 0.99218
96/96 [==============================] - 1s 10ms/step - loss: 0.2418 - accuracy: 0.9752 - val_loss: 0.2104 - ...
   val_accuracy: 0.9857
```

```
24/24 [==============================] - 0s 3ms/step - loss: 0.2321 - accuracy: 0.9922
Test Loss CNN_Model, fold 0: 0.232126846909523, Test Accuracy CNN_Model, fold 0: 0.9921773076057434
24/24 [==============================] - 0s 2ms/step
F1 score for CNN_Model, fold 0: 0.9917991477966913
24/24 [==============================] - 0s 2ms/step
```


Training History for CNN_Model, fold 0
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.99
0.98
0.97
0.96
0.95
0.94
0
10
20
30
40
50


Confusion matrix for CNN_Model, fold 0
True Labels
Predicted Labels
0 2 4 6 8 10 12
1.0
0.8
0.6
0.4
0.2
0.0

```
Epoch 1/50
93/96 [============================>.] - ETA: 0s - loss: 0.2878 - accuracy: 0.9711
Epoch 1: val_accuracy improved from -inf to 1.00000, saving model to /kaggle/working/CNN_Model_Fold_1_Checkpoint.h5
96/96 [==============================] - 7s 15ms/step - loss: 0.2894 - accuracy: 0.9710 - val_loss: 0.1971 - ...
   val_accuracy: 1.0000
Epoch 2/50
13/96 [===>..........................] - ETA: 0s - loss: 0.3059 - accuracy: 0.9591

/opt/conda/lib/python3.10/site-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an ...
   HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras ...
   format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(

91/96 [============================>..] - ETA: 0s - loss: 0.2651 - accuracy: 0.9739
Epoch 2: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2691 - accuracy: 0.9726 - val_loss: 0.1960 - ...
   val_accuracy: 0.9987
Epoch 3/50
91/96 [============================>..] - ETA: 0s - loss: 0.2834 - accuracy: 0.9736
Epoch 3: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2809 - accuracy: 0.9746 - val_loss: 0.1977 - ...
   val_accuracy: 0.9974
Epoch 4/50
91/96 [============================>..] - ETA: 0s - loss: 0.2748 - accuracy: 0.9732
Epoch 4: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2737 - accuracy: 0.9739 - val_loss: 0.1957 - ...
   val_accuracy: 0.9974
Epoch 5/50
91/96 [============================>..] - ETA: 0s - loss: 0.2725 - accuracy: 0.9722
Epoch 5: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2718 - accuracy: 0.9729 - val_loss: 0.1940 - ...
   val_accuracy: 0.9974
Epoch 6/50
91/96 [============================>..] - ETA: 0s - loss: 0.2604 - accuracy: 0.9763
Epoch 6: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2620 - accuracy: 0.9762 - val_loss: 0.1935 - ...
   val_accuracy: 0.9974
Epoch 7/50
```

```
94/96 [===========================>.] - ETA: 0s - loss: 0.2717 - accuracy: 0.9757
Epoch 7: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2713 - accuracy: 0.9759 - val_loss: 0.1907 - ...
   val_accuracy: 0.9987
Epoch 8/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2722 - accuracy: 0.9725
Epoch 8: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2702 - accuracy: 0.9726 - val_loss: 0.1905 - ...
   val_accuracy: 0.9974
Epoch 9/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2589 - accuracy: 0.9791
Epoch 9: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2617 - accuracy: 0.9778 - val_loss: 0.1889 - ...
   val_accuracy: 0.9974
Epoch 10/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2792 - accuracy: 0.9715
Epoch 10: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2807 - accuracy: 0.9710 - val_loss: 0.1918 - ...
   val_accuracy: 0.9987
Epoch 11/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2573 - accuracy: 0.9749
Epoch 11: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2566 - accuracy: 0.9752 - val_loss: 0.1870 - ...
   val_accuracy: 0.9987
Epoch 12/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2641 - accuracy: 0.9736
Epoch 12: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2644 - accuracy: 0.9739 - val_loss: 0.1884 - ...
   val_accuracy: 0.9987
Epoch 13/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2596 - accuracy: 0.9736
Epoch 13: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2614 - accuracy: 0.9736 - val_loss: 0.1892 - ...
   val_accuracy: 0.9961
Epoch 14/50
94/96 [===========================>.] - ETA: 0s - loss: 0.2512 - accuracy: 0.9767
Epoch 14: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2514 - accuracy: 0.9769 - val_loss: 0.1866 - ...
   val_accuracy: 0.9974
```

```
Epoch 15/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2411 - accuracy: 0.9784
Epoch 15: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2440 - accuracy: 0.9772 - val_loss: 0.1830 - ...
   val_accuracy: 0.9961
Epoch 16/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2578 - accuracy: 0.9756
Epoch 16: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2553 - accuracy: 0.9759 - val_loss: 0.1900 - ...
   val_accuracy: 0.9922
Epoch 17/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2509 - accuracy: 0.9760
Epoch 17: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2522 - accuracy: 0.9762 - val_loss: 0.1841 - ...
   val_accuracy: 0.9961
Epoch 18/50
94/96 [=========================>.] - ETA: 0s - loss: 0.2731 - accuracy: 0.9688
Epoch 18: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2717 - accuracy: 0.9694 - val_loss: 0.1870 - ...
   val_accuracy: 0.9948
Epoch 19/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2574 - accuracy: 0.9694
Epoch 19: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2576 - accuracy: 0.9697 - val_loss: 0.1816 - ...
   val_accuracy: 0.9961
Epoch 20/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2410 - accuracy: 0.9787
Epoch 20: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2398 - accuracy: 0.9788 - val_loss: 0.1810 - ...
   val_accuracy: 0.9961
Epoch 21/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2630 - accuracy: 0.9708
Epoch 21: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2612 - accuracy: 0.9716 - val_loss: 0.1730 - ...
   val_accuracy: 0.9987
Epoch 22/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2303 - accuracy: 0.9784
Epoch 22: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2305 - accuracy: 0.9782 - val_loss: 0.1736 - ...
```

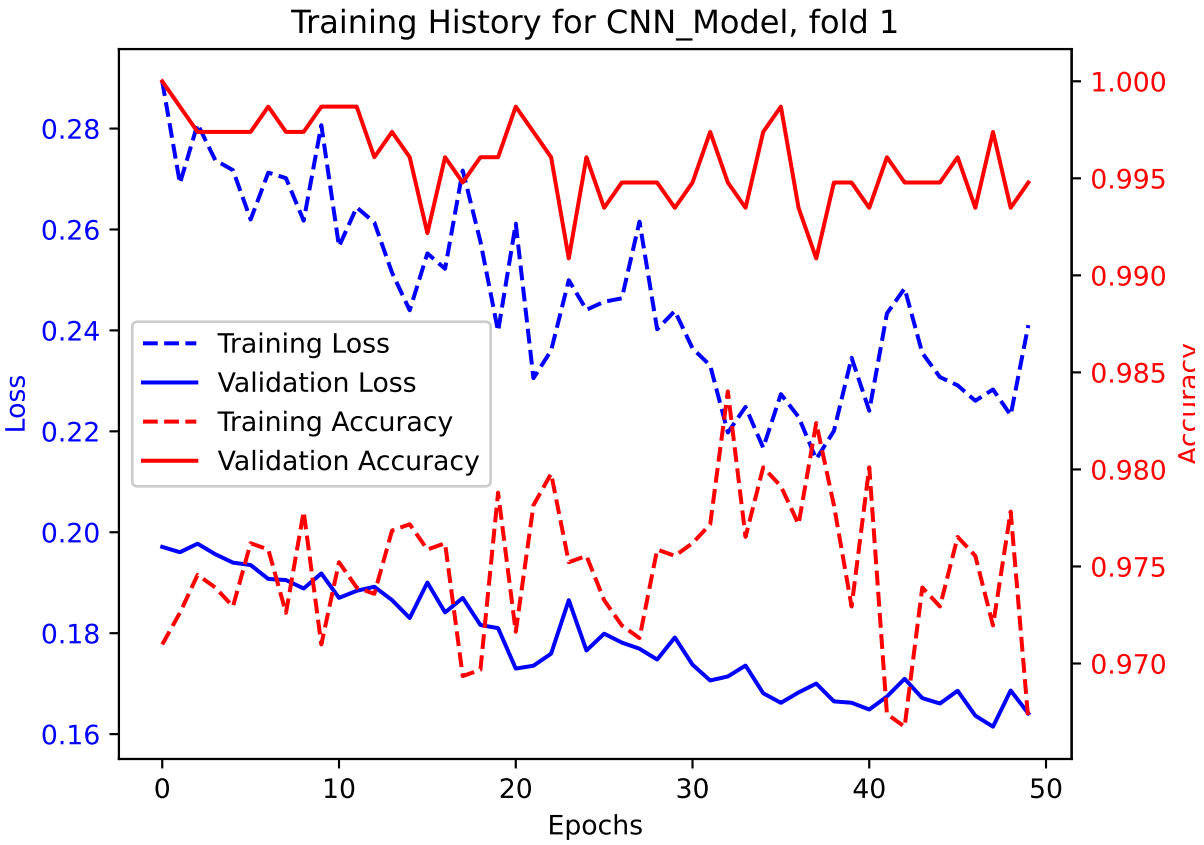
```
   val_accuracy: 0.9974
Epoch 23/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2341 - accuracy: 0.9801
Epoch 23: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2360 - accuracy: 0.9798 - val_loss: 0.1759 - ...
   val_accuracy: 0.9961
Epoch 24/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2527 - accuracy: 0.9742
Epoch 24: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2500 - accuracy: 0.9752 - val_loss: 0.1866 - ...
   val_accuracy: 0.9909
Epoch 25/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2437 - accuracy: 0.9749
Epoch 25: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2441 - accuracy: 0.9755 - val_loss: 0.1766 - ...
   val_accuracy: 0.9961
Epoch 26/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2463 - accuracy: 0.9739
Epoch 26: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2457 - accuracy: 0.9733 - val_loss: 0.1799 - ...
   val_accuracy: 0.9935
Epoch 27/50
93/96 [===========================>.] - ETA: 0s - loss: 0.2433 - accuracy: 0.9728
Epoch 27: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 12ms/step - loss: 0.2464 - accuracy: 0.9720 - val_loss: 0.1781 - ...
   val_accuracy: 0.9948
Epoch 28/50
94/96 [===========================>.] - ETA: 0s - loss: 0.2631 - accuracy: 0.9707
Epoch 28: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2616 - accuracy: 0.9713 - val_loss: 0.1769 - ...
   val_accuracy: 0.9948
Epoch 29/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2386 - accuracy: 0.9773
Epoch 29: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2402 - accuracy: 0.9759 - val_loss: 0.1748 - ...
   val_accuracy: 0.9948
Epoch 30/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2448 - accuracy: 0.9763
Epoch 30: val_accuracy did not improve from 1.00000
```

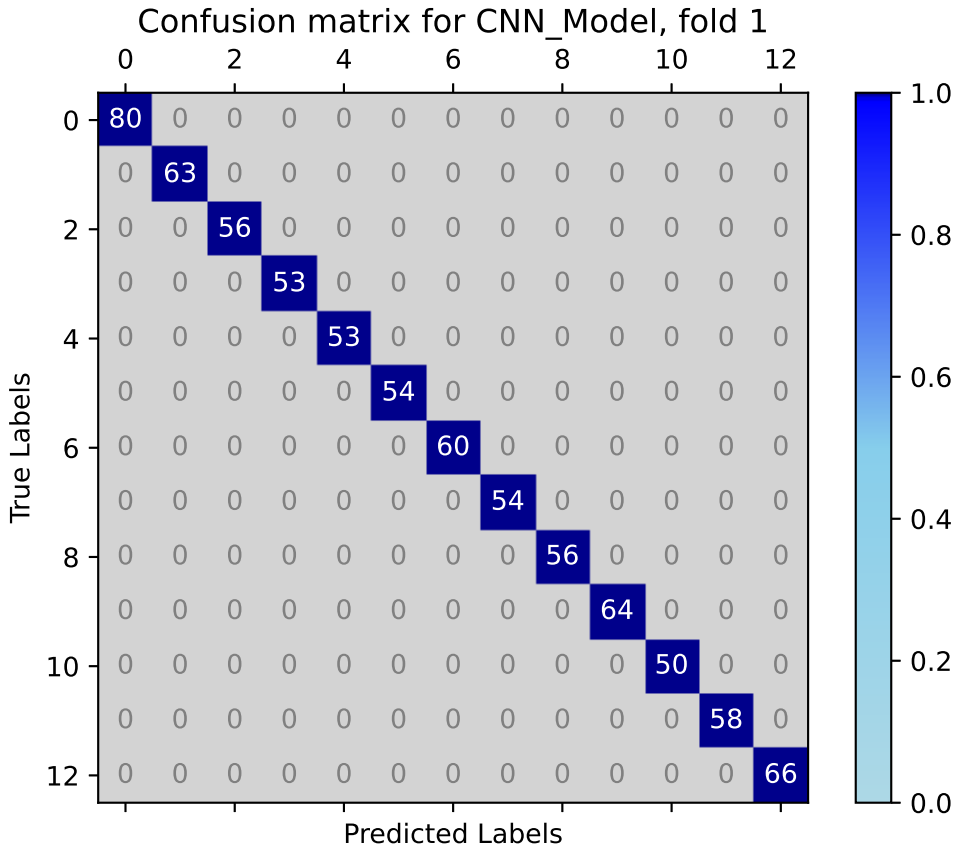
```
96/96 [==============================] - 1s 10ms/step - loss: 0.2439 - accuracy: 0.9755 - val_loss: 0.1792 - ...
   val_accuracy: 0.9935
Epoch 31/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2370 - accuracy: 0.9760
Epoch 31: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2364 - accuracy: 0.9762 - val_loss: 0.1738 - ...
   val_accuracy: 0.9948
Epoch 32/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2334 - accuracy: 0.9770
Epoch 32: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2330 - accuracy: 0.9772 - val_loss: 0.1706 - ...
   val_accuracy: 0.9974
Epoch 33/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2186 - accuracy: 0.9852
Epoch 33: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2198 - accuracy: 0.9840 - val_loss: 0.1714 - ...
   val_accuracy: 0.9948
Epoch 34/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2218 - accuracy: 0.9777
Epoch 34: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2248 - accuracy: 0.9765 - val_loss: 0.1736 - ...
   val_accuracy: 0.9935
Epoch 35/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2164 - accuracy: 0.9801
Epoch 35: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2167 - accuracy: 0.9801 - val_loss: 0.1681 - ...
   val_accuracy: 0.9974
Epoch 36/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2288 - accuracy: 0.9787
Epoch 36: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2274 - accuracy: 0.9791 - val_loss: 0.1662 - ...
   val_accuracy: 0.9987
Epoch 37/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2201 - accuracy: 0.9777
Epoch 37: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2229 - accuracy: 0.9772 - val_loss: 0.1683 - ...
   val_accuracy: 0.9935
Epoch 38/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2148 - accuracy: 0.9825
```

```
Epoch 38: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2145 - accuracy: 0.9824 - val_loss: 0.1700 - ...
   val_accuracy: 0.9909
Epoch 39/50
95/96 [============================>.] - ETA: 0s - loss: 0.2206 - accuracy: 0.9780
Epoch 39: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2201 - accuracy: 0.9782 - val_loss: 0.1665 - ...
   val_accuracy: 0.9948
Epoch 40/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2371 - accuracy: 0.9722
Epoch 40: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2346 - accuracy: 0.9729 - val_loss: 0.1662 - ...
   val_accuracy: 0.9948
Epoch 41/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2221 - accuracy: 0.9804
Epoch 41: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2241 - accuracy: 0.9801 - val_loss: 0.1649 - ...
   val_accuracy: 0.9935
Epoch 42/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2413 - accuracy: 0.9667
Epoch 42: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2434 - accuracy: 0.9674 - val_loss: 0.1675 - ...
   val_accuracy: 0.9961
Epoch 43/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2497 - accuracy: 0.9663
Epoch 43: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2483 - accuracy: 0.9667 - val_loss: 0.1710 - ...
   val_accuracy: 0.9948
Epoch 44/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2370 - accuracy: 0.9736
Epoch 44: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2355 - accuracy: 0.9739 - val_loss: 0.1671 - ...
   val_accuracy: 0.9948
Epoch 45/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2316 - accuracy: 0.9725
Epoch 45: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2307 - accuracy: 0.9729 - val_loss: 0.1661 - ...
   val_accuracy: 0.9948
Epoch 46/50
```

```
93/96 [============================>.] - ETA: 0s - loss: 0.2293 - accuracy: 0.9772
Epoch 46: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2291 - accuracy: 0.9765 - val_loss: 0.1686 - ...
   val_accuracy: 0.9961
Epoch 47/50
95/96 [============================>.] - ETA: 0s - loss: 0.2267 - accuracy: 0.9753
Epoch 47: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2261 - accuracy: 0.9755 - val_loss: 0.1637 - ...
   val_accuracy: 0.9935
Epoch 48/50
91/96 [============================>..] - ETA: 0s - loss: 0.2259 - accuracy: 0.9729
Epoch 48: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2283 - accuracy: 0.9720 - val_loss: 0.1615 - ...
   val_accuracy: 0.9974
Epoch 49/50
94/96 [============================>.] - ETA: 0s - loss: 0.2220 - accuracy: 0.9777
Epoch 49: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2232 - accuracy: 0.9778 - val_loss: 0.1686 - ...
   val_accuracy: 0.9935
Epoch 50/50
91/96 [============================>..] - ETA: 0s - loss: 0.2404 - accuracy: 0.9674
Epoch 50: val_accuracy did not improve from 1.00000
96/96 [==============================] - 1s 10ms/step - loss: 0.2410 - accuracy: 0.9671 - val_loss: 0.1641 - ...
   val_accuracy: 0.9948
```

```
24/24 [==============================] - 0s 3ms/step - loss: 0.1971 - accuracy: 1.0000
Test Loss CNN_Model, fold 1: 0.19710563123226166, Test Accuracy CNN_Model, fold 1: 1.0
24/24 [==============================] - 0s 2ms/step
F1 score for CNN_Model, fold 1: 1.0
24/24 [==============================] - 0s 2ms/step
```


Training History for CNN_Model, fold 1
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
0.28
0.26
0.24
0.22
0.20
0.18
0.16
1.000
0.995
0.990
0.985
0.980
0.975
0.970
0
10
20
30
40
50


Confusion matrix for CNN_Model, fold 1
True Labels
Predicted Labels
0
2
4
6
8
10
12
1.0
0.8
0.6
0.4
0.2
0.0

```
Epoch 1/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2738 - accuracy: 0.9722
Epoch 1: val_accuracy improved from -inf to 0.99609, saving model to /kaggle/working/CNN_Model_Fold_2_Checkpoint.h5
96/96 [==============================] - 7s 16ms/step - loss: 0.2746 - accuracy: 0.9716 - val_loss: 0.1986 - ...
   val_accuracy: 0.9961
Epoch 2/50
13/96 [===>..........................] - ETA: 0s - loss: 0.2855 - accuracy: 0.9688

/opt/conda/lib/python3.10/site-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an ...
   HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras ...
   format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(

91/96 [==========================>..] - ETA: 0s - loss: 0.2686 - accuracy: 0.9756
Epoch 2: val_accuracy improved from 0.99609 to 0.99739, saving model to /kaggle/working/CNN_Model_Fold_2_Checkpoint.h5
96/96 [==============================] - 1s 11ms/step - loss: 0.2708 - accuracy: 0.9752 - val_loss: 0.1982 - ...
   val_accuracy: 0.9974
Epoch 3/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2613 - accuracy: 0.9791
Epoch 3: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2628 - accuracy: 0.9782 - val_loss: 0.1975 - ...
   val_accuracy: 0.9974
Epoch 4/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2723 - accuracy: 0.9708
Epoch 4: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2719 - accuracy: 0.9713 - val_loss: 0.1979 - ...
   val_accuracy: 0.9961
Epoch 5/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2528 - accuracy: 0.9766
Epoch 5: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2564 - accuracy: 0.9752 - val_loss: 0.1960 - ...
   val_accuracy: 0.9948
Epoch 6/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2553 - accuracy: 0.9760
Epoch 6: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2556 - accuracy: 0.9759 - val_loss: 0.1951 - ...
   val_accuracy: 0.9948
Epoch 7/50
```

```
91/96 [==========================>..] - ETA: 0s - loss: 0.2656 - accuracy: 0.9749
Epoch 7: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2642 - accuracy: 0.9755 - val_loss: 0.1936 - ...
   val_accuracy: 0.9948
Epoch 8/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2764 - accuracy: 0.9705
Epoch 8: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2757 - accuracy: 0.9710 - val_loss: 0.1956 - ...
   val_accuracy: 0.9935
Epoch 9/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2640 - accuracy: 0.9736
Epoch 9: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2649 - accuracy: 0.9733 - val_loss: 0.1926 - ...
   val_accuracy: 0.9974
Epoch 10/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2540 - accuracy: 0.9729
Epoch 10: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2566 - accuracy: 0.9726 - val_loss: 0.1907 - ...
   val_accuracy: 0.9961
Epoch 11/50
95/96 [==========================>.] - ETA: 0s - loss: 0.2482 - accuracy: 0.9780
Epoch 11: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2477 - accuracy: 0.9782 - val_loss: 0.1954 - ...
   val_accuracy: 0.9922
Epoch 12/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2692 - accuracy: 0.9705
Epoch 12: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2671 - accuracy: 0.9710 - val_loss: 0.1900 - ...
   val_accuracy: 0.9961
Epoch 13/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2601 - accuracy: 0.9756
Epoch 13: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2619 - accuracy: 0.9752 - val_loss: 0.1973 - ...
   val_accuracy: 0.9896
Epoch 14/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2694 - accuracy: 0.9729
Epoch 14: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2667 - accuracy: 0.9736 - val_loss: 0.1918 - ...
   val_accuracy: 0.9935
```

```
Epoch 15/50
94/96 [============================>.] - ETA: 0s - loss: 0.2516 - accuracy: 0.9751
Epoch 15: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2511 - accuracy: 0.9752 - val_loss: 0.1888 - ...
   val_accuracy: 0.9922
Epoch 16/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2358 - accuracy: 0.9791
Epoch 16: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2372 - accuracy: 0.9795 - val_loss: 0.1872 - ...
   val_accuracy: 0.9948
Epoch 17/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2351 - accuracy: 0.9770
Epoch 17: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2353 - accuracy: 0.9765 - val_loss: 0.1900 - ...
   val_accuracy: 0.9896
Epoch 18/50
92/96 [===========================>..] - ETA: 0s - loss: 0.2428 - accuracy: 0.9766
Epoch 18: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 11ms/step - loss: 0.2425 - accuracy: 0.9765 - val_loss: 0.1902 - ...
   val_accuracy: 0.9909
Epoch 19/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2536 - accuracy: 0.9753
Epoch 19: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2521 - accuracy: 0.9752 - val_loss: 0.1869 - ...
   val_accuracy: 0.9935
Epoch 20/50
95/96 [============================>.] - ETA: 0s - loss: 0.2442 - accuracy: 0.9740
Epoch 20: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2447 - accuracy: 0.9739 - val_loss: 0.1830 - ...
   val_accuracy: 0.9935
Epoch 21/50
95/96 [============================>.] - ETA: 0s - loss: 0.2398 - accuracy: 0.9783
Epoch 21: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2392 - accuracy: 0.9785 - val_loss: 0.1806 - ...
   val_accuracy: 0.9948
Epoch 22/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2444 - accuracy: 0.9742
Epoch 22: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2452 - accuracy: 0.9739 - val_loss: 0.1812 - ...
```

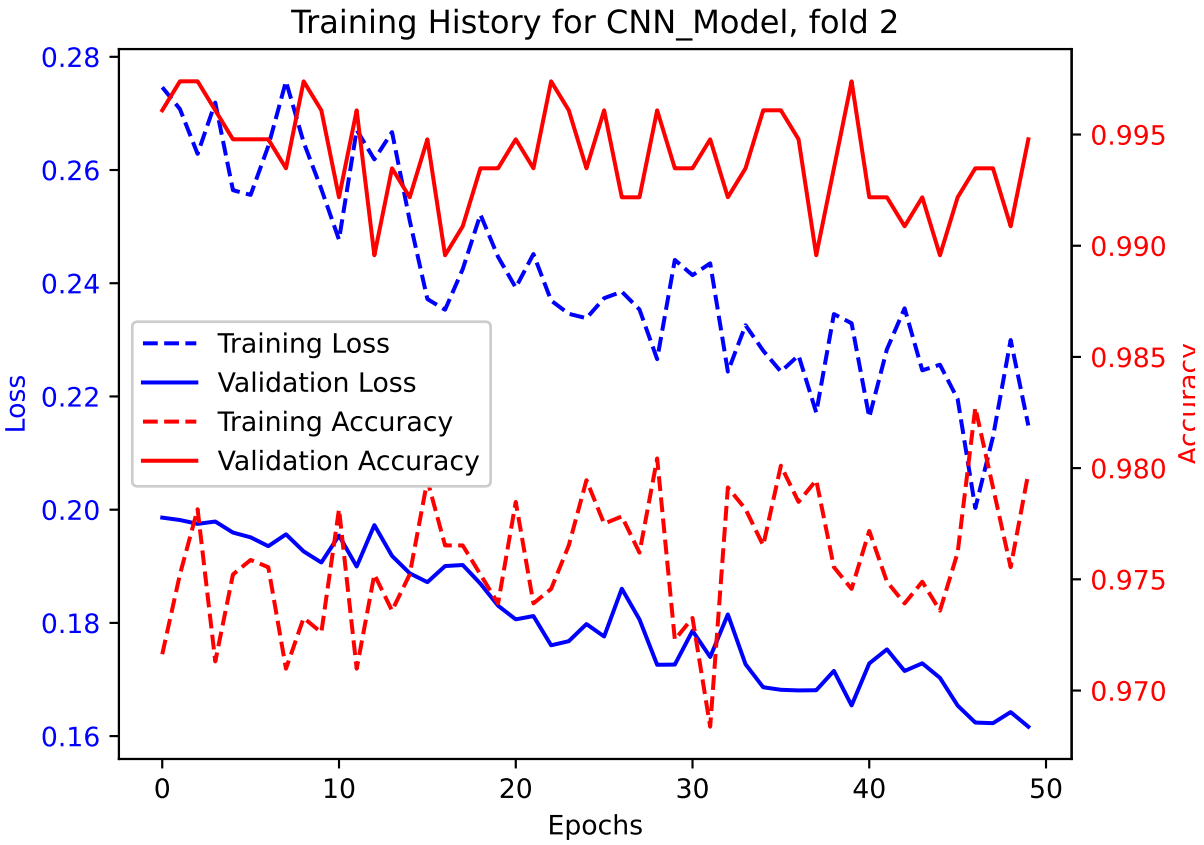
```
   val_accuracy: 0.9935
Epoch 23/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2399 - accuracy: 0.9732
Epoch 23: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2369 - accuracy: 0.9746 - val_loss: 0.1760 - ...
   val_accuracy: 0.9974
Epoch 24/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2348 - accuracy: 0.9763
Epoch 24: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2346 - accuracy: 0.9765 - val_loss: 0.1768 - ...
   val_accuracy: 0.9961
Epoch 25/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2347 - accuracy: 0.9791
Epoch 25: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2338 - accuracy: 0.9795 - val_loss: 0.1798 - ...
   val_accuracy: 0.9935
Epoch 26/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2382 - accuracy: 0.9777
Epoch 26: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2374 - accuracy: 0.9775 - val_loss: 0.1776 - ...
   val_accuracy: 0.9961
Epoch 27/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2388 - accuracy: 0.9777
Epoch 27: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2385 - accuracy: 0.9778 - val_loss: 0.1860 - ...
   val_accuracy: 0.9922
Epoch 28/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2331 - accuracy: 0.9763
Epoch 28: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2354 - accuracy: 0.9762 - val_loss: 0.1806 - ...
   val_accuracy: 0.9922
Epoch 29/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2257 - accuracy: 0.9804
Epoch 29: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2266 - accuracy: 0.9804 - val_loss: 0.1726 - ...
   val_accuracy: 0.9961
Epoch 30/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2454 - accuracy: 0.9722
Epoch 30: val_accuracy did not improve from 0.99739
```

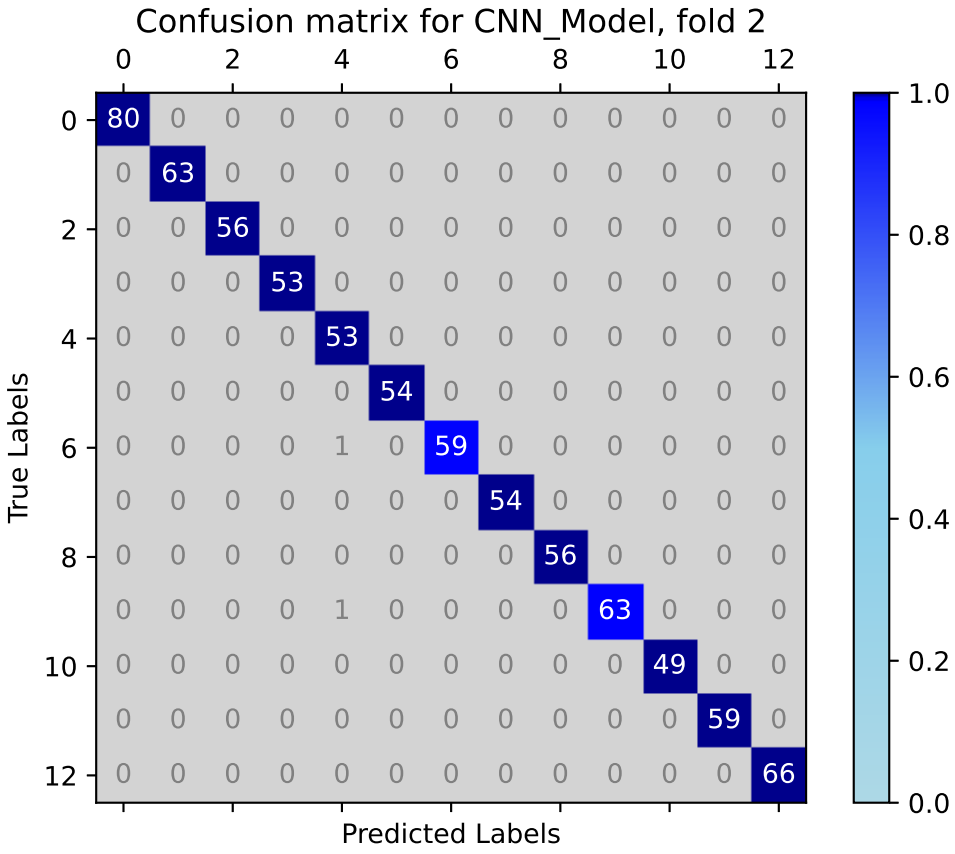
```
96/96 [==============================] - 1s 10ms/step - loss: 0.2441 - accuracy: 0.9723 - val_loss: 0.1726 - ...
    val_accuracy: 0.9935
Epoch 31/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2432 - accuracy: 0.9729
Epoch 31: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2414 - accuracy: 0.9733 - val_loss: 0.1786 - ...
    val_accuracy: 0.9935
Epoch 32/50
95/96 [============================>.] - ETA: 0s - loss: 0.2440 - accuracy: 0.9681
Epoch 32: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2435 - accuracy: 0.9684 - val_loss: 0.1740 - ...
    val_accuracy: 0.9948
Epoch 33/50
93/96 [============================>.] - ETA: 0s - loss: 0.2250 - accuracy: 0.9788
Epoch 33: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 11ms/step - loss: 0.2244 - accuracy: 0.9791 - val_loss: 0.1815 - ...
    val_accuracy: 0.9922
Epoch 34/50
92/96 [===========================>..] - ETA: 0s - loss: 0.2318 - accuracy: 0.9789
Epoch 34: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2326 - accuracy: 0.9782 - val_loss: 0.1727 - ...
    val_accuracy: 0.9935
Epoch 35/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2247 - accuracy: 0.9763
Epoch 35: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2281 - accuracy: 0.9765 - val_loss: 0.1686 - ...
    val_accuracy: 0.9961
Epoch 36/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2256 - accuracy: 0.9797
Epoch 36: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2244 - accuracy: 0.9801 - val_loss: 0.1682 - ...
    val_accuracy: 0.9961
Epoch 37/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2238 - accuracy: 0.9787
Epoch 37: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2272 - accuracy: 0.9785 - val_loss: 0.1681 - ...
    val_accuracy: 0.9948
Epoch 38/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2156 - accuracy: 0.9801
```

```
Epoch 38: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2172 - accuracy: 0.9795 - val_loss: 0.1681 - ...
   val_accuracy: 0.9896
Epoch 39/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2328 - accuracy: 0.9763
Epoch 39: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2346 - accuracy: 0.9755 - val_loss: 0.1715 - ...
   val_accuracy: 0.9935
Epoch 40/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2338 - accuracy: 0.9742
Epoch 40: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2329 - accuracy: 0.9746 - val_loss: 0.1654 - ...
   val_accuracy: 0.9974
Epoch 41/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2189 - accuracy: 0.9763
Epoch 41: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2163 - accuracy: 0.9772 - val_loss: 0.1728 - ...
   val_accuracy: 0.9922
Epoch 42/50
94/96 [============================>.] - ETA: 0s - loss: 0.2263 - accuracy: 0.9754
Epoch 42: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2283 - accuracy: 0.9749 - val_loss: 0.1753 - ...
   val_accuracy: 0.9922
Epoch 43/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2319 - accuracy: 0.9746
Epoch 43: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2356 - accuracy: 0.9739 - val_loss: 0.1715 - ...
   val_accuracy: 0.9909
Epoch 44/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2249 - accuracy: 0.9749
Epoch 44: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2246 - accuracy: 0.9749 - val_loss: 0.1729 - ...
   val_accuracy: 0.9922
Epoch 45/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2238 - accuracy: 0.9739
Epoch 45: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2256 - accuracy: 0.9736 - val_loss: 0.1703 - ...
   val_accuracy: 0.9896
Epoch 46/50
```

```
91/96 [==========================>..] - ETA: 0s - loss: 0.2217 - accuracy: 0.9756
Epoch 46: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2195 - accuracy: 0.9762 - val_loss: 0.1654 - ...
   val_accuracy: 0.9922
Epoch 47/50
91/96 [==========================>..] - ETA: 0s - loss: 0.1987 - accuracy: 0.9821
Epoch 47: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2003 - accuracy: 0.9827 - val_loss: 0.1624 - ...
   val_accuracy: 0.9935
Epoch 48/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2110 - accuracy: 0.9801
Epoch 48: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2129 - accuracy: 0.9791 - val_loss: 0.1623 - ...
   val_accuracy: 0.9935
Epoch 49/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2312 - accuracy: 0.9753
Epoch 49: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2300 - accuracy: 0.9755 - val_loss: 0.1642 - ...
   val_accuracy: 0.9909
Epoch 50/50
96/96 [==============================] - ETA: 0s - loss: 0.2149 - accuracy: 0.9798
Epoch 50: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2149 - accuracy: 0.9798 - val_loss: 0.1617 - ...
   val_accuracy: 0.9948
```

```
24/24 [==============================] - 0s 3ms/step - loss: 0.1982 - accuracy: 0.9974
Test Loss CNN_Model, fold 2: 0.19817110896110535, Test Accuracy CNN_Model, fold 2: 0.9973924160003662
24/24 [==============================] - 0s 2ms/step
F1 score for CNN_Model, fold 2: 0.9973233926453009
24/24 [==============================] - 0s 2ms/step
```


Training History for CNN_Model, fold 2
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
0.28
0.26
0.24
0.22
0.20
0.18
0.16
0.995
0.990
0.985
0.980
0.975
0.970
0
10
20
30
40
50


Confusion matrix for CNN_Model, fold 2
True Labels
Predicted Labels
0 2 4 6 8 10 12
1.0
0.8
0.6
0.4
0.2
0.0

```
Epoch 1/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2572 - accuracy: 0.9763
Epoch 1: val_accuracy improved from -inf to 0.99739, saving model to /kaggle/working/CNN_Model_Fold_3_Checkpoint.h5
96/96 [==============================] - 7s 14ms/step - loss: 0.2590 - accuracy: 0.9765 - val_loss: 0.1894 - ...
   val_accuracy: 0.9974
Epoch 2/50
13/96 [===>..........................] - ETA: 0s - loss: 0.2641 - accuracy: 0.9663

/opt/conda/lib/python3.10/site-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an ...
   HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras ...
   format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(

91/96 [==========================>..] - ETA: 0s - loss: 0.2684 - accuracy: 0.9729
Epoch 2: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2703 - accuracy: 0.9733 - val_loss: 0.1953 - ...
   val_accuracy: 0.9935
Epoch 3/50
95/96 [==========================>.] - ETA: 0s - loss: 0.2791 - accuracy: 0.9714
Epoch 3: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2788 - accuracy: 0.9716 - val_loss: 0.1930 - ...
   val_accuracy: 0.9948
Epoch 4/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2604 - accuracy: 0.9732
Epoch 4: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2589 - accuracy: 0.9742 - val_loss: 0.1925 - ...
   val_accuracy: 0.9935
Epoch 5/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2574 - accuracy: 0.9736
Epoch 5: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2552 - accuracy: 0.9742 - val_loss: 0.1950 - ...
   val_accuracy: 0.9935
Epoch 6/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2593 - accuracy: 0.9766
Epoch 6: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2582 - accuracy: 0.9769 - val_loss: 0.1911 - ...
   val_accuracy: 0.9948
Epoch 7/50
```

```
94/96 [============================>.] - ETA: 0s - loss: 0.2490 - accuracy: 0.9764
Epoch 7: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 12ms/step - loss: 0.2483 - accuracy: 0.9769 - val_loss: 0.1931 - ...
   val_accuracy: 0.9922
Epoch 8/50
94/96 [============================>.] - ETA: 0s - loss: 0.2613 - accuracy: 0.9701
Epoch 8: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2602 - accuracy: 0.9707 - val_loss: 0.2082 - ...
   val_accuracy: 0.9896
Epoch 9/50
91/96 [============================>..] - ETA: 0s - loss: 0.2638 - accuracy: 0.9712
Epoch 9: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2619 - accuracy: 0.9716 - val_loss: 0.1909 - ...
   val_accuracy: 0.9935
Epoch 10/50
91/96 [============================>..] - ETA: 0s - loss: 0.2615 - accuracy: 0.9749
Epoch 10: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2607 - accuracy: 0.9759 - val_loss: 0.1909 - ...
   val_accuracy: 0.9922
Epoch 11/50
91/96 [============================>..] - ETA: 0s - loss: 0.2430 - accuracy: 0.9780
Epoch 11: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2420 - accuracy: 0.9785 - val_loss: 0.1907 - ...
   val_accuracy: 0.9922
Epoch 12/50
91/96 [============================>..] - ETA: 0s - loss: 0.2447 - accuracy: 0.9766
Epoch 12: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2450 - accuracy: 0.9762 - val_loss: 0.1881 - ...
   val_accuracy: 0.9922
Epoch 13/50
91/96 [============================>..] - ETA: 0s - loss: 0.2706 - accuracy: 0.9674
Epoch 13: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2746 - accuracy: 0.9658 - val_loss: 0.1895 - ...
   val_accuracy: 0.9935
Epoch 14/50
91/96 [============================>..] - ETA: 0s - loss: 0.2648 - accuracy: 0.9698
Epoch 14: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2688 - accuracy: 0.9690 - val_loss: 0.1863 - ...
   val_accuracy: 0.9974
```

```
Epoch 15/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2395 - accuracy: 0.9791
Epoch 15: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2421 - accuracy: 0.9791 - val_loss: 0.1858 - ...
   val_accuracy: 0.9935
Epoch 16/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2298 - accuracy: 0.9804
Epoch 16: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2291 - accuracy: 0.9808 - val_loss: 0.1875 - ...
   val_accuracy: 0.9922
Epoch 17/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2612 - accuracy: 0.9725
Epoch 17: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2602 - accuracy: 0.9729 - val_loss: 0.1829 - ...
   val_accuracy: 0.9948
Epoch 18/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2352 - accuracy: 0.9784
Epoch 18: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2453 - accuracy: 0.9759 - val_loss: 0.1841 - ...
   val_accuracy: 0.9935
Epoch 19/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2409 - accuracy: 0.9753
Epoch 19: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2405 - accuracy: 0.9749 - val_loss: 0.1855 - ...
   val_accuracy: 0.9922
Epoch 20/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2520 - accuracy: 0.9732
Epoch 20: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2518 - accuracy: 0.9726 - val_loss: 0.1798 - ...
   val_accuracy: 0.9935
Epoch 21/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2409 - accuracy: 0.9756
Epoch 21: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2421 - accuracy: 0.9755 - val_loss: 0.1821 - ...
   val_accuracy: 0.9948
Epoch 22/50
92/96 [=========================>..] - ETA: 0s - loss: 0.2339 - accuracy: 0.9786
Epoch 22: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2354 - accuracy: 0.9782 - val_loss: 0.1774 - ...
```

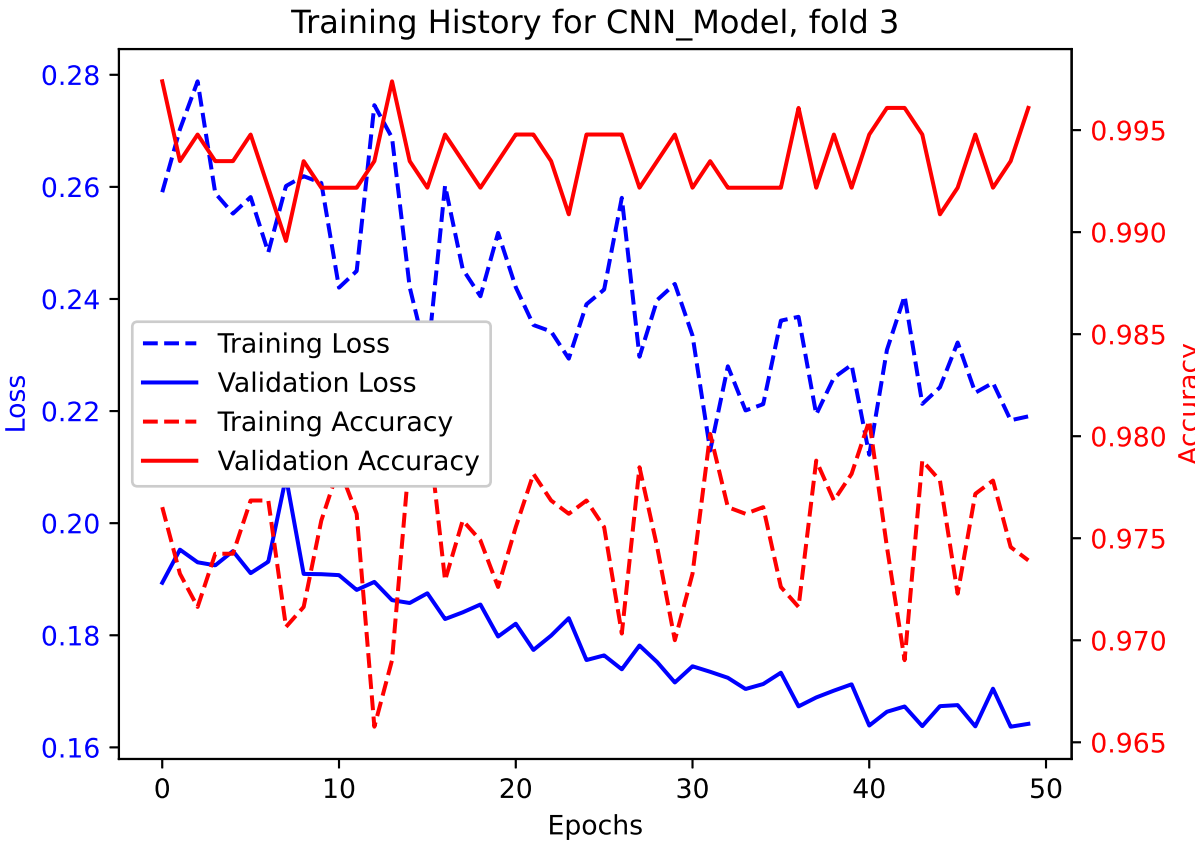
```
   val_accuracy: 0.9948
Epoch 23/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2338 - accuracy: 0.9766
Epoch 23: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2342 - accuracy: 0.9769 - val_loss: 0.1799 - ...
   val_accuracy: 0.9935
Epoch 24/50
94/96 [==========================>.] - ETA: 0s - loss: 0.2291 - accuracy: 0.9761
Epoch 24: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2294 - accuracy: 0.9762 - val_loss: 0.1831 - ...
   val_accuracy: 0.9909
Epoch 25/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2402 - accuracy: 0.9766
Epoch 25: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2391 - accuracy: 0.9769 - val_loss: 0.1756 - ...
   val_accuracy: 0.9948
Epoch 26/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2390 - accuracy: 0.9763
Epoch 26: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2417 - accuracy: 0.9755 - val_loss: 0.1764 - ...
   val_accuracy: 0.9948
Epoch 27/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2557 - accuracy: 0.9712
Epoch 27: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2581 - accuracy: 0.9703 - val_loss: 0.1739 - ...
   val_accuracy: 0.9948
Epoch 28/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2277 - accuracy: 0.9791
Epoch 28: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2297 - accuracy: 0.9785 - val_loss: 0.1782 - ...
   val_accuracy: 0.9922
Epoch 29/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2412 - accuracy: 0.9739
Epoch 29: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2398 - accuracy: 0.9746 - val_loss: 0.1752 - ...
   val_accuracy: 0.9935
Epoch 30/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2397 - accuracy: 0.9701
Epoch 30: val_accuracy did not improve from 0.99739
```

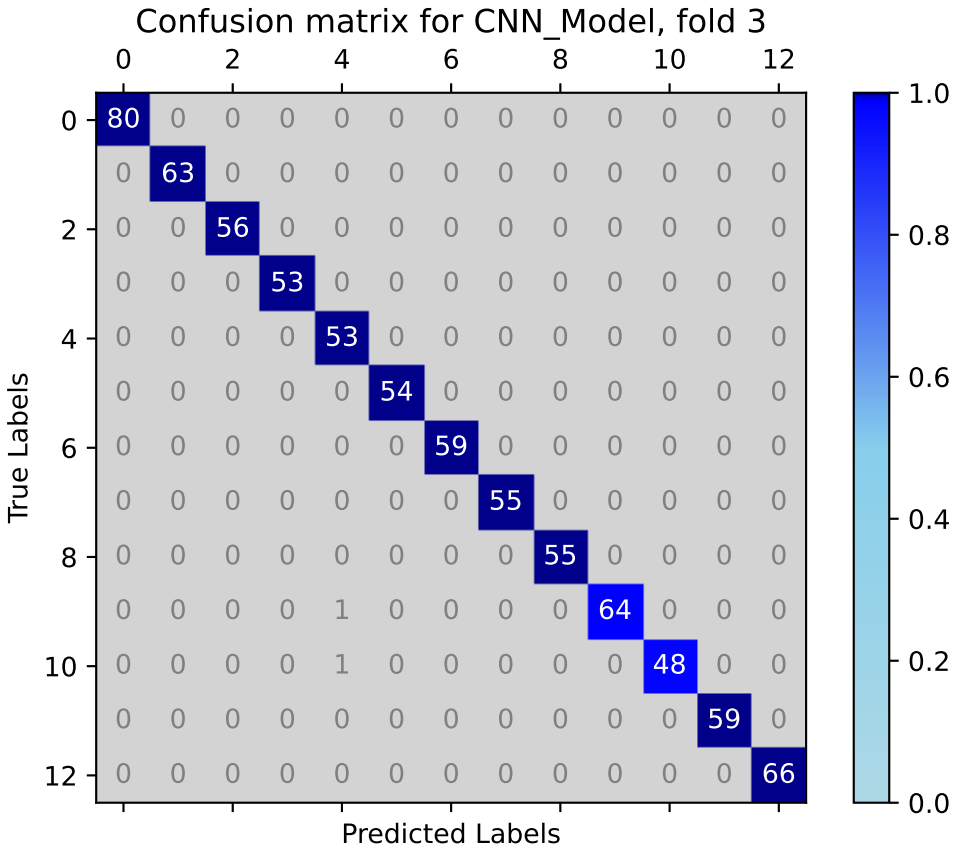
```
96/96 [==============================] - 1s 10ms/step - loss: 0.2427 - accuracy: 0.9700 - val_loss: 0.1716 - ...
   val_accuracy: 0.9948
Epoch 31/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2337 - accuracy: 0.9736
Epoch 31: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2335 - accuracy: 0.9733 - val_loss: 0.1745 - ...
   val_accuracy: 0.9922
Epoch 32/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2136 - accuracy: 0.9797
Epoch 32: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2130 - accuracy: 0.9801 - val_loss: 0.1735 - ...
   val_accuracy: 0.9935
Epoch 33/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2253 - accuracy: 0.9773
Epoch 33: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2280 - accuracy: 0.9765 - val_loss: 0.1724 - ...
   val_accuracy: 0.9922
Epoch 34/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2205 - accuracy: 0.9760
Epoch 34: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2201 - accuracy: 0.9762 - val_loss: 0.1704 - ...
   val_accuracy: 0.9922
Epoch 35/50
95/96 [============================>.] - ETA: 0s - loss: 0.2212 - accuracy: 0.9766
Epoch 35: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2212 - accuracy: 0.9765 - val_loss: 0.1713 - ...
   val_accuracy: 0.9922
Epoch 36/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2320 - accuracy: 0.9739
Epoch 36: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2361 - accuracy: 0.9726 - val_loss: 0.1733 - ...
   val_accuracy: 0.9922
Epoch 37/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2371 - accuracy: 0.9715
Epoch 37: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2368 - accuracy: 0.9716 - val_loss: 0.1673 - ...
   val_accuracy: 0.9961
Epoch 38/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2151 - accuracy: 0.9797
```

```
Epoch 38: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2194 - accuracy: 0.9788 - val_loss: 0.1689 - ...
   val_accuracy: 0.9922
Epoch 39/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2267 - accuracy: 0.9763
Epoch 39: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2260 - accuracy: 0.9769 - val_loss: 0.1701 - ...
   val_accuracy: 0.9948
Epoch 40/50
96/96 [==============================] - ETA: 0s - loss: 0.2282 - accuracy: 0.9782
Epoch 40: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 11ms/step - loss: 0.2282 - accuracy: 0.9782 - val_loss: 0.1713 - ...
   val_accuracy: 0.9922
Epoch 41/50
93/96 [============================>.] - ETA: 0s - loss: 0.2134 - accuracy: 0.9805
Epoch 41: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2122 - accuracy: 0.9808 - val_loss: 0.1639 - ...
   val_accuracy: 0.9948
Epoch 42/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2290 - accuracy: 0.9749
Epoch 42: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2309 - accuracy: 0.9746 - val_loss: 0.1664 - ...
   val_accuracy: 0.9961
Epoch 43/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2407 - accuracy: 0.9691
Epoch 43: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2405 - accuracy: 0.9690 - val_loss: 0.1673 - ...
   val_accuracy: 0.9961
Epoch 44/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2181 - accuracy: 0.9801
Epoch 44: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2213 - accuracy: 0.9788 - val_loss: 0.1638 - ...
   val_accuracy: 0.9948
Epoch 45/50
94/96 [============================>.] - ETA: 0s - loss: 0.2245 - accuracy: 0.9774
Epoch 45: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2242 - accuracy: 0.9778 - val_loss: 0.1674 - ...
   val_accuracy: 0.9909
Epoch 46/50
```

```
96/96 [==============================] - ETA: 0s - loss: 0.2322 - accuracy: 0.9723
Epoch 46: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2322 - accuracy: 0.9723 - val_loss: 0.1675 - ...
   val_accuracy: 0.9922
Epoch 47/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2201 - accuracy: 0.9784
Epoch 47: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2233 - accuracy: 0.9772 - val_loss: 0.1637 - ...
   val_accuracy: 0.9948
Epoch 48/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2246 - accuracy: 0.9777
Epoch 48: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2250 - accuracy: 0.9778 - val_loss: 0.1705 - ...
   val_accuracy: 0.9922
Epoch 49/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2182 - accuracy: 0.9749
Epoch 49: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2184 - accuracy: 0.9746 - val_loss: 0.1637 - ...
   val_accuracy: 0.9935
Epoch 50/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2197 - accuracy: 0.9739
Epoch 50: val_accuracy did not improve from 0.99739
96/96 [==============================] - 1s 10ms/step - loss: 0.2191 - accuracy: 0.9739 - val_loss: 0.1642 - ...
   val_accuracy: 0.9961
```

```
24/24 [==============================] - 0s 3ms/step - loss: 0.1894 - accuracy: 0.9974
Test Loss CNN_Model, fold 3: 0.18937250971794128, Test Accuracy CNN_Model, fold 3: 0.9973924160003662
24/24 [==============================] - 0s 2ms/step
F1 score for CNN_Model, fold 3: 0.9971861742420361
24/24 [==============================] - 0s 2ms/step
```


Training History for CNN_Model, fold 3
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
0.28
0.26
0.24
0.22
0.20
0.18
0.16
0.995
0.990
0.985
0.980
0.975
0.970
0.965
0
10
20
30
40
50


Confusion matrix for CNN_Model, fold 3
True Labels
Predicted Labels
0
2
4
6
8
10
12
1.0
0.8
0.6
0.4
0.2
0.0

```
Epoch 1/50
96/96 [==============================] - ETA: 0s - loss: 0.2716 - accuracy: 0.9720
Epoch 1: val_accuracy improved from -inf to 0.99478, saving model to /kaggle/working/CNN_Model_Fold_4_Checkpoint.h5
96/96 [==============================] - 7s 17ms/step - loss: 0.2716 - accuracy: 0.9720 - val_loss: 0.1917 - ...
    val_accuracy: 0.9948
Epoch 2/50
 7/96 [=>............................] - ETA: 0s - loss: 0.2468 - accuracy: 0.9821

/opt/conda/lib/python3.10/site-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an ...
    HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras ...
    format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(

91/96 [===========================>..] - ETA: 0s - loss: 0.2660 - accuracy: 0.9760
Epoch 2: val_accuracy did not improve from 0.99478
96/96 [==============================] - 1s 10ms/step - loss: 0.2655 - accuracy: 0.9762 - val_loss: 0.1936 - ...
    val_accuracy: 0.9948
Epoch 3/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2600 - accuracy: 0.9732
Epoch 3: val_accuracy did not improve from 0.99478
96/96 [==============================] - 1s 10ms/step - loss: 0.2626 - accuracy: 0.9726 - val_loss: 0.1922 - ...
    val_accuracy: 0.9948
Epoch 4/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2683 - accuracy: 0.9763
Epoch 4: val_accuracy did not improve from 0.99478
96/96 [==============================] - 1s 10ms/step - loss: 0.2672 - accuracy: 0.9769 - val_loss: 0.1969 - ...
    val_accuracy: 0.9922
Epoch 5/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2732 - accuracy: 0.9715
Epoch 5: val_accuracy improved from 0.99478 to 0.99608, saving model to /kaggle/working/CNN_Model_Fold_4_Checkpoint.h5
96/96 [==============================] - 1s 11ms/step - loss: 0.2729 - accuracy: 0.9716 - val_loss: 0.1912 - ...
    val_accuracy: 0.9961
Epoch 6/50
93/96 [============================>.] - ETA: 0s - loss: 0.2592 - accuracy: 0.9728
Epoch 6: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2572 - accuracy: 0.9736 - val_loss: 0.1930 - ...
    val_accuracy: 0.9935
Epoch 7/50
```

```
91/96 [==========================>..] - ETA: 0s - loss: 0.2510 - accuracy: 0.9791
Epoch 7: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2502 - accuracy: 0.9795 - val_loss: 0.1948 - ...
   val_accuracy: 0.9909
Epoch 8/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2573 - accuracy: 0.9749
Epoch 8: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2564 - accuracy: 0.9749 - val_loss: 0.1920 - ...
   val_accuracy: 0.9935
Epoch 9/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2709 - accuracy: 0.9677
Epoch 9: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2714 - accuracy: 0.9681 - val_loss: 0.2257 - ...
   val_accuracy: 0.9817
Epoch 10/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2440 - accuracy: 0.9770
Epoch 10: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2439 - accuracy: 0.9772 - val_loss: 0.1912 - ...
   val_accuracy: 0.9948
Epoch 11/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2522 - accuracy: 0.9732
Epoch 11: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2518 - accuracy: 0.9736 - val_loss: 0.1939 - ...
   val_accuracy: 0.9922
Epoch 12/50
95/96 [==========================>.] - ETA: 0s - loss: 0.2504 - accuracy: 0.9766
Epoch 12: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2507 - accuracy: 0.9762 - val_loss: 0.1894 - ...
   val_accuracy: 0.9922
Epoch 13/50
95/96 [==========================>.] - ETA: 0s - loss: 0.2469 - accuracy: 0.9766
Epoch 13: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 11ms/step - loss: 0.2473 - accuracy: 0.9762 - val_loss: 0.1896 - ...
   val_accuracy: 0.9948
Epoch 14/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2456 - accuracy: 0.9773
Epoch 14: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2446 - accuracy: 0.9772 - val_loss: 0.1915 - ...
   val_accuracy: 0.9922
```

```
Epoch 15/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2337 - accuracy: 0.9770
Epoch 15: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2342 - accuracy: 0.9772 - val_loss: 0.1933 - ...
   val_accuracy: 0.9909
Epoch 16/50
94/96 [=========================>.] - ETA: 0s - loss: 0.2382 - accuracy: 0.9771
Epoch 16: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2396 - accuracy: 0.9762 - val_loss: 0.1896 - ...
   val_accuracy: 0.9909
Epoch 17/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2404 - accuracy: 0.9739
Epoch 17: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2418 - accuracy: 0.9736 - val_loss: 0.1912 - ...
   val_accuracy: 0.9922
Epoch 18/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2460 - accuracy: 0.9760
Epoch 18: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2513 - accuracy: 0.9749 - val_loss: 0.1828 - ...
   val_accuracy: 0.9935
Epoch 19/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2504 - accuracy: 0.9753
Epoch 19: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2534 - accuracy: 0.9746 - val_loss: 0.1852 - ...
   val_accuracy: 0.9922
Epoch 20/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2438 - accuracy: 0.9739
Epoch 20: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2462 - accuracy: 0.9726 - val_loss: 0.1887 - ...
   val_accuracy: 0.9883
Epoch 21/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2342 - accuracy: 0.9797
Epoch 21: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2374 - accuracy: 0.9791 - val_loss: 0.1865 - ...
   val_accuracy: 0.9909
Epoch 22/50
91/96 [=========================>..] - ETA: 0s - loss: 0.2479 - accuracy: 0.9742
Epoch 22: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2476 - accuracy: 0.9743 - val_loss: 0.1854 - ...
```

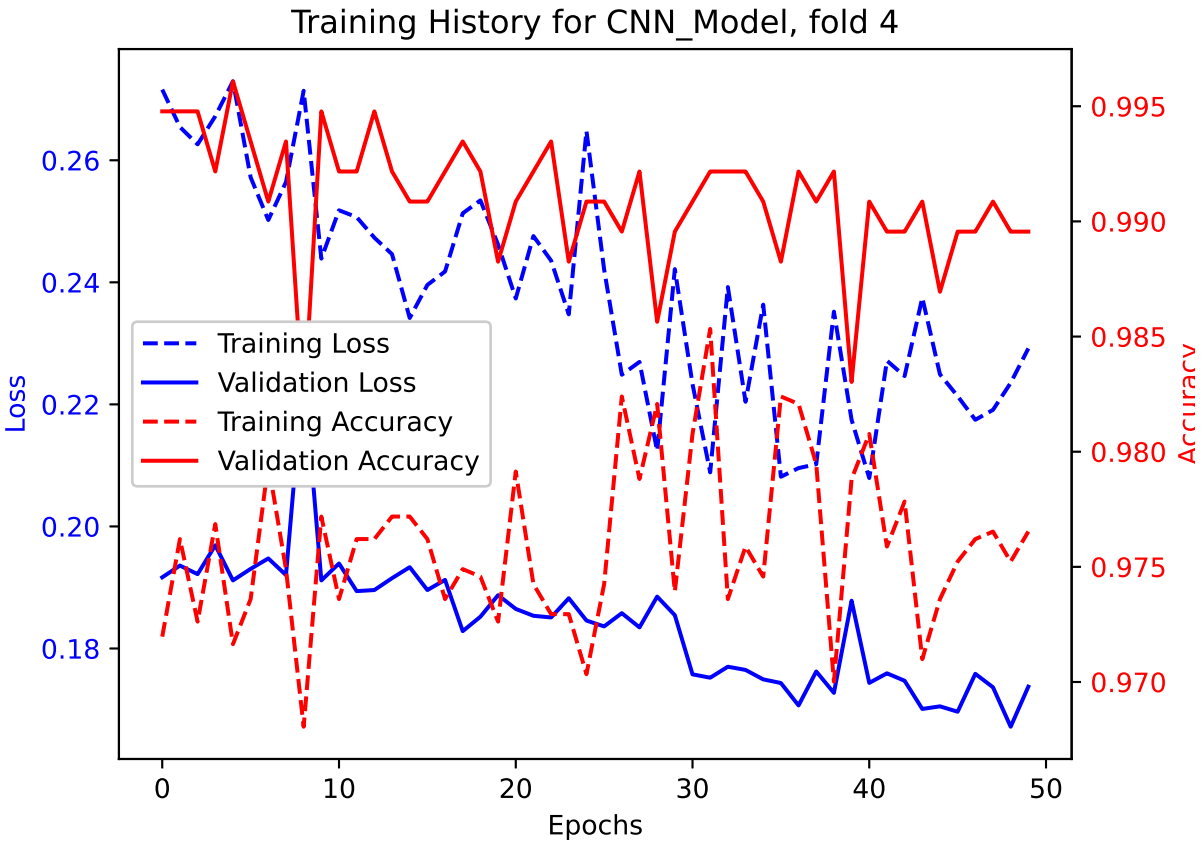
```
   val_accuracy: 0.9922
Epoch 23/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2457 - accuracy: 0.9725
Epoch 23: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2436 - accuracy: 0.9729 - val_loss: 0.1851 - ...
   val_accuracy: 0.9935
Epoch 24/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2325 - accuracy: 0.9742
Epoch 24: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2348 - accuracy: 0.9729 - val_loss: 0.1882 - ...
   val_accuracy: 0.9883
Epoch 25/50
92/96 [==========================>..] - ETA: 0s - loss: 0.2678 - accuracy: 0.9694
Epoch 25: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 11ms/step - loss: 0.2648 - accuracy: 0.9703 - val_loss: 0.1846 - ...
   val_accuracy: 0.9909
Epoch 26/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2412 - accuracy: 0.9742
Epoch 26: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2421 - accuracy: 0.9743 - val_loss: 0.1836 - ...
   val_accuracy: 0.9909
Epoch 27/50
95/96 [==========================>.] - ETA: 0s - loss: 0.2213 - accuracy: 0.9829
Epoch 27: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2249 - accuracy: 0.9824 - val_loss: 0.1858 - ...
   val_accuracy: 0.9896
Epoch 28/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2273 - accuracy: 0.9784
Epoch 28: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2270 - accuracy: 0.9788 - val_loss: 0.1835 - ...
   val_accuracy: 0.9922
Epoch 29/50
96/96 [==============================] - ETA: 0s - loss: 0.2124 - accuracy: 0.9821
Epoch 29: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2124 - accuracy: 0.9821 - val_loss: 0.1885 - ...
   val_accuracy: 0.9856
Epoch 30/50
91/96 [==========================>..] - ETA: 0s - loss: 0.2436 - accuracy: 0.9732
Epoch 30: val_accuracy did not improve from 0.99608
```

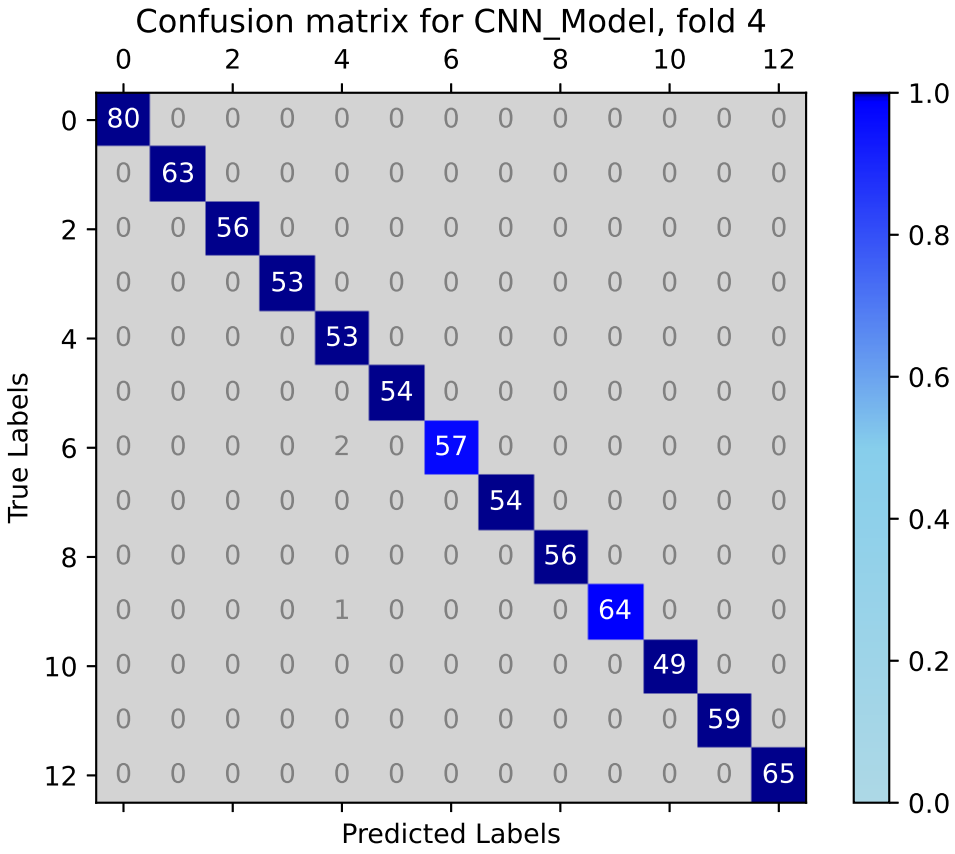
```
96/96 [==============================] - 1s 10ms/step - loss: 0.2422 - accuracy: 0.9739 - val_loss: 0.1855 - ...
    val_accuracy: 0.9896
Epoch 31/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2243 - accuracy: 0.9808
Epoch 31: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2233 - accuracy: 0.9808 - val_loss: 0.1758 - ...
    val_accuracy: 0.9909
Epoch 32/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2099 - accuracy: 0.9849
Epoch 32: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2089 - accuracy: 0.9853 - val_loss: 0.1752 - ...
    val_accuracy: 0.9922
Epoch 33/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2396 - accuracy: 0.9736
Epoch 33: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2392 - accuracy: 0.9736 - val_loss: 0.1770 - ...
    val_accuracy: 0.9922
Epoch 34/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2200 - accuracy: 0.9756
Epoch 34: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2204 - accuracy: 0.9759 - val_loss: 0.1765 - ...
    val_accuracy: 0.9922
Epoch 35/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2382 - accuracy: 0.9739
Epoch 35: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2364 - accuracy: 0.9746 - val_loss: 0.1749 - ...
    val_accuracy: 0.9909
Epoch 36/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2085 - accuracy: 0.9828
Epoch 36: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2081 - accuracy: 0.9824 - val_loss: 0.1743 - ...
    val_accuracy: 0.9883
Epoch 37/50
95/96 [============================>.] - ETA: 0s - loss: 0.2098 - accuracy: 0.9819
Epoch 37: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2096 - accuracy: 0.9821 - val_loss: 0.1707 - ...
    val_accuracy: 0.9922
Epoch 38/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2092 - accuracy: 0.9801
```

```
Epoch 38: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2102 - accuracy: 0.9795 - val_loss: 0.1762 - ...
   val_accuracy: 0.9909
Epoch 39/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2369 - accuracy: 0.9698
Epoch 39: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2352 - accuracy: 0.9700 - val_loss: 0.1727 - ...
   val_accuracy: 0.9922
Epoch 40/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2174 - accuracy: 0.9787
Epoch 40: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2175 - accuracy: 0.9788 - val_loss: 0.1878 - ...
   val_accuracy: 0.9830
Epoch 41/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2089 - accuracy: 0.9804
Epoch 41: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2079 - accuracy: 0.9808 - val_loss: 0.1744 - ...
   val_accuracy: 0.9909
Epoch 42/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2275 - accuracy: 0.9760
Epoch 42: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2272 - accuracy: 0.9759 - val_loss: 0.1759 - ...
   val_accuracy: 0.9896
Epoch 43/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2229 - accuracy: 0.9780
Epoch 43: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2246 - accuracy: 0.9778 - val_loss: 0.1747 - ...
   val_accuracy: 0.9896
Epoch 44/50
91/96 [===========================>..] - ETA: 0s - loss: 0.2398 - accuracy: 0.9701
Epoch 44: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2375 - accuracy: 0.9710 - val_loss: 0.1701 - ...
   val_accuracy: 0.9909
Epoch 45/50
93/96 [============================>.] - ETA: 0s - loss: 0.2254 - accuracy: 0.9731
Epoch 45: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 11ms/step - loss: 0.2249 - accuracy: 0.9736 - val_loss: 0.1705 - ...
   val_accuracy: 0.9869
Epoch 46/50
```

```
95/96 [============================>.] - ETA: 0s - loss: 0.2220 - accuracy: 0.9750
Epoch 46: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 11ms/step - loss: 0.2213 - accuracy: 0.9752 - val_loss: 0.1696 - ...
   val_accuracy: 0.9896
Epoch 47/50
91/96 [============================>..] - ETA: 0s - loss: 0.2162 - accuracy: 0.9763
Epoch 47: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2175 - accuracy: 0.9762 - val_loss: 0.1758 - ...
   val_accuracy: 0.9896
Epoch 48/50
91/96 [============================>..] - ETA: 0s - loss: 0.2142 - accuracy: 0.9777
Epoch 48: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2191 - accuracy: 0.9765 - val_loss: 0.1736 - ...
   val_accuracy: 0.9909
Epoch 49/50
91/96 [============================>..] - ETA: 0s - loss: 0.2257 - accuracy: 0.9749
Epoch 49: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2235 - accuracy: 0.9752 - val_loss: 0.1672 - ...
   val_accuracy: 0.9896
Epoch 50/50
91/96 [============================>..] - ETA: 0s - loss: 0.2281 - accuracy: 0.9763
Epoch 50: val_accuracy did not improve from 0.99608
96/96 [==============================] - 1s 10ms/step - loss: 0.2292 - accuracy: 0.9765 - val_loss: 0.1737 - ...
   val_accuracy: 0.9896
```

```
24/24 [==============================] - 0s 3ms/step - loss: 0.1912 - accuracy: 0.9961
Test Loss CNN_Model, fold 4: 0.19115708768367767, Test Accuracy CNN_Model, fold 4: 0.9960835576057434
24/24 [==============================] - 0s 2ms/step
F1 score for CNN_Model, fold 4: 0.9959602882250264
24/24 [==============================] - 0s 2ms/step
```

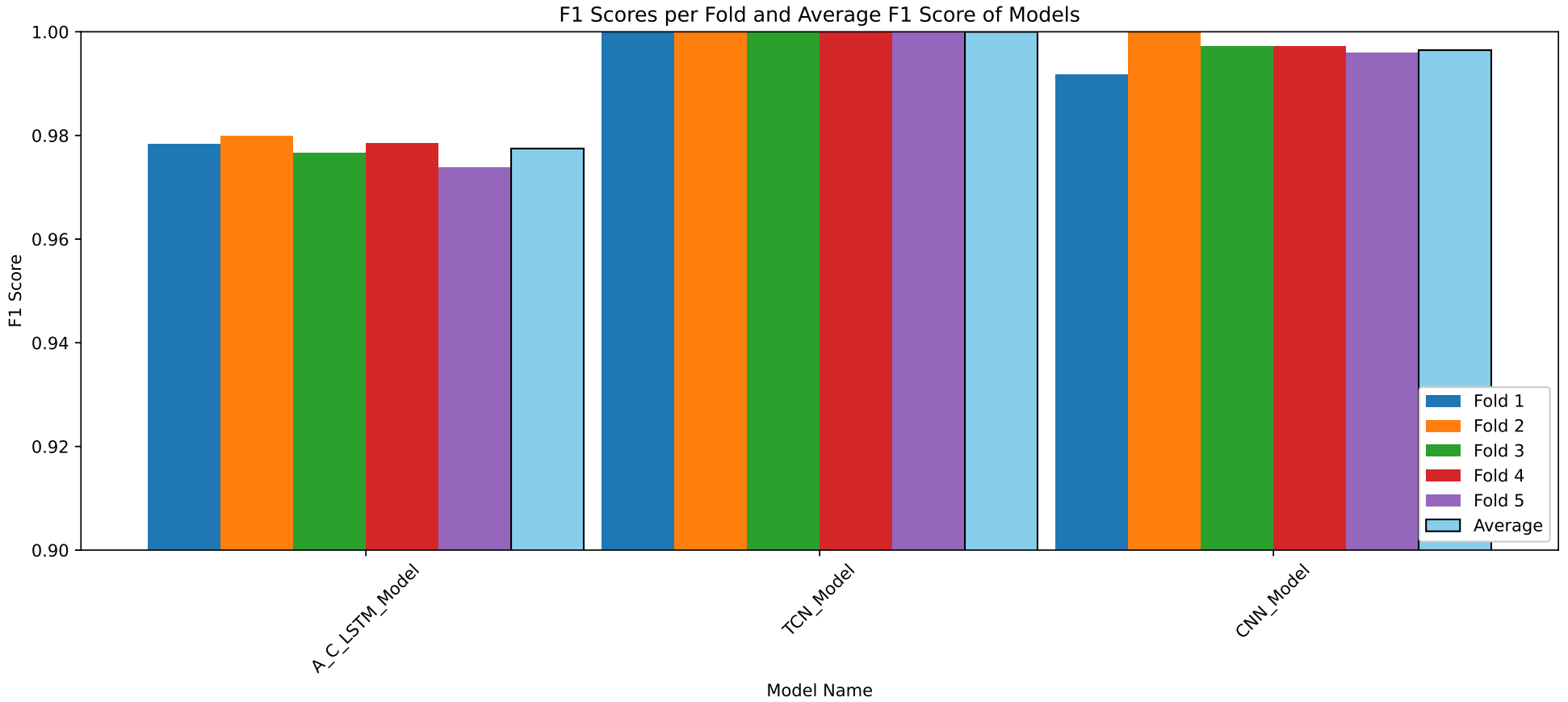

Training History for CNN_Model, fold 4
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
0.26
0.24
0.22
0.20
0.18
0.995
0.990
0.985
0.980
0.975
0.970
0
10
20
30
40
50


Confusion matrix for CNN_Model, fold 4
True Labels
Predicted Labels
0 2 4 6 8 10 12
1.0
0.8
0.6
0.4
0.2
0.0

RESULTS for CNN Model

```
Average F1 Score for CNN_Model: 0.996453800581811
F1 scores for all folds for CNN_Model: [0.9917991477966913, 1.0, 0.9973233926453009, 0.9971861742420361, 0.9959602882250264]
```

Results from Average-F1, K-fold cross-validation



Model Name	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average F1
A_C_LSTM_Model	0.9783559218526776	0.9799765088683448	0.9766877451617154	0.9785225793293548	0.9738313786126882	0.9774748267649562
TCN_Model	1.0	1.0	1.0	1.0	1.0	1.0
CNN_Model	0.9917991477966913	1.0	0.9973233926453009	0.9971861742420361	0.9959602882250264	0.996453800581811

```
!zip -r UIA_IMU_ID_Models_Val.zip /kaggle/working
```

```
  adding: kaggle/working/ (stored 0%)
  adding: kaggle/working/Training_History_fold4_CNN_Model.eps (deflated 70%)
  adding: kaggle/working/TCN_Model_Fold_2_Checkpoint.h5 (deflated 20%)
  adding: kaggle/working/Training_History_fold1_CNN_Model.eps (deflated 71%)
  adding: kaggle/working/f1_scores_grouped_performance.jpeg (deflated 49%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_3_A_C_LSTM_Model.eps (deflated 98%)
  adding: kaggle/working/Training_History_fold3_CNN_Model.eps (deflated 70%)
  adding: kaggle/working/A_C_LSTM_Model_Fold_1_Checkpoint.h5 (deflated 9%)
  adding: kaggle/working/Training_History_fold1_A_C_LSTM_Model.eps (deflated 70%)
  adding: kaggle/working/f1_scores_grouped_performance.eps (deflated 69%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_2_A_C_LSTM_Model.eps (deflated 98%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_2_TCN_Model.eps (deflated 98%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_0_TCN_Model.eps (deflated 98%)
  adding: kaggle/working/Training_History_fold3_A_C_LSTM_Model.eps (deflated 71%)
  adding: kaggle/working/TCN_Model_Fold_1_Checkpoint.h5 (deflated 21%)
  adding: kaggle/working/CNN_Model_Fold_3_Checkpoint.h5 (deflated 29%)
  adding: kaggle/working/Training_History_fold4_A_C_LSTM_Model.eps (deflated 70%)
  adding: kaggle/working/Training_History_fold1_TCN_Model.eps (deflated 71%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_4_CNN_Model.eps (deflated 98%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_0_A_C_LSTM_Model.eps (deflated 98%)
  adding: kaggle/working/CNN_Model_Fold_1_Checkpoint.h5 (deflated 29%)
  adding: kaggle/working/Training_History_fold2_TCN_Model.eps (deflated 71%)
  adding: kaggle/working/f1_scores_table.eps (deflated 70%)
  adding: kaggle/working/.virtual_documents/ (stored 0%)
  adding: kaggle/working/CNN_Model_Fold_4_Checkpoint.h5 (deflated 29%)
  adding: kaggle/working/TCN_Model_Fold_0_Checkpoint.h5 (deflated 20%)
  adding: kaggle/working/A_C_LSTM_Model_Fold_4_Checkpoint.h5 (deflated 10%)
  adding: kaggle/working/A_C_LSTM_Model_Fold_0_Checkpoint.h5 (deflated 9%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_1_CNN_Model.eps (deflated 98%)
  adding: kaggle/working/Training_History_fold2_CNN_Model.eps (deflated 70%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_3_TCN_Model.eps (deflated 98%)
  adding: kaggle/working/Training_History_fold0_CNN_Model.eps (deflated 70%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_1_A_C_LSTM_Model.eps (deflated 98%)
```

```
  adding: kaggle/working/TCN_Model_Fold_3_Checkpoint.h5 (deflated 20%)
  adding: kaggle/working/Training_History_fold3_TCN_Model.eps (deflated 72%)
  adding: kaggle/working/Training_History_fold2_A_C_LSTM_Model.eps (deflated 69%)
  adding: kaggle/working/CNN_Model_Fold_2_Checkpoint.h5 (deflated 29%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_3_CNN_Model.eps (deflated 98%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_2_CNN_Model.eps (deflated 98%)
  adding: kaggle/working/CNN_Model_Fold_0_Checkpoint.h5 (deflated 29%)
  adding: kaggle/working/A_C_LSTM_Model_Fold_3_Checkpoint.h5 (deflated 10%)
  adding: kaggle/working/TCN_Model_Fold_4_Checkpoint.h5 (deflated 20%)
  adding: kaggle/working/Training_History_fold4_TCN_Model.eps (deflated 72%)
  adding: kaggle/working/Training_History_fold0_A_C_LSTM_Model.eps (deflated 70%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_4_TCN_Model.eps (deflated 98%)
  adding: kaggle/working/Training_History_fold0_TCN_Model.eps (deflated 71%)
  adding: kaggle/working/f1_scores_table.jpeg (deflated 29%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_1_TCN_Model.eps (deflated 98%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_4_A_C_LSTM_Model.eps (deflated 98%)
  adding: kaggle/working/A_C_LSTM_Model_Fold_2_Checkpoint.h5 (deflated 9%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_0_CNN_Model.eps (deflated 98%)
```



```
print(model_metrics)
```

```
{'A_C_LSTM_Model': {'F1_Scores': [0.9783559218526776, 0.9799765088683448, 0.9766877451617154, 0.9785225793293548, 0.9738313786126882], 'Av
```

Topman: Feature importance analysis / feature selection using random forest and Support Vector Machine (SVM) models

```python
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/lim-movement-dataset/leg_test_features.csv
/kaggle/input/lim-movement-dataset/leg_test_raw_labeled_4_11_complete.csv
/kaggle/input/lim-movement-dataset/TopMan_complete_ID_DetLabel_Ready_12_12.parquet
/kaggle/input/lim-movement-dataset/df_TopMan_635_steps__Detection_and_ID_Labeled_Original_Index.csv
/kaggle/input/lim-movement-dataset/leg_train_raw_unlabeled.csv
/kaggle/input/lim-movement-dataset/leg_train_raw_labeled_4_11_complete.csv
/kaggle/input/lim-movement-dataset/df_GE_Tags_13_12.csv
/kaggle/input/lim-movement-dataset/TopMan_ML_ID_0_29.csv
/kaggle/input/lim-movement-dataset/TopMan_complete_ID_DetLabel_Ready_8_12.csv
/kaggle/input/lim-movement-dataset/leg_test_raw.csv
/kaggle/input/lim-movement-dataset/leg_train_raw_clean.csv
/kaggle/input/lim-movement-dataset/Leg_Raw_Full_W_Engineered_Feats_26_11.csv
/kaggle/input/lim-movement-dataset/leg_Gait_Detection_complete_raw_labeled_20_11.csv
/kaggle/input/lim-movement-dataset/leg_train_features.csv
/kaggle/input/lim-movement-dataset/TopMan_complete_12_8_ID_Det_Ready_12_8.csv
/kaggle/input/lim-movement-dataset/leg_train_raw_labeled_12_11_complete.csv
/kaggle/input/lim-movement-dataset/TopMan_635_steps.csv
```

```
/kaggle/input/lim-movement-dataset/df_TopMan_635_steps__Detection_and_ID_Labeled_Reset_Index.csv
/kaggle/input/lim-movement-dataset/leg_train_raw.csv
```

```
pd.set_option('display.max_columns', 500)
```

```
df = pd.read_csv('/kaggle/input/lim-movement-dataset/leg_train_raw_labeled_12_11_complete.csv')

#df_1 = pd.read_csv('/kaggle/input/lim-movement-dataset/leg_test_raw_labeled_4_11_complete.csv')
#df_train.head()
```

Concatenate train and test

```
df.columns
```

```
Index(['Gait_Detection_4', 'Gait_Detection_10', 'Gait_Detection_40',
       'Gait_Identification', 'Timestamp', 'q0', 'q1', 'q2', 'q3', 'MotionDeg',
       'Roll', 'Pitch', 'Yaw', 'accX', 'accY', 'accZ', 'gyrX', 'gyrY', 'gyrZ',
       'magX', 'magY', 'magZ'],
      dtype='object')
```

```
df.isna().sum()
```

```
Gait_Detection_4       0
Gait_Detection_10      0
Gait_Detection_40      0
Gait_Identification    0
Timestamp              0
q0                     0
q1                     0
q2                     0
q3                     0
MotionDeg              0
Roll                   0
Pitch                  0
Yaw                    0
accX                   0
accY                   0
accZ                   0
```

```
gyrX                    0
gyrY                    0
gyrZ                    0
magX                    0
magY                    0
magZ                    0
dtype: int64

df

        Gait_Detection_4  Gait_Detection_10  Gait_Detection_40  \
0                      0                  0                  0
1                      0                  0                  0
2                      0                  0                  0
3                      0                  0                  0
4                      0                  0                  0
...                  ...                ...                ...
101264                 0                  0                  0
101265                 0                  0                  0
101266                 0                  0                  0
101267                 0                  0                  0
101268                 0                  0                  0

        Gait_Identification     Timestamp      q0      q1      q2      q3  \
0                         0      0.000000  0.5739 -0.7689 -0.2633  0.0997
1                         0      0.012000  0.5738 -0.7689 -0.2636  0.0996
2                         0      0.024000  0.5736 -0.7690 -0.2638  0.0994
3                         0      0.036000  0.5733 -0.7692 -0.2639  0.0992
4                         0      0.048000  0.5731 -0.7693 -0.2640  0.0989
...                     ...           ...     ...     ...     ...     ...
101264                   29  13215.228067  0.5152 -0.6502  0.3419 -0.4413
101265                   29  13225.572067  0.5155 -0.6501  0.3419 -0.4411
101266                   29  13235.928067  0.5158 -0.6501  0.3418 -0.4408
101267                   29  13246.296067  0.5160 -0.6502  0.3418 -0.4404
101268                   29  13256.676067  0.5163 -0.6502  0.3417 -0.4400
```

```
        MotionDeg        Roll      Pitch        Yaw    accX    accY    accZ  \
0        0.037984  -95.547110  155.24887  139.96880  0.1513 -0.9418 -0.3095
1        0.076017  -95.518974  155.16757  139.89722  0.1484 -0.9428 -0.3125
2        0.117990  -95.504340  155.08208  139.80590  0.1499 -0.9423 -0.3237
3        0.159124  -95.509030  155.01852  139.70930  0.1362 -0.9531 -0.3286
4        0.188294  -95.497375  154.92510  139.61806  0.1479 -0.9536 -0.3222
...           ...         ...        ...        ...     ...     ...     ...
101264   4.964061 -103.596620  152.92755  178.48517  0.1655 -0.9848 -0.0717
101265   4.943203 -103.560100  153.00020  178.46739  0.1679 -0.9868 -0.0708
101266   4.923749 -103.530480  153.06064  178.45561  0.1708 -0.9907 -0.0708
101267   4.900333 -103.505930  153.11937  178.50203  0.1762 -0.9990 -0.0766
101268   4.886818 -103.472490  153.19025  178.50578  0.1743 -0.9995 -0.0747

          gyrX   gyrY   gyrZ   magX    magY    magZ
0      -0.026 -0.030 -0.047  0.000  16.213  10.664
1      -0.050 -0.022 -0.045  0.000  16.213  10.664
2      -0.061 -0.003 -0.043  0.000  16.213  10.664
3      -0.067  0.019 -0.040  0.000  16.213  10.664
4      -0.068  0.021 -0.038  0.000  16.213  10.664
...       ...    ...    ...    ...     ...     ...
101264  0.017 -0.028  0.033  6.736  -5.987  26.713
101265  0.012 -0.047  0.034  6.736  -5.987  26.713
101266  0.018 -0.061  0.030  6.736  -5.987  26.713
101267  0.005 -0.072  0.034  6.736  -5.987  26.713
101268  0.016 -0.089  0.031  6.736  -5.987  26.713

[101269 rows x 22 columns]
```

```
Columns = list(df.columns)
Columns.remove('Timestamp')

#Columns_Detection = Columns
Columns_ID = Columns

Columns_ID = [
```

```
 'Gait_Identification',
 'q0',
 'q1',
 'q2',
 'q3',
 'MotionDeg',
 'Roll',
 'Pitch',
 'Yaw',
 'accX',
 'accY',
 'accZ',
 'gyrX',
 'gyrY',
 'gyrZ',
 'magX',
 'magY',
 'magZ']

df_ID = df[Columns_ID]

df_ID
```

```
        Gait_Identification      q0      q1      q2      q3  MotionDeg  \
0                         0  0.5739 -0.7689 -0.2633  0.0997   0.037984
1                         0  0.5738 -0.7689 -0.2636  0.0996   0.076017
2                         0  0.5736 -0.7690 -0.2638  0.0994   0.117990
3                         0  0.5733 -0.7692 -0.2639  0.0992   0.159124
4                         0  0.5731 -0.7693 -0.2640  0.0989   0.188294
...                     ...     ...     ...     ...     ...        ...
101264                   29  0.5152 -0.6502  0.3419 -0.4413   4.964061
101265                   29  0.5155 -0.6501  0.3419 -0.4411   4.943203
101266                   29  0.5158 -0.6501  0.3418 -0.4408   4.923749
101267                   29  0.5160 -0.6502  0.3418 -0.4404   4.900333
101268                   29  0.5163 -0.6502  0.3417 -0.4400   4.886818
```

```
              Roll      Pitch        Yaw    accX    accY    accZ   gyrX  \
0       -95.547110  155.24887  139.96880  0.1513 -0.9418 -0.3095 -0.026
1       -95.518974  155.16757  139.89722  0.1484 -0.9428 -0.3125 -0.050
2       -95.504340  155.08208  139.80590  0.1499 -0.9423 -0.3237 -0.061
3       -95.509030  155.01852  139.70930  0.1362 -0.9531 -0.3286 -0.067
4       -95.497375  154.92510  139.61806  0.1479 -0.9536 -0.3222 -0.068
...            ...        ...        ...     ...     ...     ...    ...
101264 -103.596620  152.92755  178.48517  0.1655 -0.9848 -0.0717  0.017
101265 -103.560100  153.00020  178.46739  0.1679 -0.9868 -0.0708  0.012
101266 -103.530480  153.06064  178.45561  0.1708 -0.9907 -0.0708  0.018
101267 -103.505930  153.11937  178.50203  0.1762 -0.9990 -0.0766  0.005
101268 -103.472490  153.19025  178.50578  0.1743 -0.9995 -0.0747  0.016

         gyrY   gyrZ   magX    magY    magZ
0      -0.030 -0.047  0.000  16.213  10.664
1      -0.022 -0.045  0.000  16.213  10.664
2      -0.003 -0.043  0.000  16.213  10.664
3       0.019 -0.040  0.000  16.213  10.664
4       0.021 -0.038  0.000  16.213  10.664
...       ...    ...    ...     ...     ...
101264 -0.028  0.033  6.736  -5.987  26.713
101265 -0.047  0.034  6.736  -5.987  26.713
101266 -0.061  0.030  6.736  -5.987  26.713
101267 -0.072  0.034  6.736  -5.987  26.713
101268 -0.089  0.031  6.736  -5.987  26.713

[101269 rows x 18 columns]
```

```
Feats_ID = [
 'q0',
 'q1',
 'q2',
 'q3',
 'MotionDeg',
 'Roll',
```

```
 'Pitch',
 'Yaw',
 'accX',
 'accY',
 'accZ',
 'gyrX',
 'gyrY',
 'gyrZ',
 'magX',
 'magY',
 'magZ']


Feats_ID

['q0',
 'q1',
 'q2',
 'q3',
 'MotionDeg',
 'Roll',
 'Pitch',
 'Yaw',
 'accX',
 'accY',
 'accZ',
 'gyrX',
 'gyrY',
 'gyrZ',
 'magX',
 'magY',
 'magZ']

Feats_ID

['q0',
```

```
 'q1',
 'q2',
 'q3',
 'MotionDeg',
 'Roll',
 'Pitch',
 'Yaw',
 'accX',
 'accY',
 'accZ',
 'gyrX',
 'gyrY',
 'gyrZ',
 'magX',
 'magY',
 'magZ']
```



```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
#df[Feats] = scaler.fit_transform(df[Feats])

df_ID[Feats_ID] = scaler.fit_transform(df_ID[Feats_ID])
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:767: FutureWarning: is_sparse is deprecated and will be removed in a f
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:605: FutureWarning: is_sparse is deprecated and will be removed in a f
  if is_sparse(pd_dtype):
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:614: FutureWarning: is_sparse is deprecated and will be removed in a f
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:767: FutureWarning: is_sparse is deprecated and will be removed in a f
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:605: FutureWarning: is_sparse is deprecated and will be removed in a f
  if is_sparse(pd_dtype):
/opt/conda/lib/python3.10/site-packages/sklearn/utils/validation.py:614: FutureWarning: is_sparse is deprecated and will be removed in a f
```

```
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/tmp/ipykernel_47/4253417415.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_ID[Feats_ID] = scaler.fit_transform(df_ID[Feats_ID])

df_ID

        Gait_Identification        q0        q1        q2        q3  \
0                         0  0.742816  0.066647  0.358883  0.558561
1                         0  0.742687  0.066647  0.358719  0.558500
2                         0  0.742428  0.066588  0.358609  0.558377
3                         0  0.742040  0.066470  0.358555  0.558254
4                         0  0.741781  0.066411  0.358500  0.558069
...                     ...       ...       ...       ...       ...
101264                   29  0.666839  0.136780  0.690227  0.225945
101265                   29  0.667228  0.136839  0.690227  0.226068
101266                   29  0.667616  0.136839  0.690172  0.226253
101267                   29  0.667875  0.136780  0.690172  0.226499
101268                   29  0.668263  0.136780  0.690118  0.226745

        MotionDeg      Roll     Pitch       Yaw      accX      accY      accZ  \
0        0.000924  0.147191  0.322033  0.295213  0.523130  0.356374  0.453036
1        0.001030  0.147307  0.321665  0.294810  0.522687  0.356221  0.452578
2        0.001146  0.147367  0.321278  0.294296  0.522916  0.356298  0.450868
3        0.001261  0.147347  0.320991  0.293751  0.520825  0.354649  0.450120
4        0.001342  0.147395  0.320568  0.293237  0.522611  0.354573  0.451097
...           ...       ...       ...       ...       ...       ...       ...
101264   0.014634  0.114080  0.311534  0.512211  0.525297  0.349810  0.489349
101265   0.014576  0.114230  0.311863  0.512111  0.525664  0.349505  0.489486
101266   0.014522  0.114352  0.312136  0.512044  0.526106  0.348909  0.489486
101267   0.014457  0.114453  0.312402  0.512306  0.526930  0.347642  0.488601
101268   0.014419  0.114590  0.312722  0.512327  0.526640  0.347566  0.488891
```

```
            gyrX      gyrY      gyrZ      magX      magY      magZ
0       0.639180  0.442701  0.531497  0.500076  0.747575  0.663038
1       0.638589  0.442847  0.531581  0.500076  0.747575  0.663038
2       0.638318  0.443194  0.531665  0.500076  0.747575  0.663038
3       0.638170  0.443596  0.531791  0.500076  0.747575  0.663038
4       0.638145  0.443633  0.531875  0.500076  0.747575  0.663038
...          ...       ...       ...       ...       ...       ...
101264  0.640238  0.442737  0.534856  0.602875  0.408504  0.908083
101265  0.640115  0.442390  0.534898  0.602875  0.408504  0.908083
101266  0.640263  0.442134  0.534730  0.602875  0.408504  0.908083
101267  0.639943  0.441934  0.534898  0.602875  0.408504  0.908083
101268  0.640214  0.441623  0.534772  0.602875  0.408504  0.908083

[101269 rows x 18 columns]
```

```python
def create_train_test_sets_ID(df, test_percentage):
    # Sort the DataFrame based on 'Gait_Identification'
    df_sorted = df.sort_values(by='Gait_Identification')

    # Get unique labels
    unique_labels = np.unique(df_sorted['Gait_Identification'].values)

    train_sequences, train_labels, test_sequences, test_labels = [], [], [], []

    for label in unique_labels:
        label_df = df_sorted[df_sorted['Gait_Identification'] == label]

        # Calculate the number of sequences for the test set
        test_size = int(len(label_df) * (test_percentage / 100))

        # Split the label-specific DataFrame into training and testing sets
        label_df_test = label_df.iloc[:test_size]
        label_df_train = label_df.iloc[test_size:]
```

```
        # Extract sequences and labels from the DataFrames
        train_sequences.append(label_df_train.drop(columns=['Gait_Identification']).values)
        train_labels.append(label_df_train['Gait_Identification'].values)
        test_sequences.append(label_df_test.drop(columns=['Gait_Identification']).values)
        test_labels.append(label_df_test['Gait_Identification'].values)

    # Concatenate sequences and labels
    x_train = np.concatenate(train_sequences)
    y_train = np.concatenate(train_labels)
    x_test = np.concatenate(test_sequences)
    y_test = np.concatenate(test_labels)

    return x_train, y_train, x_test, y_test

x_train_ID_Unseq, y_train_ID_Unseq, x_test_ID_Unseq, y_test_ID_Unseq = create_train_test_sets_ID(df_ID, 25)
from sklearn.model_selection import train_test_split


unique_labels_train = np.unique(y_train_ID_Unseq)
unique_labels_test = np.unique(y_test_ID_Unseq)

print("Unique labels in training set:", unique_labels_train)
print("Unique labels in testing set:", unique_labels_test)

print(x_test_ID_Unseq.shape)
print(x_train_ID_Unseq.shape)
print(y_test_ID_Unseq.shape)
print(y_train_ID_Unseq.shape)


Unique labels in training set: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29]
Unique labels in testing set: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29]
(25307, 17)
```

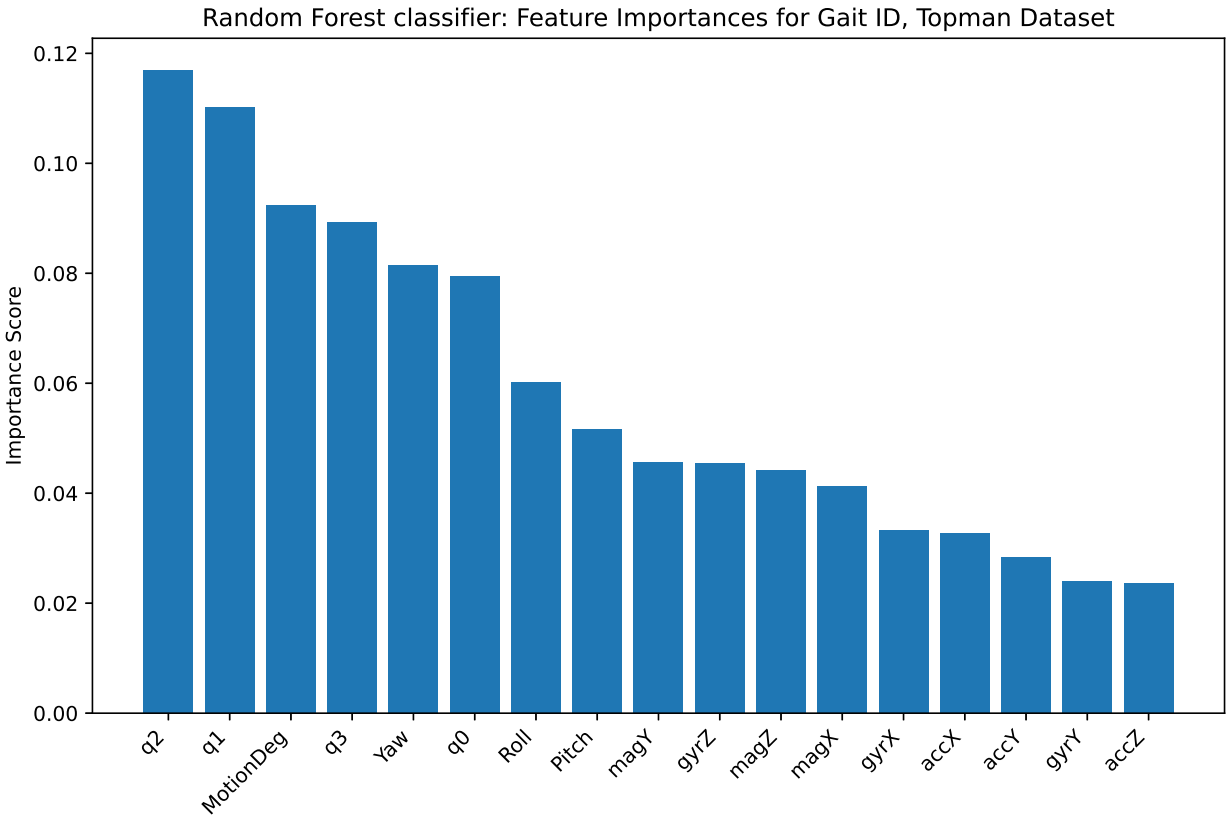
```
(75962, 17)
(25307,)
(75962,)
```

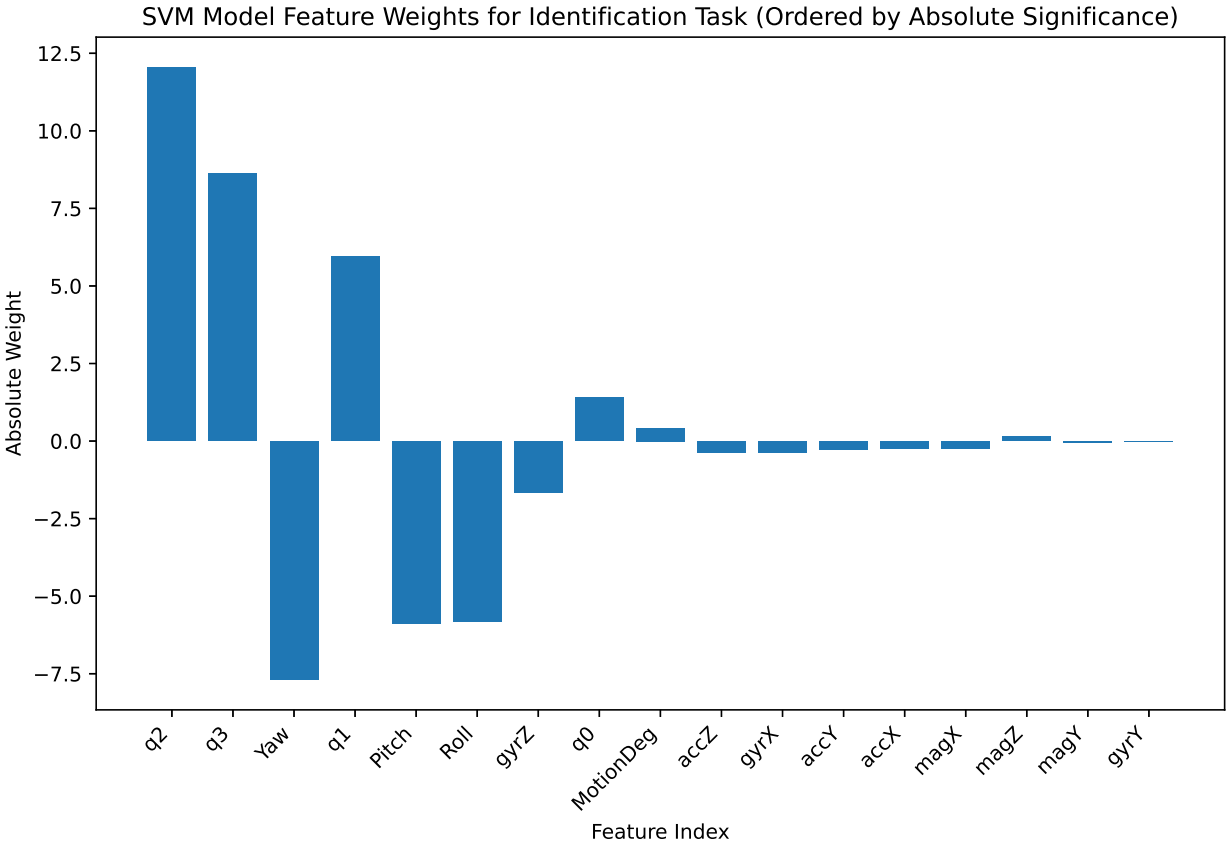
Altering function to account for label name, Detection task

```
import matplotlib.pyplot as plt

from sklearn import svm #SVM

from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt

# Feats_ID
Feats_ID = Feats_ID

# Fit the random forest classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(x_train_ID_Unseq, y_train_ID_Unseq)

# Get feature importances
feature_importances = model.feature_importances_

# Get the indices of features sorted by importance
indices_rf_identification = sorted(range(len(feature_importances)), key=lambda i: feature_importances[i], reverse=True)

# Plot the feature importances with flipped axis
plt.figure(figsize=(10, 6))
plt.bar(range(len(indices_rf_identification)), feature_importances[indices_rf_identification], align="center")
plt.xticks(range(len(indices_rf_identification)), [Feats_ID[i] for i in indices_rf_identification], rotation=45, ha='right')
plt.ylabel("Importance Score")
plt.title("Random Forest classifier: Feature Importances for Gait ID")

# Save the figure as EPS
plt.savefig('Identification_Random_Forest_feature_importances.eps', format='eps', bbox_inches='tight')
```

```
plt.show()
```

```
from sklearn import svm
import matplotlib.pyplot as plt

# Feats_ID
Feats_ID = Feats_ID


# Fit the SVM model for identification task
svm_model_Feature_Analysis_Identification = svm.SVC(kernel='linear')
svm_model_Feature_Analysis_Identification.fit(x_train_ID_Unseq, y_train_ID_Unseq)
feature_weights_Identification = svm_model_Feature_Analysis_Identification.coef_

# Get the indices of features sorted by absolute weight
indices_svm_identification = sorted(range(len(feature_weights_Identification[0])), key=lambda i: abs(feature_weights_Identification[0][i])

# Plot the feature weights with flipped axis
plt.figure(figsize=(10, 6))
plt.bar(range(len(indices_svm_identification)), [feature_weights_Identification[0][i] for i in indices_svm_identification], align="center"
plt.xticks(range(len(indices_svm_identification)), [Feats_ID[i] for i in indices_svm_identification])
plt.xlabel('Feature Index')
plt.ylabel('Absolute Weight')
plt.title('SVM Model Feature Weights for Identification Task (Ordered by Absolute Significance)')
plt.show()
```


Random Forest classifier: Feature Importances for Gait ID, Topman Dataset
Importance Score
0.12
0.10
0.08
0.06
0.04
0.02
0.00
q2
q1
MotionDeg
q3
Yaw
q0
Roll
Pitch
magY
gyrZ
magZ
magX
gyrX
accX
accY
gyrY
accZ


SVM Model Feature Weights for Identification Task (Ordered by Absolute Significance)
Absolute Weight
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
q2
q3
Yaw
q1
Pitch
Roll
gyrZ
q0
MotionDeg
accZ
gyrX
accY
accX
magX
magZ
magY
gyrY
Feature Index

```
# Top 5 features from Random Forest for Identification task
top_rf_identification = [Feats_ID[i] for i in indices_rf_identification[:5]]

# Top 5 features from SVM for Identification task
top_svm_identification = [Feats_ID[i] for i in indices_svm_identification[:5]]

# Combine top features for Identification task
combined_top_identification = set(top_rf_identification) | set(top_svm_identification)


print("RF Top Features for Identification Task:", top_rf_identification)
print("SVM Top Features for Identification Task:", top_svm_identification)
print("\n")
print("Combined Top Features for Identification Task:", combined_top_identification)
```

```
RF Top Features for Identification Task: ['q2', 'q1', 'MotionDeg', 'q3', 'Yaw']
SVM Top Features for Identification Task: ['q2', 'q3', 'Yaw', 'q1', 'Pitch']


Combined Top Features for Identification Task: {'Yaw', 'Pitch', 'q1', 'MotionDeg', 'q2', 'q3'}
```

Topman: Max F1-Averaged Stratified k-Fold of GI models

```python
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/a-c-lstm-uia-imu-id/Attentive_Conv_LSTM_model_24_1.h5
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_ID_Walking_Gait_Dataset_8_1_2024
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_IMU_9ax_WG_Dataset_W_Calibration_Data_U_21_Des_23.csv
/kaggle/input/uia-imu-gait-analysis-dataset-19-des-23/UIA_IMU_9ax_WG_Dataset_U_19_Des_23
/kaggle/input/lim-movement-dataset/leg_test_features.csv
/kaggle/input/lim-movement-dataset/leg_test_raw_labeled_4_11_complete.csv
/kaggle/input/lim-movement-dataset/TopMan_complete_ID_DetLabel_Ready_12_12.parquet
/kaggle/input/lim-movement-dataset/df_TopMan_635_steps__Detection_and_ID_Labeled_Original_Index.csv
/kaggle/input/lim-movement-dataset/leg_train_raw_unlabeled.csv
/kaggle/input/lim-movement-dataset/leg_train_raw_labeled_4_11_complete.csv
/kaggle/input/lim-movement-dataset/df_GE_Tags_13_12.csv
/kaggle/input/lim-movement-dataset/TopMan_ML_ID_0_29.csv
/kaggle/input/lim-movement-dataset/TopMan_complete_ID_DetLabel_Ready_8_12.csv
/kaggle/input/lim-movement-dataset/leg_test_raw.csv
/kaggle/input/lim-movement-dataset/leg_train_raw_clean.csv
/kaggle/input/lim-movement-dataset/Leg_Raw_Full_W_Engineered_Feats_26_11.csv
/kaggle/input/lim-movement-dataset/leg_Gait_Detection_complete_raw_labeled_20_11.csv
/kaggle/input/lim-movement-dataset/leg_train_features.csv
/kaggle/input/lim-movement-dataset/TopMan_complete_12_8_ID_Det_Ready_12_8.csv
```

```
/kaggle/input/lim-movement-dataset/leg_train_raw_labeled_12_11_complete.csv
/kaggle/input/lim-movement-dataset/TopMan_635_steps.csv
/kaggle/input/lim-movement-dataset/df_TopMan_635_steps__Detection_and_ID_Labeled_Reset_Index.csv
/kaggle/input/lim-movement-dataset/leg_train_raw.csv
/kaggle/input/tcn-trained-23/kaggle/working/TCN_H_Format.h5
/kaggle/input/topman-models-for-eval/TCN_Model_TopMan_ID.h5
/kaggle/input/topman-models-for-eval/Attentive_Conv_LSTM_model_Topman_ID.h5
/kaggle/input/topman-models-for-eval/Strict_CNN_model_Topman_ID.h5
/kaggle/input/cnn-uia-imu-id-25-1/Strict_CNN_model_24_1.h5
/kaggle/input/topman-id-tcn-trial/TCN_Model_TopMan_ID.h5
/kaggle/input/h-model-tcn-1/TCN_Model_HH.h5
```

```
df = pd.read_csv('/kaggle/input/lim-movement-dataset/TopMan_ML_ID_0_29.csv')
```

Running through preparation very quickly, this is all a repeat from the code where the models are trained

```
Feats = ['Pitch', 'q1', 'Yaw', 'MotionDeg', 'q2', 'q3']
```



```
Label = ['Gait_Identification']

Columns = ['Gait_Identification', 'Pitch', 'q1', 'Yaw', 'MotionDeg', 'q2', 'q3']

df = df[Columns]

df
```

```
        Gait_Identification      Pitch      q1        Yaw  MotionDeg      q2  \
0                         0  155.24887 -0.7689  139.96880   0.037984 -0.2633
1                         0  155.16757 -0.7689  139.89722   0.076017 -0.2636
2                         0  155.08208 -0.7690  139.80590   0.117990 -0.2638
3                         0  155.01852 -0.7692  139.70930   0.159124 -0.2639
4                         0  154.92510 -0.7693  139.61806   0.188294 -0.2640
...                     ...        ...     ...        ...        ...     ...
147152                   29  157.51231 -0.3255  181.17784   3.923514 -0.6589
```

```
147153                  29  157.54697 -0.3255  181.18188   3.924339 -0.6588
147154                  29  157.56210 -0.3255  181.17296   3.917008 -0.6589
147155                  29  157.59895 -0.3255  181.16030   3.911465 -0.6589
147156                  29  157.61192 -0.3255  181.16806   3.911900 -0.6589

            q3
0       0.0997
1       0.0996
2       0.0994
3       0.0992
4       0.0989
...        ...
147152  0.5425
147153  0.5426
147154  0.5427
147155  0.5428
147156  0.5429

[147157 rows x 7 columns]
```



```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler

# Assuming df is your DataFrame
sequence_length = 90
overlap_percentage = 0.5

# Extract the relevant columns for X (features) and y (labels)
X_columns = Feats
y_column = Label

# Separate features and labels
X = df[X_columns].values
```

```
# Scale the features using StandardScaler
scaler = StandardScaler()
df[X_columns] = scaler.fit_transform(X)


# Calculate the overlap and step size
overlap_size = int(sequence_length * overlap_percentage)
step_size = sequence_length - overlap_size

# Initialize lists to store sequences and labels
sequences = []
labels = []

# Iterate through the dataframe to create sequences
for i in range(0, len(df) - sequence_length + 1, step_size):
    sequence = df[X_columns].values[i:i + sequence_length]
    label_values = tuple(tuple(row) for row in df[y_column].values[i:i + sequence_length])  # Convert nested arrays to tuples

    # Check if all label values in the sequence are the same
    if len(set(label_values)) == 1:
        label = label_values[0]
        sequences.append(sequence)
        labels.append(label)

# Convert lists to numpy arrays
X = np.array(sequences)
y = np.array(labels)
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is ...
   required for this version of SciPy (detected version 1.24.3
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

```
print(X.shape)

print(y.shape)
```

```
(3154, 90, 6)
(3154, 1)
```

Running combined k-fold cross-validation and F1 scoring for all fold. End result is an average F1 score for all folds, this robust testing scheme should give a realistic validation of model performance.

The custom layer in the model is causing some trouble when saving, and loading the model.

This was solved by first building the model from scratch, and then loading the saved file from the training session.

```
from keras.layers import Multiply
from keras.models import Sequential
from keras.layers import Conv1D, Dense, Activation, Input, Reshape, Lambda, concatenate, Layer
from keras.layers import Add, Dropout, BatchNormalization, GlobalAveragePooling1D
from keras.optimizers import Adam

class ResidualBlock(Layer):
    def __init__(self, dilation_rate, nb_filters, kernel_size, padding, dropout_rate, **kwargs):
        super(ResidualBlock, self).__init__(**kwargs)
        self.dilation_rate = dilation_rate
        self.nb_filters = nb_filters
        self.kernel_size = kernel_size
        self.padding = padding
        self.dropout_rate = dropout_rate

    def build(self, input_shape):
        super(ResidualBlock, self).build(input_shape)
        self.tanh_conv = Conv1D(filters=self.nb_filters, kernel_size=self.kernel_size, dilation_rate=self.dilation_rate,
                                padding=self.padding, activation='tanh')
        self.sigm_conv = Conv1D(filters=self.nb_filters, kernel_size=self.kernel_size, dilation_rate=self.dilation_rate,
                                padding=self.padding, activation='sigmoid')
        self.multiply = Multiply()
        self.one_by_one_conv = Conv1D(filters=self.nb_filters, kernel_size=1, padding='same')
        self.add_layer = Add()
        self.activation = Activation('relu')
```

```
    def call(self, x):
        prev_x = x
        tanh_out = self.tanh_conv(x)
        sigm_out = self.sigm_conv(x)
        x = self.multiply([tanh_out, sigm_out])
        x = self.one_by_one_conv(x)
        res_x = self.add_layer([prev_x, x])
        activation = self.activation(res_x)
        return activation


    # For saving model in Keras:
    def get_config(self):
        config = super(ResidualBlock, self).get_config()
        config.update({
            'dilation_rate': self.dilation_rate,
            'nb_filters': self.nb_filters,
            'kernel_size': self.kernel_size,
            'padding': self.padding,
            'dropout_rate': self.dropout_rate
        })
        return config

def build_tcn_model(input_shape, nb_filters, kernel_size, dilations, nb_stacks, dropout_rate=0.0, output_dim=30):
    model = Sequential()
    model.add(Conv1D(filters=nb_filters, kernel_size=1, padding='same', input_shape=input_shape))
    for _ in range(nb_stacks):
        for dilation_rate in dilations:
            model.add(ResidualBlock(dilation_rate=dilation_rate,
                                    nb_filters=nb_filters, kernel_size=kernel_size,
                                    padding='causal', dropout_rate=dropout_rate))
    model.add(GlobalAveragePooling1D())
    model.add(Dense(units=output_dim, activation='softmax'))
    model.compile(optimizer=Adam(lr=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
    return model

input_dim = X.shape[2]
input_shape = (sequence_length, input_dim)
output_dim = 30
nb_filters = 64
kernel_size = 3
dilations = [1, 2, 4, 8]
nb_stacks = 3
dropout_rate = 0.2

TCN_Model = build_tcn_model(input_shape, nb_filters, kernel_size, dilations, nb_stacks, dropout_rate, output_dim)
TCN_Model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 90, 64)            448

 residual_block (ResidualBl  (None, 90, 64)            28864
 ock)

 residual_block_1 (Residual  (None, 90, 64)            28864
 Block)

 residual_block_2 (Residual  (None, 90, 64)            28864
 Block)

 residual_block_3 (Residual  (None, 90, 64)            28864
 Block)

 residual_block_4 (Residual  (None, 90, 64)            28864
 Block)
```

```
 residual_block_5 (Residual  (None, 90, 64)           28864
 Block)

 residual_block_6 (Residual  (None, 90, 64)           28864
 Block)

 residual_block_7 (Residual  (None, 90, 64)           28864
 Block)

 residual_block_8 (Residual  (None, 90, 64)           28864
 Block)

 residual_block_9 (Residual  (None, 90, 64)           28864
 Block)

 residual_block_10 (Residua  (None, 90, 64)           28864
 lBlock)

 residual_block_11 (Residua  (None, 90, 64)           28864
 lBlock)

 global_average_pooling1d (  (None, 64)               0
 GlobalAveragePooling1D)

 dense (Dense)               (None, 30)               1950

=================================================================
Total params: 348766 (1.33 MB)
Trainable params: 348766 (1.33 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```python
from keras.models import load_model

# Define the custom object
```

```
custom_objects = {'ResidualBlock': ResidualBlock}

# Load the model
TCN_Model = load_model('/kaggle/input/topman-models-for-eval/TCN_Model_TopMan_ID.h5', custom_objects=custom_objects)

from keras.models import Sequential
from keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Dropout, BatchNormalization
from keras.optimizers import RMSprop
from keras.regularizers import l2
from keras.callbacks import EarlyStopping

#Variables:

input_dim = X.shape[2]
sequence_length = X.shape[1]
input_shape = (sequence_length, input_dim)
output_dim = len(np.unique(y))
nb_filters = 64
kernel_size = 3
dilations = [1, 2, 3, 6]
nb_stacks = 3
dropout_rate = 0.2

#model-specific variables:

#Dense layer, to exand num combinations
Units_Dens_Expand_Dim = 80

Units_Unity_Connected_Layer = 64

#Unity_Conected_Layer = Conv

#Conv layer, create new abstractions from features
Filters = 100
```

```
Kernel_Size = 2


# Pooling
Pool_Size = 2


#LSTM, learn patterns in data
LSTM_Units = 64

Recurrent_Dropout = 0.2




# dropout, prevent overfitting
DropOut_Rate = 0.4
```



```
from keras.layers import Input, Conv1D, LSTM, Concatenate, Flatten, Layer
from tensorflow.keras.models import Model
import tensorflow as tf

class MyAttention(Layer):
    def __init__(self, **kwargs):
        super(MyAttention, self).__init__(**kwargs)

    def build(self, input_shape):
        sequence_dim = input_shape[1] if input_shape[1] is not None else 1
        feature_dim = input_shape[-1] if input_shape[-1] is not None else 1
        initializer = tf.keras.initializers.GlorotUniform(seed=None)  # You can choose a different initializer if needed
        self.kernel = self.add_weight(
            name='kernel',
            shape=(feature_dim, 1),
            initializer=initializer,
            trainable=True
```

```
        )
        super(MyAttention, self).build(input_shape)

    def call(self, x):
        scores = tf.matmul(x, self.kernel)
        attention_weights = tf.nn.softmax(scores, axis=1)
        attended_output = x * attention_weights
        return attended_output


# Input layer
In_Layer = Input(shape=input_shape)

# Convolutional layer

C_Layer = Conv1D(data_format= 'channels_last', padding= 'same', filters = Filters, kernel_size = Kernel_Size)(In_Layer)

C_Layer_2 = Conv1D(data_format= 'channels_last', padding= 'same', filters = Filters*2, kernel_size = Kernel_Size)(C_Layer)

C_Layer_3 = Conv1D(data_format= 'channels_last', padding= 'same', filters = Filters*2, kernel_size = Kernel_Size)(C_Layer_2)

Max_Pool = MaxPooling1D(data_format='channels_first',
                            padding= 'same',
                            pool_size=4)(C_Layer_3)

dropout_layer = Dropout(rate=DropOut_Rate)(Max_Pool)

Unity_Connected_Layer = Conv1D(filters=Units_Unity_Connected_Layer,
                             kernel_size=1,
                             use_bias=False,
                            data_format='channels_last')(dropout_layer)

# LSTM layer
lstm_layer_ = LSTM(units=LSTM_Units, return_sequences=True, activation='tanh',
```

```
                    recurrent_activation='sigmoid', recurrent_dropout=Recurrent_Dropout,
                    unroll=False, use_bias=True, return_state=False)(Unity_Connected_Layer)
'''
lstm_layer_1 = LSTM(units=LSTM_Units, return_sequences=True, activation='tanh',
                    recurrent_activation='sigmoid', recurrent_dropout=Recurrent_Dropout,
                    unroll=False, use_bias=True, return_state=False)(lstm_layer_)
'''

attention_layer = MyAttention()

attended_output = attention_layer(lstm_layer_)


# Dropout layer
dropout_layer_2 = Dropout(rate=DropOut_Rate)(attended_output)


# LSTM layer
lstm_layer_1 = LSTM(units=LSTM_Units, return_sequences=True, activation='tanh',
                    recurrent_activation='sigmoid', recurrent_dropout=Recurrent_Dropout,
                    unroll=False, use_bias=True, return_state=False)(dropout_layer_2)
'''
lstm_layer_1 = LSTM(units=LSTM_Units, return_sequences=True, activation='tanh',
                    recurrent_activation='sigmoid', recurrent_dropout=Recurrent_Dropout,
                    unroll=False, use_bias=True, return_state=False)(lstm_layer_)
'''

attention_layer_1 = MyAttention()

attended_output_1 = attention_layer_1(lstm_layer_1)


# Dropout layer
dropout_layer_2 = Dropout(rate=DropOut_Rate)(attended_output_1)
```

```
# Flatten layer
flat_layer = Flatten()(dropout_layer_2)

# Output layer
OUT = Dense(output_dim, activation='softmax')(flat_layer)



# Build the Attentive_Conv_LSTM
Attentive_Conv_LSTM_model = Model(inputs=In_Layer, outputs=OUT)



# Compile the Attentive_Conv_LSTM
Attentive_Conv_LSTM_model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
    from keras.models import load_model

# Define the custom object
custom_objects = {'MyAttention': MyAttention}

# Load the model
#TCN_Model = load_model('/kaggle/input/h-model-tcn-1/TCN_Model_HH.h5', custom_objects=custom_objects)

A_C_LSTM_Model = load_model('/kaggle/input/topman-models-for-eval/Attentive_Conv_LSTM_model_Topman_ID.h5', ...
    custom_objects=custom_objects)
```

```
CNN_Model = load_model('/kaggle/input/topman-models-for-eval/Strict_CNN_model_Topman_ID.h5')

CNN_Model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 90, 32)            224

 conv1d_1 (Conv1D)           (None, 90, 32)            2080

 max_pooling1d (MaxPooling1  (None, 90, 16)            0
 D)

 batch_normalization (Batch  (None, 90, 16)            64
 Normalization)

 conv1d_2 (Conv1D)           (None, 90, 64)            2112

 max_pooling1d_1 (MaxPoolin  (None, 90, 32)            0
 g1D)

 batch_normalization_1 (Bat  (None, 90, 32)            128
 chNormalization)

 conv1d_3 (Conv1D)           (None, 90, 64)            4160

 max_pooling1d_2 (MaxPoolin  (None, 90, 32)            0
 g1D)

 batch_normalization_2 (Bat  (None, 90, 32)            128
 chNormalization)

 conv1d_4 (Conv1D)           (None, 90, 64)            4160

 max_pooling1d_3 (MaxPoolin  (None, 90, 32)            0
 g1D)
```

```
 batch_normalization_3 (Bat  (None, 90, 32)            128
 chNormalization)

 conv1d_5 (Conv1D)           (None, 90, 64)            4160

 max_pooling1d_4 (MaxPoolin  (None, 90, 32)            0
 g1D)

 batch_normalization_4 (Bat  (None, 90, 32)            128
 chNormalization)

 flatten (Flatten)           (None, 2880)              0

 dense (Dense)               (None, 256)               737536

 dropout (Dropout)           (None, 256)               0

 batch_normalization_5 (Bat  (None, 256)               1024
 chNormalization)

 dense_1 (Dense)             (None, 30)                7710

=================================================================
Total params: 763742 (2.91 MB)
Trainable params: 762942 (2.91 MB)
Non-trainable params: 800 (3.12 KB)
_________________________________________________________________


test_loss, test_accuracy = TCN_Model.evaluate(X, y)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')

test_loss, test_accuracy = A_C_LSTM_Model.evaluate(X, y)
```

```
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')

test_loss, test_accuracy = CNN_Model.evaluate(X, y)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')
```

```
99/99 [==============================] - 3s 7ms/step - loss: 0.0350 - accuracy: 0.9914
Test Loss: 0.03497104346752167, Test Accuracy: 0.9914394617080688
99/99 [==============================] - 6s 48ms/step - loss: 0.5971 - accuracy: 0.8050
Test Loss: 0.5970556139945984, Test Accuracy: 0.8050094842910767
99/99 [==============================] - 1s 4ms/step - loss: 1.7976 - accuracy: 0.8272
Test Loss: 1.7975622415542603, Test Accuracy: 0.8272035717964172
```

All three models loaded with weights from training session.

Load the model eval kit, declaring the model_metrics dictionary:

```
    #The combined Average F1 - K-fold cross validation, Model Validation kit by Martin B Gresli. Made in 2024

### PS, only run once, as you will reset the dictonary holding model metrics if run again.

from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import f1_score
from tensorflow.keras.models import Sequential
from keras.optimizers import Adam
import tensorflow as tf
from keras.models import load_model
from keras.callbacks import ModelCheckpoint


import matplotlib.pyplot as plt
import matplotlib.colors as colors
import numpy as np
import tensorflow as tf
import os

def plot_training_history(training_history, title, filename):

    import matplotlib.pyplot as plt
```

```
    fig, ax1 = plt.subplots()

    # Plot training and validation loss on the left y-axis
    ax1.plot(training_history.history['loss'], 'b--', alpha=0.3, label='Training Loss')
    ax1.plot(training_history.history['val_loss'], 'b-', label='Validation Loss')
    ax1.set_xlabel('Epochs')
    ax1.set_ylabel('Loss', color='b')
    ax1.tick_params(axis='y', labelcolor='b')

    # Create a second y-axis
    ax2 = ax1.twinx()

    # Plot training and validation accuracy on the right y-axis
    ax2.plot(training_history.history['accuracy'], 'r--', alpha=0.3, label='Training Accuracy')
    ax2.plot(training_history.history['val_accuracy'], 'r-', label='Validation Accuracy')
    ax2.set_ylabel('Accuracy', color='r')
    ax2.tick_params(axis='y', labelcolor='r')


    lines, labels = ax1.get_legend_handles_labels()
    lines2, labels2 = ax2.get_legend_handles_labels()
    ax2.legend(lines + lines2, labels + labels2, loc='best')

    # Set the title
    plt.title(f'Training History for {title}')

    # Save the plot as a file
    fig.savefig(f'{filename}.eps', format='eps')
    plt.show()
    plt.close()

    import matplotlib.pyplot as plt
    import numpy as np

def save_f1_scores_grouped_as_eps(model_metrics, filename='f1_scores_grouped_performance.eps'):



    model_names = list(model_metrics.keys())
```

```
n_models = len(model_names)
n_folds = len(model_metrics[model_names[0]]['F1_Scores'])

# Data preparation
f1_scores = np.array([model_metrics[model_name]['F1_Scores'] for model_name in model_names]).T  # Transposed for ...
    grouping

average_f1_scores = [model_metrics[model_name]['Average_F1'] for model_name in model_names]

fig, ax = plt.subplots(figsize=(10 + n_models, 6))  # Dynamic width based on number of models

# Calculate the width of each bar and the positions for the groups
bar_width = 0.6 / n_folds  # The total width for each group is 0.8, leaving some space between groups
indices = np.arange(n_models)

# Plot each fold's F1 scores
for i in range(n_folds):
    ax.bar(indices - 0.4 + i * bar_width, f1_scores[i], bar_width, label=f'Fold {i+1}')

# Plot the average F1 score
ax.bar(indices + 0.4, average_f1_scores, bar_width, label='Average', color='skyblue', edgecolor='black')

# Add model names to the x-axis
ax.set_xticks(indices)
ax.set_xticklabels(model_names, rotation=45)
ax.set_ylim(0.9, 1)



ax.set_xlabel('Model Name')
ax.set_ylabel('F1 Score')
ax.set_title('F1 Scores per Fold and Average F1 Score of Models')


ax.legend(loc='lower right')

# Save the plot as a vector graphic file .eps
plt.tight_layout()
plt.savefig(filename, format='eps')
```

```
    plt.savefig(filename.replace('.eps', '.jpeg'), format='jpeg')
    plt.show()
    plt.close()



def save_f1_scores_table_as_eps(model_metrics, filename='f1_scores_table.eps'):

    '''

    This is a vector graphic file of the results in the form of a table.
    I you prefer to make an actual table, the data is available in the
    dictionary:

    model_metrics = {}

    '''



    # Prepare data for the table
    columns = ['Model Name'] + [f'Fold {i+1}' for i in range(len(next(iter(model_metrics.values()))['F1_Scores']))] + ...
       ['Average F1']
    cell_text = []

    for model_name, metrics in model_metrics.items():
        row = [model_name] + metrics['F1_Scores'] + [metrics['Average_F1']]
        cell_text.append(row)

    column_width = 3
    table_width = column_width * (len(columns))
    table_height = 0.1 * len(cell_text)

    fig, ax = plt.subplots(figsize=(table_width, table_height))
    ax.axis('off')

    the_table = ax.table(cellText=cell_text, colLabels=columns, loc='center', cellLoc='center', colLoc='center')

    # Make the column names bold
```

```
    for (i, j), cell in the_table.get_celld().items():
        if i == 0:
            cell.set_fontsize(12)
            cell.set_text_props(weight='bold')

    fig.tight_layout()

    # Save the plot as a vector graphic file .eps
    plt.savefig(filename, format='eps', bbox_inches='tight')
    plt.savefig(filename.replace('.eps', '.jpeg'), format='jpeg', bbox_inches='tight')
    plt.show()
    plt.close()




def plot_confusion_matrix_heatmap(model, model_name, X, y, title, filename_prop):

    '''

    This is a tweaked CM with a colormap mapped to the rate of correct predictions, but the numbers in
    the cells of the matrix displaying the actual count for predicitons.

    This way you can quickly grasp the distribution for the different classes, as well as
    per-class performance for the model.

    The heatmap is exponential at the very top of the scale, so that any sub-100% performance is
    clearly visible. Likse so:

    [(0, 'lightblue'), (0.5, 'skyblue'), (0.99, 'blue'), (1, 'darkblue')]

    '''



    predict = model.predict(X)
```

```
predictions = np.argmax(predict, axis=1)

output_dim = len(np.unique(y))

confusion_matrix = tf.math.confusion_matrix(labels=y,
                                            predictions=predictions,
                                            num_classes=output_dim)

# Calculate the correct prediction
total_predictions_per_class = np.sum(confusion_matrix, axis=1)  # Sum over rows for total predictions
correct_predictions = np.diag(confusion_matrix)
correct_prediction_rate = correct_predictions / total_predictions_per_class.astype(float)

# Create a matrix to hold the values for coloring the heatmap

heatmap_data = np.full_like(confusion_matrix, np.nan, dtype=float)  # Fill with NaN
np.fill_diagonal(heatmap_data, correct_prediction_rate)  # Set the diagonal with correct prediction rate

# Create a colormap that is light blue for 0 and dark blue for high correct prediction rate
cmap = colors.LinearSegmentedColormap.from_list(
    'custom blue',
    [(0, 'lightblue'), (0.5, 'skyblue'), (0.99, 'blue'), (1, 'darkblue')]
)
cmap.set_bad('lightgrey', 1.0)  # Color for NaN values

# Plotting the heatmap
fig, ax = plt.subplots()
cax = ax.matshow(heatmap_data, cmap=cmap, vmin=0, vmax=1)

# Add colorbar
plt.colorbar(cax)

# Annotate all cells with the actual count
for (i, j), val in np.ndenumerate(confusion_matrix):
    if i == j:  # Diagonal: correct predictions
        text_color = 'white' if correct_prediction_rate[i] > 0.5 else 'black'
    else:  # Off-diagonal: incorrect predictions
        text_color = 'grey'
    ax.text(j, i, f'{val}', ha='center', va='center', color=text_color)
```

```
    plt.xlabel('Predicted Labels')
    plt.ylabel('True Labels')
    plt.title(f'{title}')
    plt.savefig(f'{filename_prop}.eps', format='eps')

    plt.show()

    plt.close()



# plot_training_history(training_history, 'training_history')
# plot_confusion_matrix_heatmap(confusion_matrix, 'confusion_matrix_heatmap')



model_metrics = {}



def validate_and_store_metrics(model_list, X, y, k=5, learning_rate=0.01, epochs=15, batch_size=32):
    # Ensure model_metrics is accessible within this function
    global model_metrics

    for model, model_name, custom_object in model_list:

        # Initialize StratifiedKFold

        skf = StratifiedKFold(n_splits=k, shuffle=True, random_state=55)

        # Initialize a list to store F1 scores for each fold
        f1_scores = []

        j = 0
        for train_index, test_index in skf.split(X, y):
```

```
# Split data
X_train, X_test = X[train_index], X[test_index]
y_train, y_test = y[train_index], y[test_index]

# Compile and fit the model
model.compile(optimizer=Adam(lr=learning_rate), loss='sparse_categorical_crossentropy', metrics=['accuracy'])



'''

Dyneamic save/load model, to ensure the best result from each fold is the version used for F1-scoring


'''



checkpoint_filepath =  os.path.join('/kaggle/working/', f'{model_name}_Fold_{j}_Checkpoint.h5')

best_validation_callback = ModelCheckpoint(
    checkpoint_filepath,
    monitor='val_accuracy',
    verbose=1,
    save_best_only=True,
    mode='max',
    save_format='h5'
)


history = model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, validation_data=(X_test, ...
   y_test), callbacks=[best_validation_callback])


'''

If problems accessing the model file arises some conditions can be added.

Example:
```

```
if os.path.exists(checkpoint_filepath):
    print(f"Checkpoint file created: {checkpoint_filepath}")
    model_loaded = load_model(checkpoint_filepath)
else:
    print(f"Checkpoint file not found at: {checkpoint_filepath}. Saving the last model state manually.")
    model.save(checkpoint_filepath)
    model_loaded = model  # Use the current model state

'''

# Plot training history
plot_training_history(history, f'{model_name}, fold {j}', f'Training_History_fold{j}_{model_name}')



'''

In this line:

model = load_model(checkpoint_filepath, custom_object)

The custom object is allready loaded in the workspace, as this loop
extrapolates all three nested items in each list entry.



'''

model = load_model(checkpoint_filepath, custom_object)

# Evaluate the model
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f'Test Loss {model_name}, fold {j}: {test_loss}, Test Accuracy {model_name}, fold {j}: {test_accuracy}')
y_pred = np.argmax(model.predict(X_test), axis=1)
y_pred_classes = np.round(y_pred).astype(int)
```

```
            # Calculate and store F1 score
            f1 = f1_score(y_test, y_pred_classes, average='macro')
            if model_name not in model_metrics:
                model_metrics[model_name] = {'F1_Scores': [], 'Average_F1': 0}
            model_metrics[model_name]['F1_Scores'].append(f1)

            print(f'F1 score for {model_name}, fold {j}: {f1}')
            f1_scores.append(f1)

            # Plot confusion matrix heatmap
            plot_confusion_matrix_heatmap(model, model_name, X_test, y_test, f'Confusion matrix for {model_name}, fold ...
                {j}', f'Prop_Confusion_Matrix_fold_{j}_{model_name}')

            j += 1


        average_f1_score = np.mean(model_metrics[model_name]['F1_Scores'])
        model_metrics[model_name]['Average_F1'] = average_f1_score

        print(f'RESULTS for {model_name}')

        print(f"Average F1 Score for {model_name}: {average_f1_score}")
        print(f'F1 scores for all folds for {model_name}:', model_metrics[model_name]['F1_Scores'])
        print('') # Make room


    save_f1_scores_table_as_eps(model_metrics, filename='f1_scores_table.eps')
    save_f1_scores_grouped_as_eps(model_metrics, filename='f1_scores_grouped_performance.eps')



'''

Please use this format:


model_list = [

    (Attentive_LSTM_Model, 'A_C_LSTM_Model', {'MyAttention': MyAttention}),
```

```
    (TCN_Model, 'TCN_Model', {'ResidualBlock': ResidualBlock}),
    (CNN_Model, 'CNN_Model', None)

]

The last entry for each line is a dictionary declaring any costum objects used in the models.
If costum objects are used, declaring the entire model before loading the model file might be necessary.
Please enter " None " for all models that stricktly contain objects that are in the libraries loaded in workspace.


validate_and_store_metrics(model_list, X, y, k=5, epochs=50) #Example

'''
```

Run validation for all models, and dump all eps files in zip for report

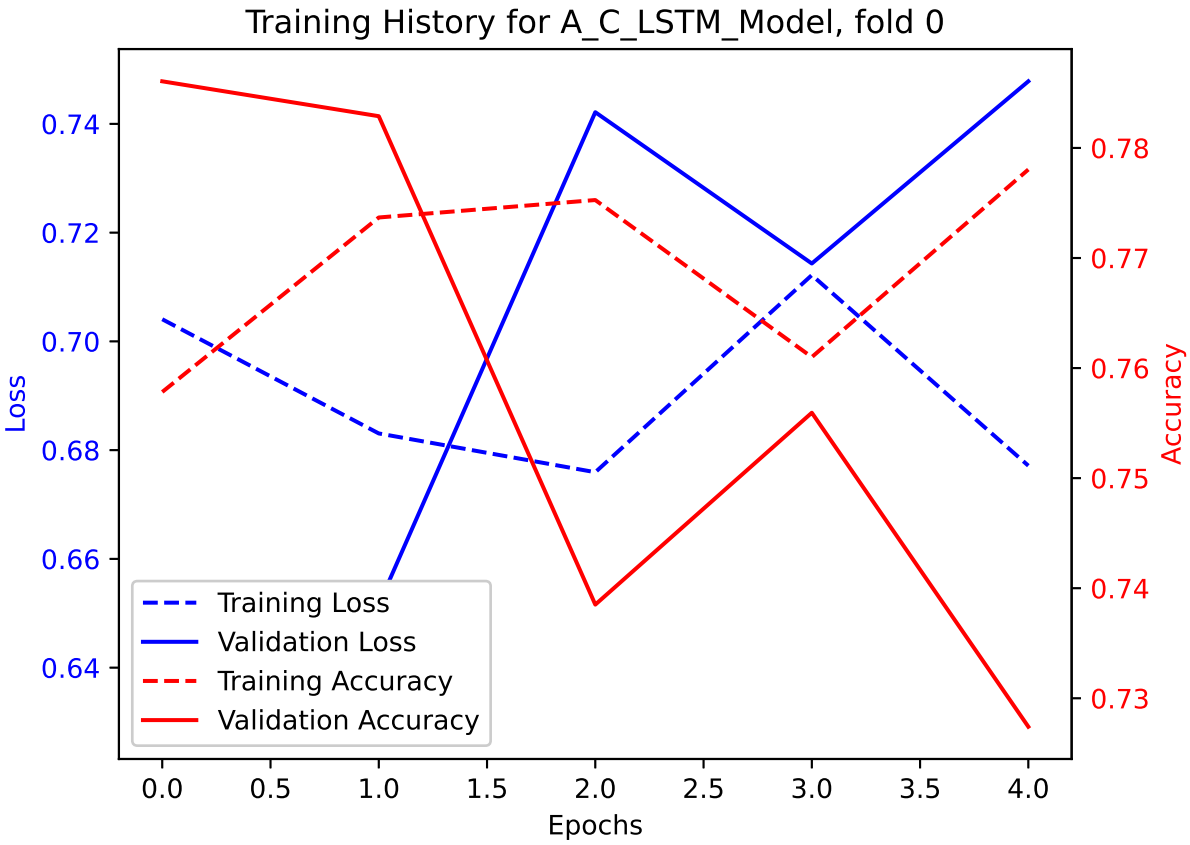
```
model_list = [

    (A_C_LSTM_Model, 'A_C_LSTM_Model', {'MyAttention': MyAttention}),
    (TCN_Model, 'TCN_Model', {'ResidualBlock': ResidualBlock}),
    (CNN_Model, 'CNN_Model', None)

]
'''

model_list = [


    (TCN_Model, 'TCN_Model', {'ResidualBlock': ResidualBlock})


]
'''
```

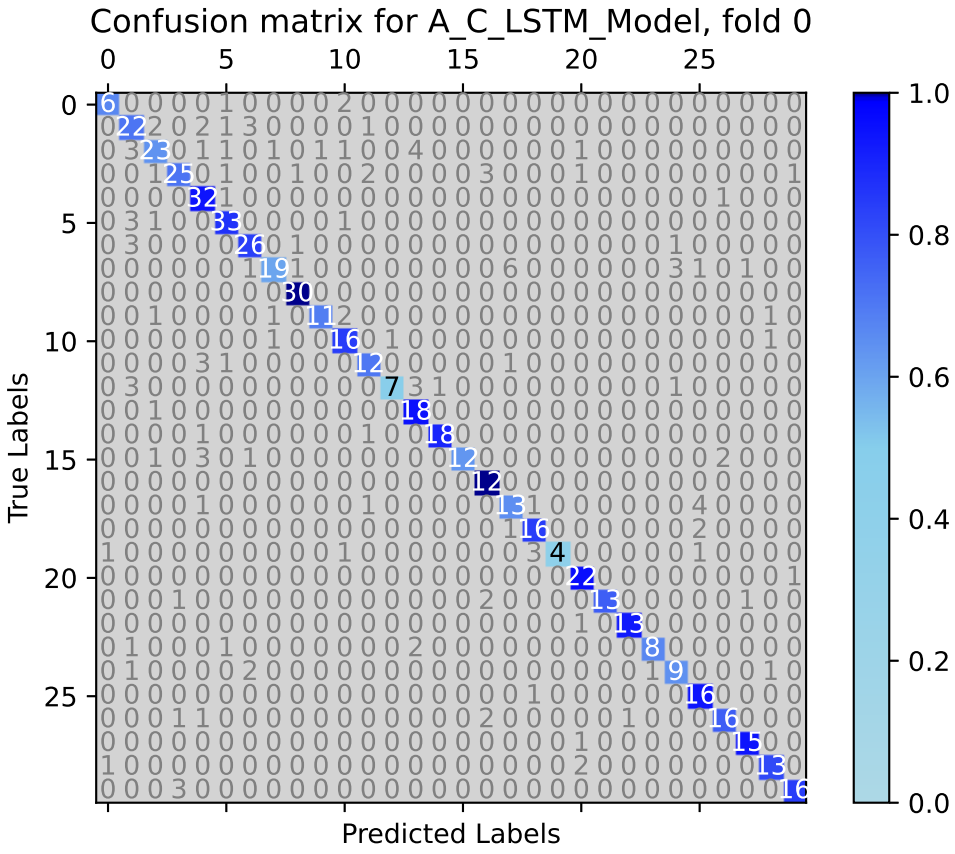
```
validate_and_store_metrics(model_list, X, y, k=5, epochs=5)


!zip -r UIA_IMU_ID_Models_Val.zip /kaggle/working
```

A-C-LSTM model

```
Epoch 1/5
79/79 [==============================] - ETA: 0s - loss: 0.7041 - accuracy: 0.7578
Epoch 1: val_accuracy improved from -inf to 0.78605, saving model to /kaggle/working/A_C_LSTM_Model_Fold_0_Checkpoint.h5
79/79 [==============================] - 51s 535ms/step - loss: 0.7041 - accuracy: 0.7578 - val_loss: 0.6291 - ...
    val_accuracy: 0.7861
Epoch 2/5

79/79 [==============================] - ETA: 0s - loss: 0.6831 - accuracy: 0.7737
Epoch 2: val_accuracy did not improve from 0.78605
79/79 [==============================] - 40s 511ms/step - loss: 0.6831 - accuracy: 0.7737 - val_loss: 0.6516 - ...
    val_accuracy: 0.7829
Epoch 3/5
79/79 [==============================] - ETA: 0s - loss: 0.6759 - accuracy: 0.7753
Epoch 3: val_accuracy did not improve from 0.78605
79/79 [==============================] - 40s 509ms/step - loss: 0.6759 - accuracy: 0.7753 - val_loss: 0.7421 - ...
    val_accuracy: 0.7385
Epoch 4/5
79/79 [==============================] - ETA: 0s - loss: 0.7122 - accuracy: 0.7610
Epoch 4: val_accuracy did not improve from 0.78605
79/79 [==============================] - 40s 507ms/step - loss: 0.7122 - accuracy: 0.7610 - val_loss: 0.7143 - ...
    val_accuracy: 0.7559
Epoch 5/5
79/79 [==============================] - ETA: 0s - loss: 0.6771 - accuracy: 0.7780
Epoch 5: val_accuracy did not improve from 0.78605
79/79 [==============================] - 40s 512ms/step - loss: 0.6771 - accuracy: 0.7780 - val_loss: 0.7478 - ...
    val_accuracy: 0.7274
```

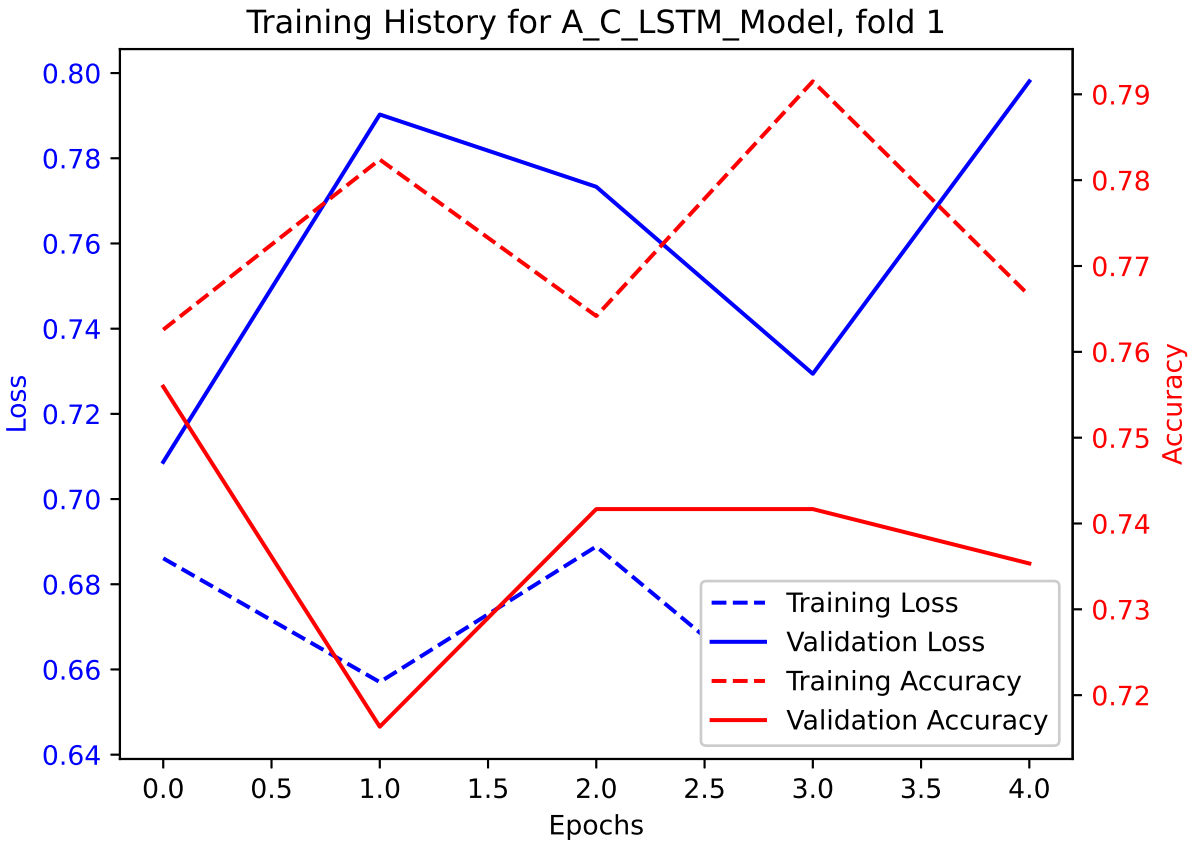
```
20/20 [==============================] - 2s 47ms/step - loss: 0.6291 - accuracy: 0.7861
Test Loss A_C_LSTM_Model, fold 0: 0.6291362643241882, Test Accuracy A_C_LSTM_Model, fold 0: 0.7860538959503174
20/20 [==============================] - 1s 47ms/step
F1 score for A_C_LSTM_Model, fold 0: 0.7771861305935289
20/20 [==============================] - 1s 51ms/step
```

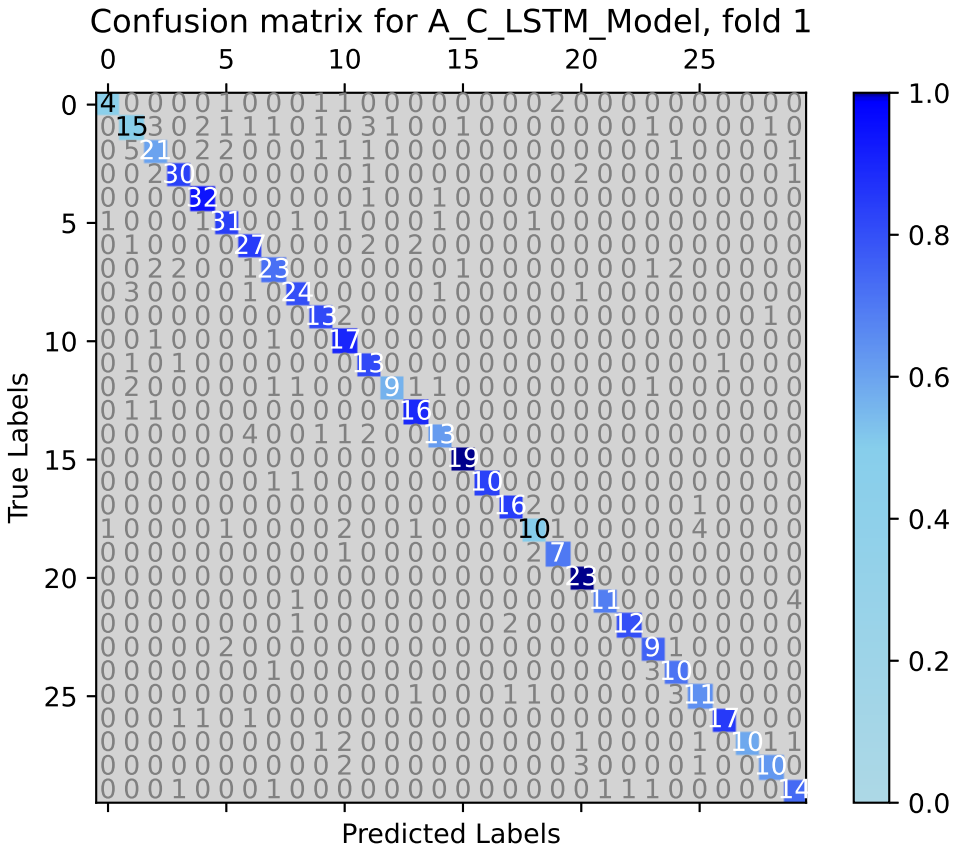

Training History for A_C_LSTM_Model, fold 0
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy


Confusion matrix for A_C_LSTM_Model, fold 0
True Labels
Predicted Labels

```
Epoch 1/5
79/79 [==============================] - ETA: 0s - loss: 0.6861 - accuracy: 0.7626
Epoch 1: val_accuracy improved from -inf to 0.75594, saving model to /kaggle/working/A_C_LSTM_Model_Fold_1_Checkpoint.h5
79/79 [==============================] - 47s 517ms/step - loss: 0.6861 - accuracy: 0.7626 - val_loss: 0.7087 - ...
   val_accuracy: 0.7559
Epoch 2/5
79/79 [==============================] - ETA: 0s - loss: 0.6570 - accuracy: 0.7824
Epoch 2: val_accuracy did not improve from 0.75594
79/79 [==============================] - 40s 505ms/step - loss: 0.6570 - accuracy: 0.7824 - val_loss: 0.7903 - ...
   val_accuracy: 0.7163
Epoch 3/5
79/79 [==============================] - ETA: 0s - loss: 0.6889 - accuracy: 0.7642
Epoch 3: val_accuracy did not improve from 0.75594
79/79 [==============================] - 40s 505ms/step - loss: 0.6889 - accuracy: 0.7642 - val_loss: 0.7733 - ...
   val_accuracy: 0.7417
Epoch 4/5
79/79 [==============================] - ETA: 0s - loss: 0.6466 - accuracy: 0.7915
Epoch 4: val_accuracy did not improve from 0.75594
79/79 [==============================] - 40s 503ms/step - loss: 0.6466 - accuracy: 0.7915 - val_loss: 0.7294 - ...
   val_accuracy: 0.7417
Epoch 5/5
79/79 [==============================] - ETA: 0s - loss: 0.6799 - accuracy: 0.7665
Epoch 5: val_accuracy did not improve from 0.75594
79/79 [==============================] - 40s 502ms/step - loss: 0.6799 - accuracy: 0.7665 - val_loss: 0.7981 - ...
   val_accuracy: 0.7353
```

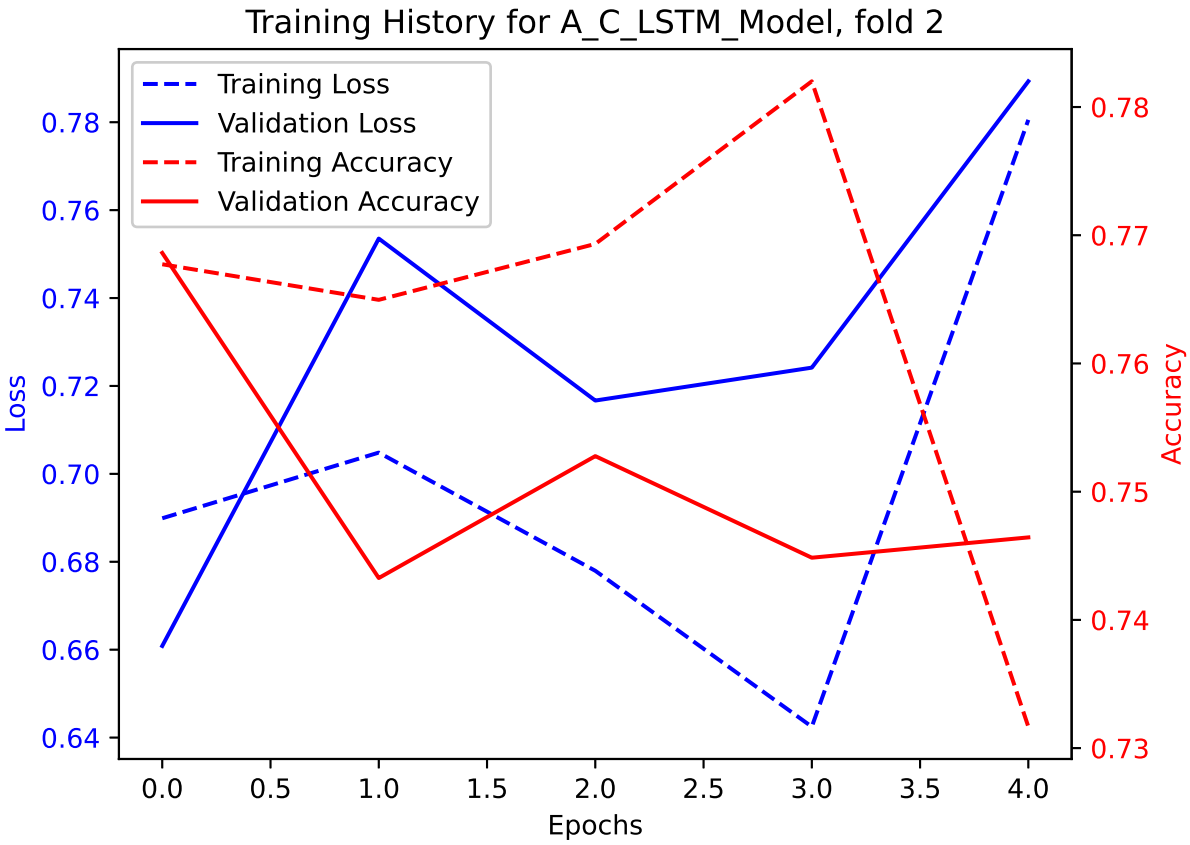
```
20/20 [==============================] - 1s 45ms/step - loss: 0.7087 - accuracy: 0.7559
Test Loss A_C_LSTM_Model, fold 1: 0.7087114453315735, Test Accuracy A_C_LSTM_Model, fold 1: 0.7559429407119751
20/20 [==============================] - 1s 45ms/step
F1 score for A_C_LSTM_Model, fold 1: 0.7451250906981052
20/20 [==============================] - 1s 45ms/step
```

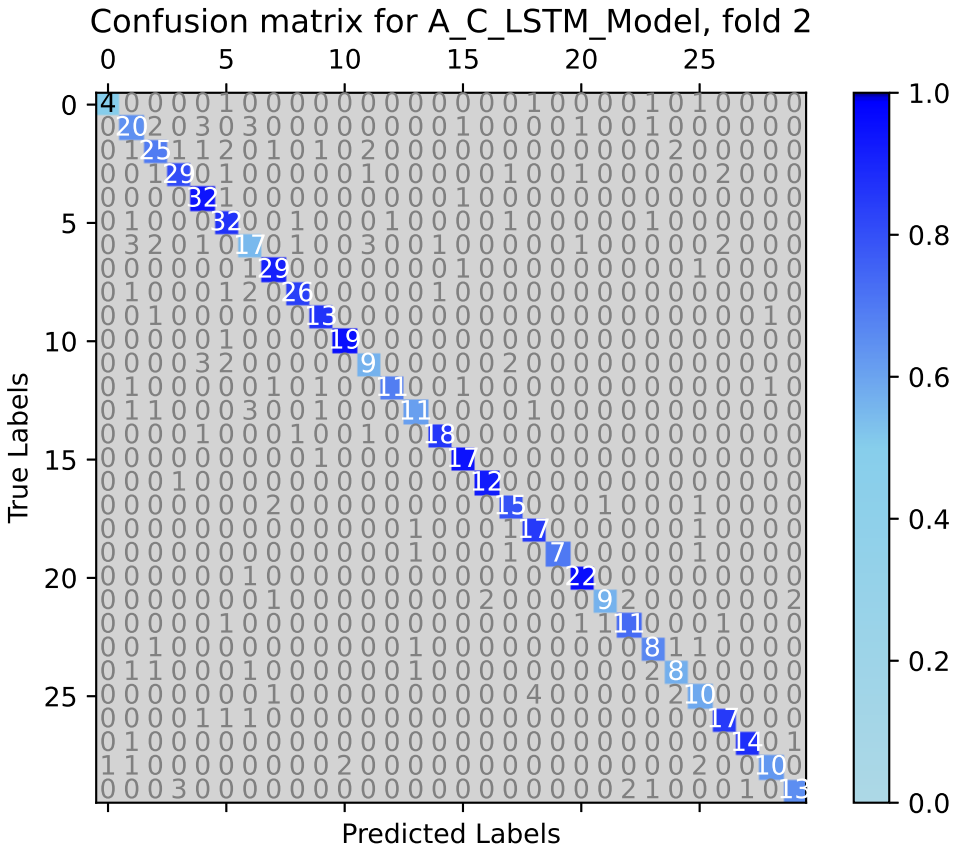

Training History for A_C_LSTM_Model, fold 1
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy


Confusion matrix for A_C_LSTM_Model, fold 1
True Labels
Predicted Labels

```
Epoch 1/5
79/79 [==============================] - ETA: 0s - loss: 0.6899 - accuracy: 0.7677
Epoch 1: val_accuracy improved from -inf to 0.76862, saving model to /kaggle/working/A_C_LSTM_Model_Fold_2_Checkpoint.h5
79/79 [==============================] - 48s 522ms/step - loss: 0.6899 - accuracy: 0.7677 - val_loss: 0.6608 - ...
   val_accuracy: 0.7686
Epoch 2/5
79/79 [==============================] - ETA: 0s - loss: 0.7048 - accuracy: 0.7650
Epoch 2: val_accuracy did not improve from 0.76862
79/79 [==============================] - 40s 508ms/step - loss: 0.7048 - accuracy: 0.7650 - val_loss: 0.7535 - ...
   val_accuracy: 0.7433
Epoch 3/5
79/79 [==============================] - ETA: 0s - loss: 0.6779 - accuracy: 0.7693
Epoch 3: val_accuracy did not improve from 0.76862
79/79 [==============================] - 40s 504ms/step - loss: 0.6779 - accuracy: 0.7693 - val_loss: 0.7167 - ...
   val_accuracy: 0.7528
Epoch 4/5
79/79 [==============================] - ETA: 0s - loss: 0.6425 - accuracy: 0.7820
Epoch 4: val_accuracy did not improve from 0.76862
79/79 [==============================] - 40s 509ms/step - loss: 0.6425 - accuracy: 0.7820 - val_loss: 0.7242 - ...
   val_accuracy: 0.7448
Epoch 5/5
79/79 [==============================] - ETA: 0s - loss: 0.7805 - accuracy: 0.7317
Epoch 5: val_accuracy did not improve from 0.76862
79/79 [==============================] - 40s 511ms/step - loss: 0.7805 - accuracy: 0.7317 - val_loss: 0.7893 - ...
   val_accuracy: 0.7464
```

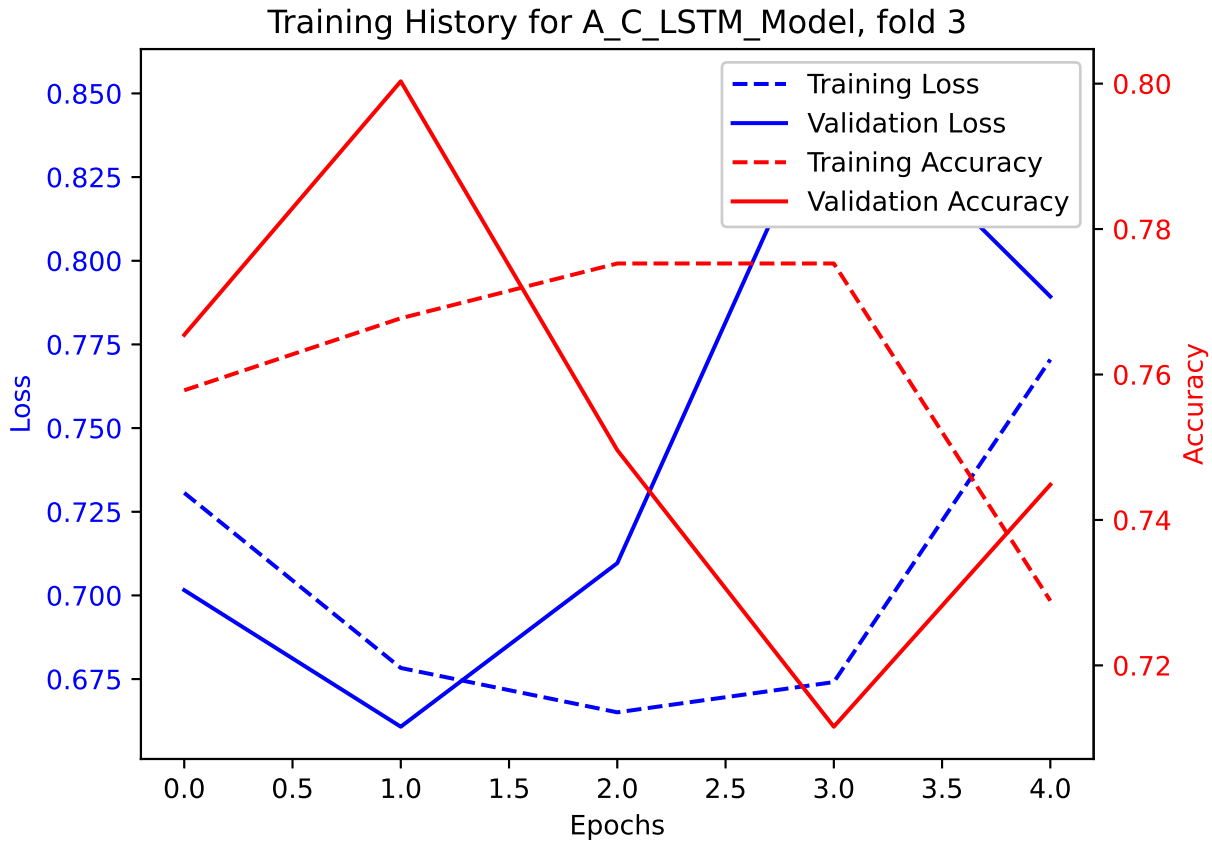
```
20/20 [==============================] - 2s 48ms/step - loss: 0.6608 - accuracy: 0.7686
Test Loss A_C_LSTM_Model, fold 2: 0.6608448624610901, Test Accuracy A_C_LSTM_Model, fold 2: 0.7686212658882141
20/20 [==============================] - 1s 46ms/step
F1 score for A_C_LSTM_Model, fold 2: 0.7576101883479243
20/20 [==============================] - 1s 45ms/step
```

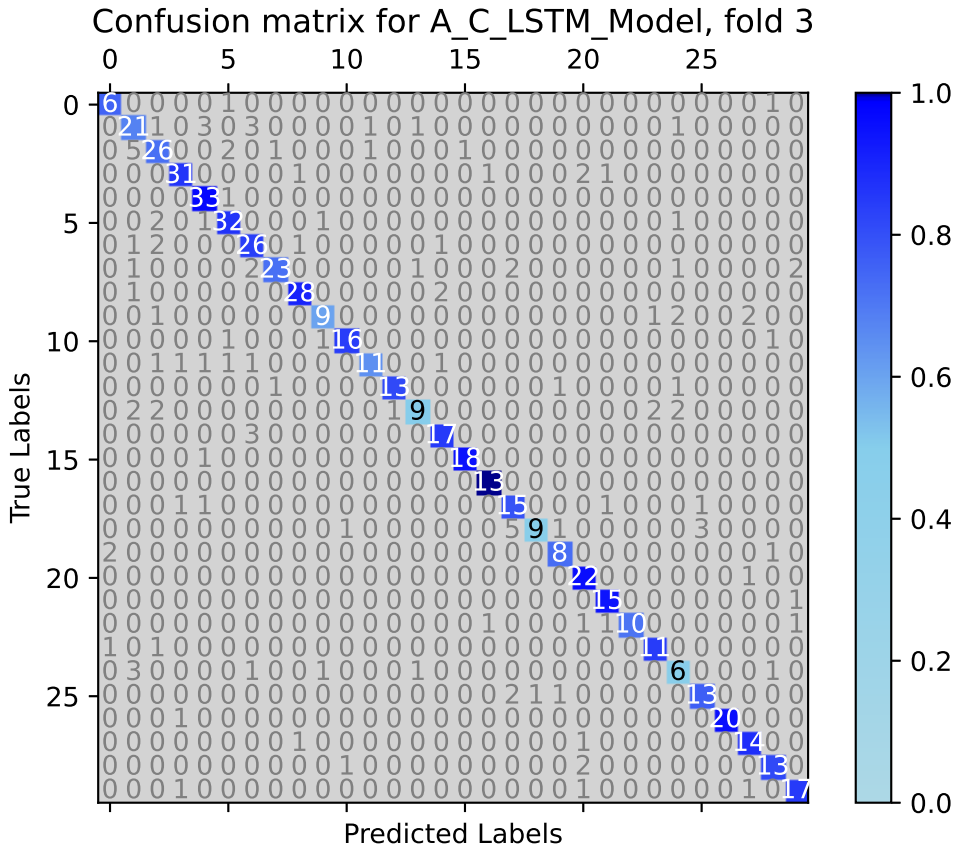

Training History for A_C_LSTM_Model, fold 2
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
Loss
Accuracy
Epochs
0.78
0.76
0.74
0.72
0.70
0.68
0.66
0.64
0.77
0.75
0.73
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Confusion matrix for A_C_LSTM_Model, fold 2
True Labels
Predicted Labels
0
5
10
15
20
25
1.0
0.8
0.6
0.4
0.2
0.0

```
Epoch 1/5
79/79 [==============================] - ETA: 0s - loss: 0.7307 - accuracy: 0.7578
Epoch 1: val_accuracy improved from -inf to 0.76545, saving model to /kaggle/working/A_C_LSTM_Model_Fold_3_Checkpoint.h5
79/79 [==============================] - 47s 514ms/step - loss: 0.7307 - accuracy: 0.7578 - val_loss: 0.7016 - ...
   val_accuracy: 0.7655
Epoch 2/5
79/79 [==============================] - ETA: 0s - loss: 0.6783 - accuracy: 0.7677
Epoch 2: val_accuracy improved from 0.76545 to 0.80032, saving model to /kaggle/working/A_C_LSTM_Model_Fold_3_Checkpoint.h5
79/79 [==============================] - 40s 507ms/step - loss: 0.6783 - accuracy: 0.7677 - val_loss: 0.6608 - ...
   val_accuracy: 0.8003
Epoch 3/5
79/79 [==============================] - ETA: 0s - loss: 0.6651 - accuracy: 0.7753
Epoch 3: val_accuracy did not improve from 0.80032
79/79 [==============================] - 40s 505ms/step - loss: 0.6651 - accuracy: 0.7753 - val_loss: 0.7096 - ...
   val_accuracy: 0.7496
Epoch 4/5
79/79 [==============================] - ETA: 0s - loss: 0.6741 - accuracy: 0.7753
Epoch 4: val_accuracy did not improve from 0.80032
79/79 [==============================] - 40s 508ms/step - loss: 0.6741 - accuracy: 0.7753 - val_loss: 0.8536 - ...
   val_accuracy: 0.7116
Epoch 5/5
79/79 [==============================] - ETA: 0s - loss: 0.7705 - accuracy: 0.7289
Epoch 5: val_accuracy did not improve from 0.80032
79/79 [==============================] - 40s 509ms/step - loss: 0.7705 - accuracy: 0.7289 - val_loss: 0.7893 - ...
   val_accuracy: 0.7448
```

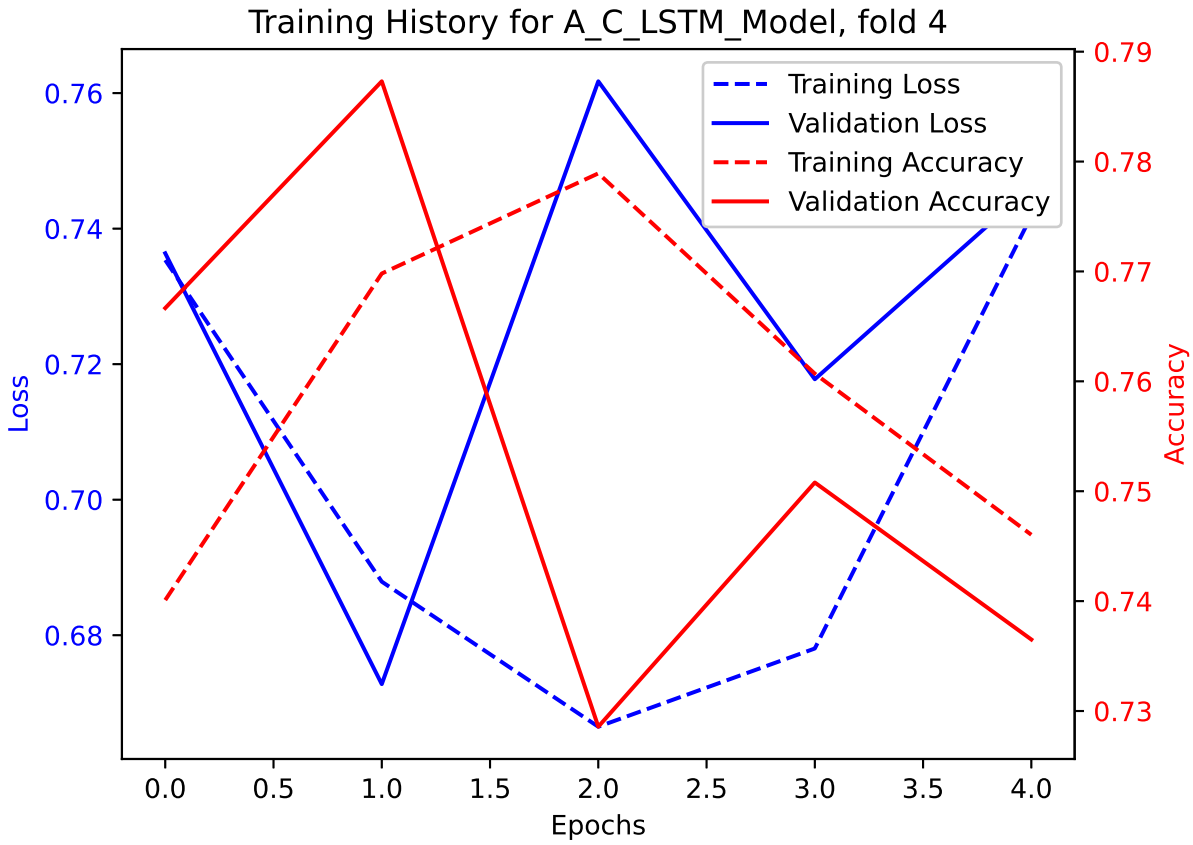
```
20/20 [==============================] - 1s 46ms/step - loss: 0.6608 - accuracy: 0.8003
Test Loss A_C_LSTM_Model, fold 3: 0.6607687473297119, Test Accuracy A_C_LSTM_Model, fold 3: 0.8003169298171997
20/20 [==============================] - 1s 45ms/step
F1 score for A_C_LSTM_Model, fold 3: 0.7887851274015556
20/20 [==============================] - 1s 44ms/step
```

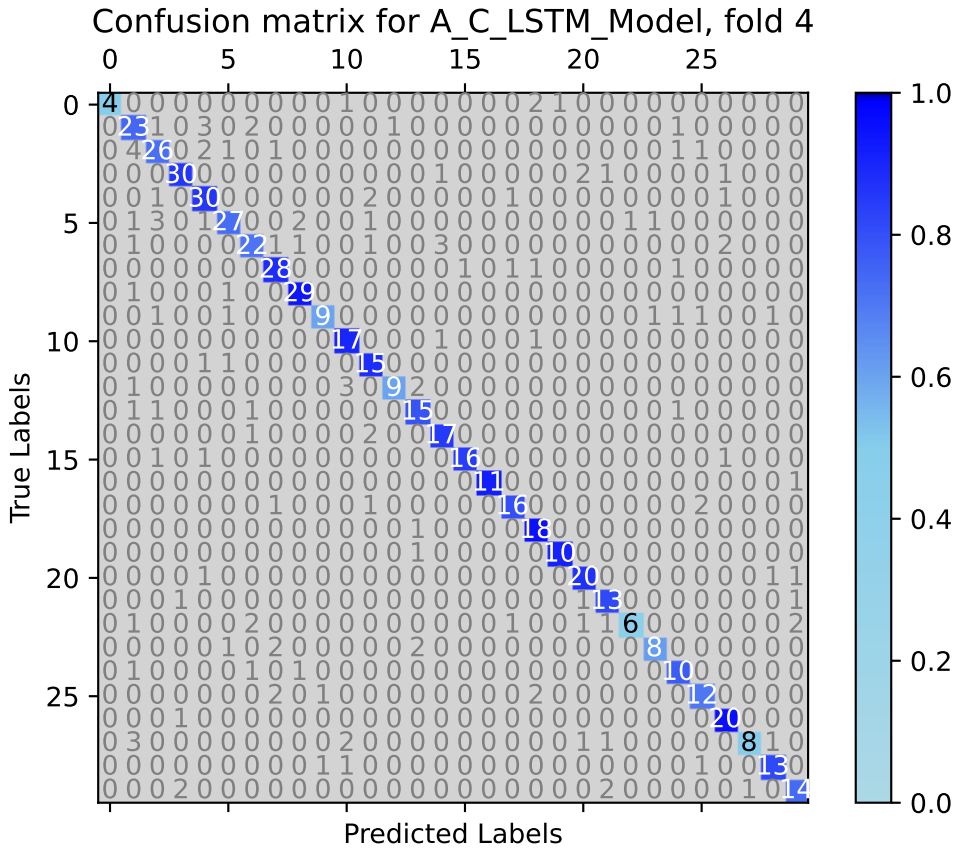

Training History for A_C_LSTM_Model, fold 3
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
Loss
Accuracy
Epochs
0.850
0.825
0.800
0.775
0.750
0.725
0.700
0.675
0.80
0.78
0.76
0.74
0.72
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Confusion matrix for A_C_LSTM_Model, fold 3
True Labels
Predicted Labels
0
5
10
15
20
25
1.0
0.8
0.6
0.4
0.2
0.0

```
Epoch 1/5
79/79 [==============================] - ETA: 0s - loss: 0.7354 - accuracy: 0.7401
Epoch 1: val_accuracy improved from -inf to 0.76667, saving model to /kaggle/working/A_C_LSTM_Model_Fold_4_Checkpoint.h5
79/79 [==============================] - 49s 533ms/step - loss: 0.7354 - accuracy: 0.7401 - val_loss: 0.7364 - ...
   val_accuracy: 0.7667
Epoch 2/5
79/79 [==============================] - ETA: 0s - loss: 0.6879 - accuracy: 0.7698
Epoch 2: val_accuracy improved from 0.76667 to 0.78730, saving model to /kaggle/working/A_C_LSTM_Model_Fold_4_Checkpoint.h5
79/79 [==============================] - 40s 509ms/step - loss: 0.6879 - accuracy: 0.7698 - val_loss: 0.6728 - ...
   val_accuracy: 0.7873
Epoch 3/5
79/79 [==============================] - ETA: 0s - loss: 0.6665 - accuracy: 0.7789
Epoch 3: val_accuracy did not improve from 0.78730
79/79 [==============================] - 40s 508ms/step - loss: 0.6665 - accuracy: 0.7789 - val_loss: 0.7617 - ...
   val_accuracy: 0.7286
Epoch 4/5
79/79 [==============================] - ETA: 0s - loss: 0.6780 - accuracy: 0.7607
Epoch 4: val_accuracy did not improve from 0.78730
79/79 [==============================] - 40s 506ms/step - loss: 0.6780 - accuracy: 0.7607 - val_loss: 0.7178 - ...
   val_accuracy: 0.7508
Epoch 5/5
79/79 [==============================] - ETA: 0s - loss: 0.7419 - accuracy: 0.7460
Epoch 5: val_accuracy did not improve from 0.78730
79/79 [==============================] - 41s 513ms/step - loss: 0.7419 - accuracy: 0.7460 - val_loss: 0.7462 - ...
   val_accuracy: 0.7365
```

```
20/20 [==============================] - 2s 47ms/step - loss: 0.6728 - accuracy: 0.7873
Test Loss A_C_LSTM_Model, fold 4: 0.6727830171585083, Test Accuracy A_C_LSTM_Model, fold 4: 0.7873015999794006
20/20 [==============================] - 1s 45ms/step
F1 score for A_C_LSTM_Model, fold 4: 0.777085410226929
20/20 [==============================] - 1s 45ms/step
```


Training History for A_C_LSTM_Model, fold 4
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
Loss
Accuracy
Epochs
0.76
0.74
0.72
0.70
0.68
0.79
0.78
0.77
0.76
0.75
0.74
0.73
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Confusion matrix for A_C_LSTM_Model, fold 4
True Labels
Predicted Labels
0
5
10
15
20
25
1.0
0.8
0.6
0.4
0.2
0.0

RESULTS for A-C-LSTM model

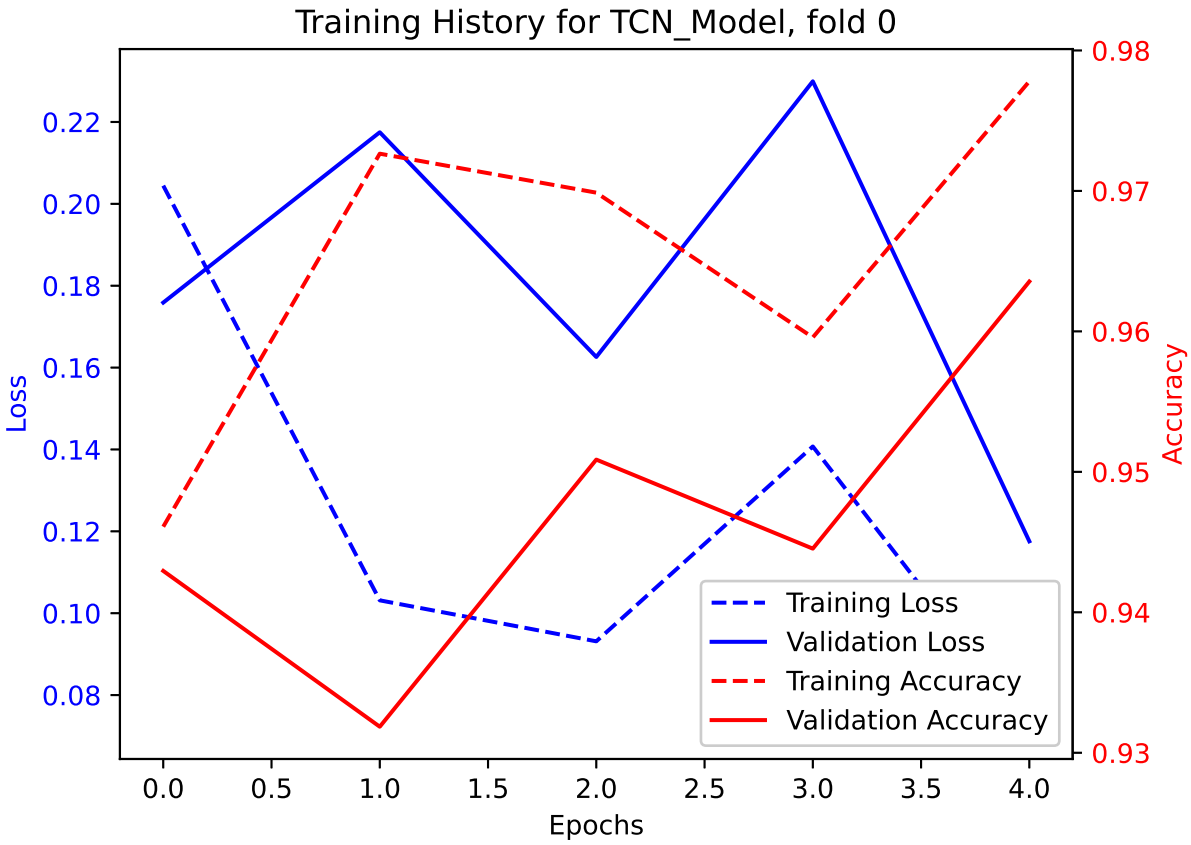
```
RESULTS for A_C_LSTM_Model
Average F1 Score for A_C_LSTM_Model: 0.7691583894536087
F1 scores for all folds for A_C_LSTM_Model: [0.7771861305935289, 0.7451250906981052, 0.7576101883479243, 0.7887851274015556, 0.77708541022
```

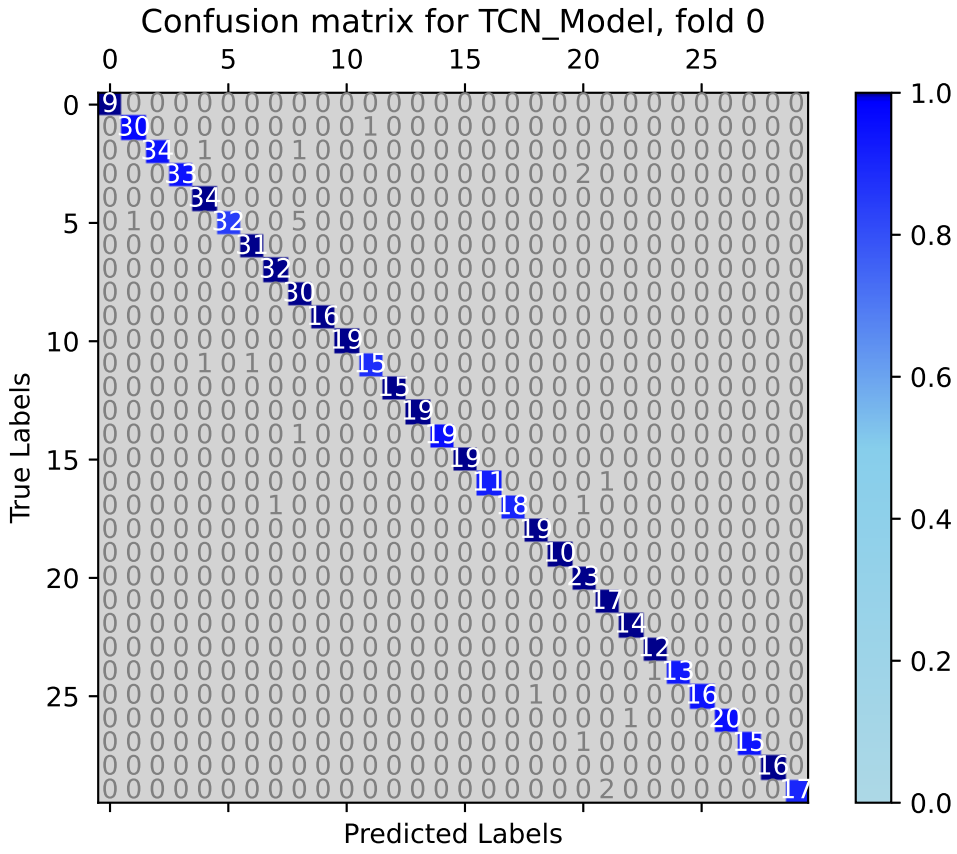
TCN model

```
Epoch 1/5
79/79 [==============================] - ETA: 0s - loss: 0.2045 - accuracy: 0.9461
Epoch 1: val_accuracy improved from -inf to 0.94295, saving model to /kaggle/working/TCN_Model_Fold_0_Checkpoint.h5
79/79 [==============================] - 16s 37ms/step - loss: 0.2045 - accuracy: 0.9461 - val_loss: 0.1759 - ...
   val_accuracy: 0.9429
Epoch 2/5
 1/79 [..............................] - ETA: 1s - loss: 0.1138 - accuracy: 0.9375
77/79 [============================>.] - ETA: 0s - loss: 0.1043 - accuracy: 0.9728
Epoch 2: val_accuracy did not improve from 0.94295
79/79 [==============================] - 1s 18ms/step - loss: 0.1031 - accuracy: 0.9727 - val_loss: 0.2175 - ...
   val_accuracy: 0.9319
Epoch 3/5
78/79 [============================>.] - ETA: 0s - loss: 0.0936 - accuracy: 0.9700
Epoch 3: val_accuracy improved from 0.94295 to 0.95087, saving model to /kaggle/working/TCN_Model_Fold_0_Checkpoint.h5
79/79 [==============================] - 2s 20ms/step - loss: 0.0931 - accuracy: 0.9699 - val_loss: 0.1626 - ...
   val_accuracy: 0.9509
Epoch 4/5
78/79 [============================>.] - ETA: 0s - loss: 0.1414 - accuracy: 0.9595
Epoch 4: val_accuracy did not improve from 0.95087
79/79 [==============================] - 1s 18ms/step - loss: 0.1407 - accuracy: 0.9596 - val_loss: 0.2299 - ...
   val_accuracy: 0.9445
Epoch 5/5
79/79 [==============================] - ETA: 0s - loss: 0.0723 - accuracy: 0.9778
Epoch 5: val_accuracy improved from 0.95087 to 0.96355, saving model to /kaggle/working/TCN_Model_Fold_0_Checkpoint.h5
79/79 [==============================] - 2s 21ms/step - loss: 0.0723 - accuracy: 0.9778 - val_loss: 0.1176 - ...
   val_accuracy: 0.9635
```

```
20/20 [==============================] - 1s 8ms/step - loss: 0.1176 - accuracy: 0.9635
Test Loss TCN_Model, fold 0: 0.11756225675344467, Test Accuracy TCN_Model, fold 0: 0.9635499119758606
```

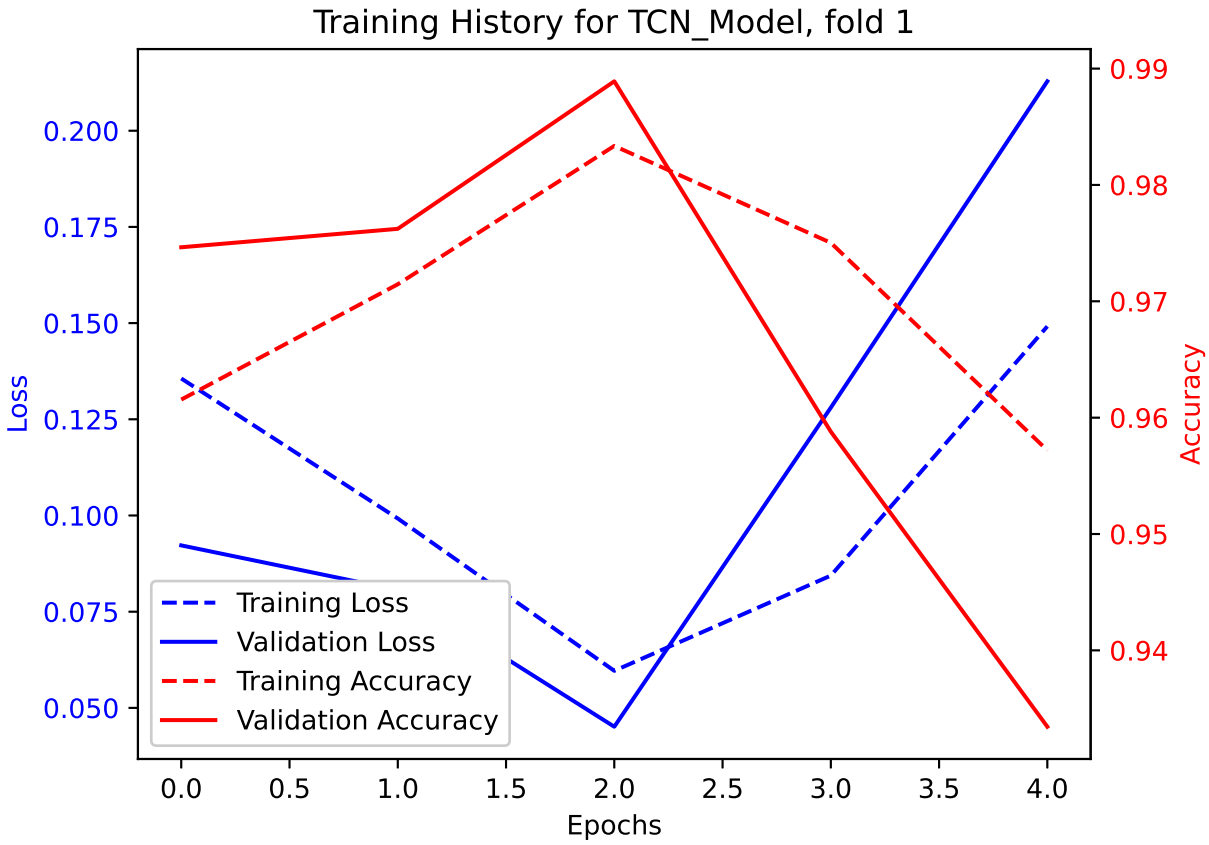
```
20/20 [==============================] - 1s 6ms/step
F1 score for TCN_Model, fold 0: 0.9668776343899625
20/20 [==============================] - 0s 5ms/step
```

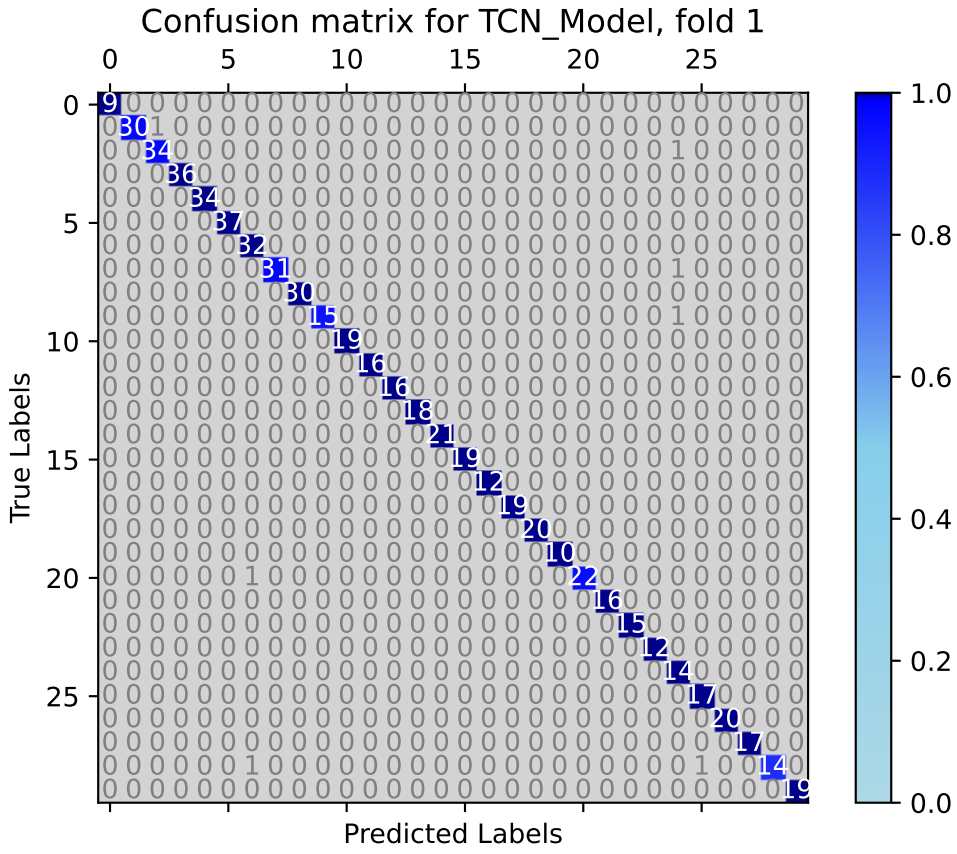

Training History for TCN_Model, fold 0
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy


Confusion matrix for TCN_Model, fold 0
True Labels
Predicted Labels

```
Epoch 1/5
77/79 [============================>.] - ETA: 0s - loss: 0.1330 - accuracy: 0.9627
Epoch 1: val_accuracy improved from -inf to 0.97464, saving model to /kaggle/working/TCN_Model_Fold_1_Checkpoint.h5
79/79 [==============================] - 16s 30ms/step - loss: 0.1356 - accuracy: 0.9616 - val_loss: 0.0922 - ...
   val_accuracy: 0.9746
Epoch 2/5
 1/79 [..............................] - ETA: 1s - loss: 0.0303 - accuracy: 1.0000


79/79 [==============================] - ETA: 0s - loss: 0.0992 - accuracy: 0.9715
Epoch 2: val_accuracy improved from 0.97464 to 0.97623, saving model to /kaggle/working/TCN_Model_Fold_1_Checkpoint.h5
79/79 [==============================] - 2s 20ms/step - loss: 0.0992 - accuracy: 0.9715 - val_loss: 0.0806 - ...
   val_accuracy: 0.9762
Epoch 3/5
79/79 [==============================] - ETA: 0s - loss: 0.0596 - accuracy: 0.9834
Epoch 3: val_accuracy improved from 0.97623 to 0.98891, saving model to /kaggle/working/TCN_Model_Fold_1_Checkpoint.h5
79/79 [==============================] - 2s 20ms/step - loss: 0.0596 - accuracy: 0.9834 - val_loss: 0.0451 - ...
   val_accuracy: 0.9889
Epoch 4/5
76/79 [============================>..] - ETA: 0s - loss: 0.0835 - accuracy: 0.9753
Epoch 4: val_accuracy did not improve from 0.98891
79/79 [==============================] - 1s 18ms/step - loss: 0.0844 - accuracy: 0.9750 - val_loss: 0.1280 - ...
   val_accuracy: 0.9588
Epoch 5/5
78/79 [============================>.] - ETA: 0s - loss: 0.1492 - accuracy: 0.9575
Epoch 5: val_accuracy did not improve from 0.98891
79/79 [==============================] - 1s 18ms/step - loss: 0.1492 - accuracy: 0.9572 - val_loss: 0.2128 - ...
   val_accuracy: 0.9334
```

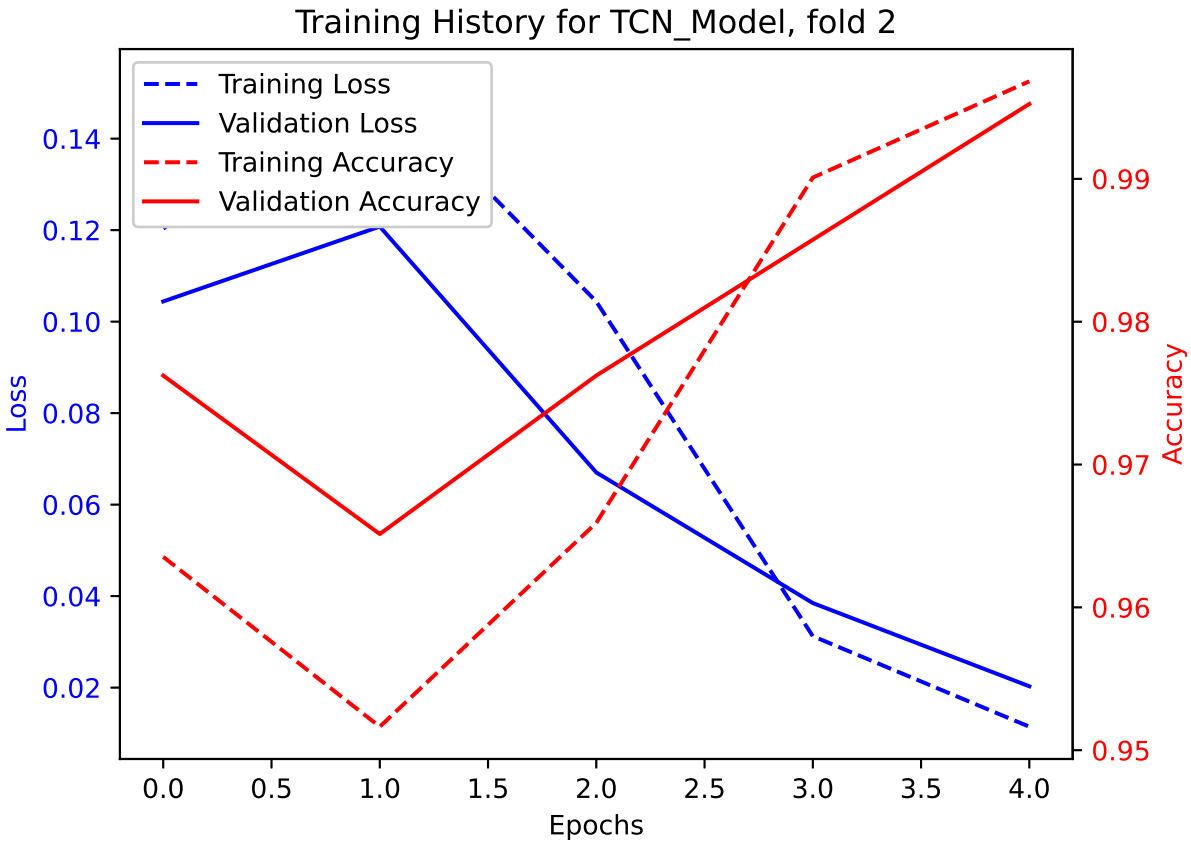
```
20/20 [==============================] - 1s 8ms/step - loss: 0.0451 - accuracy: 0.9889
Test Loss TCN_Model, fold 1: 0.04510357230901718, Test Accuracy TCN_Model, fold 1: 0.9889065027236938
20/20 [==============================] - 1s 6ms/step
F1 score for TCN_Model, fold 1: 0.9887455502368246
20/20 [==============================] - 0s 5ms/step
```

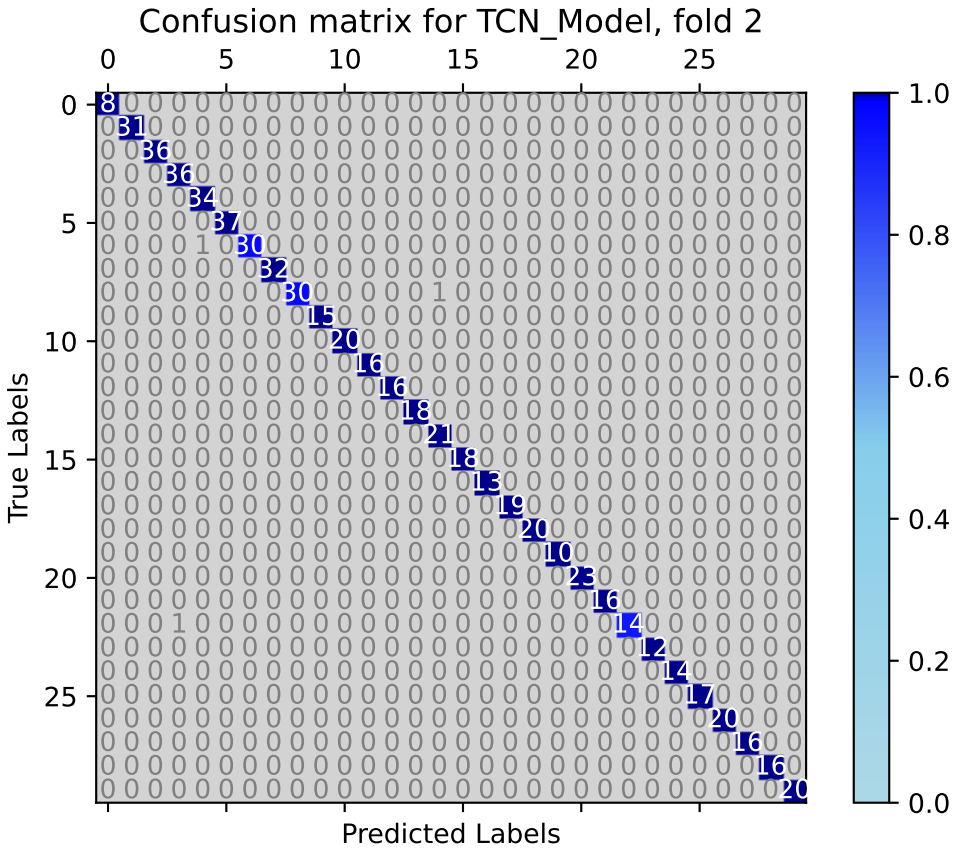

Training History for TCN_Model, fold 1
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy


Confusion matrix for TCN_Model, fold 1
True Labels
Predicted Labels

```
Epoch 1/5
79/79 [==============================] - ETA: 0s - loss: 0.1203 - accuracy: 0.9635
Epoch 1: val_accuracy improved from -inf to 0.97623, saving model to /kaggle/working/TCN_Model_Fold_2_Checkpoint.h5
79/79 [==============================] - 15s 30ms/step - loss: 0.1203 - accuracy: 0.9635 - val_loss: 0.1044 - ...
   val_accuracy: 0.9762
Epoch 2/5
 1/79 [..............................] - ETA: 1s - loss: 0.0115 - accuracy: 1.0000

77/79 [============================>.] - ETA: 0s - loss: 0.1536 - accuracy: 0.9513
Epoch 2: val_accuracy did not improve from 0.97623
79/79 [==============================] - 1s 19ms/step - loss: 0.1525 - accuracy: 0.9516 - val_loss: 0.1208 - ...
   val_accuracy: 0.9651
Epoch 3/5
77/79 [============================>.] - ETA: 0s - loss: 0.1042 - accuracy: 0.9659
Epoch 3: val_accuracy did not improve from 0.97623
79/79 [==============================] - 1s 18ms/step - loss: 0.1044 - accuracy: 0.9659 - val_loss: 0.0670 - ...
   val_accuracy: 0.9762
Epoch 4/5
79/79 [==============================] - ETA: 0s - loss: 0.0312 - accuracy: 0.9901
Epoch 4: val_accuracy improved from 0.97623 to 0.98574, saving model to /kaggle/working/TCN_Model_Fold_2_Checkpoint.h5
79/79 [==============================] - 2s 20ms/step - loss: 0.0312 - accuracy: 0.9901 - val_loss: 0.0385 - ...
   val_accuracy: 0.9857
Epoch 5/5
78/79 [============================>.] - ETA: 0s - loss: 0.0116 - accuracy: 0.9968
Epoch 5: val_accuracy improved from 0.98574 to 0.99525, saving model to /kaggle/working/TCN_Model_Fold_2_Checkpoint.h5
79/79 [==============================] - 2s 20ms/step - loss: 0.0115 - accuracy: 0.9968 - val_loss: 0.0203 - ...
   val_accuracy: 0.9952
```

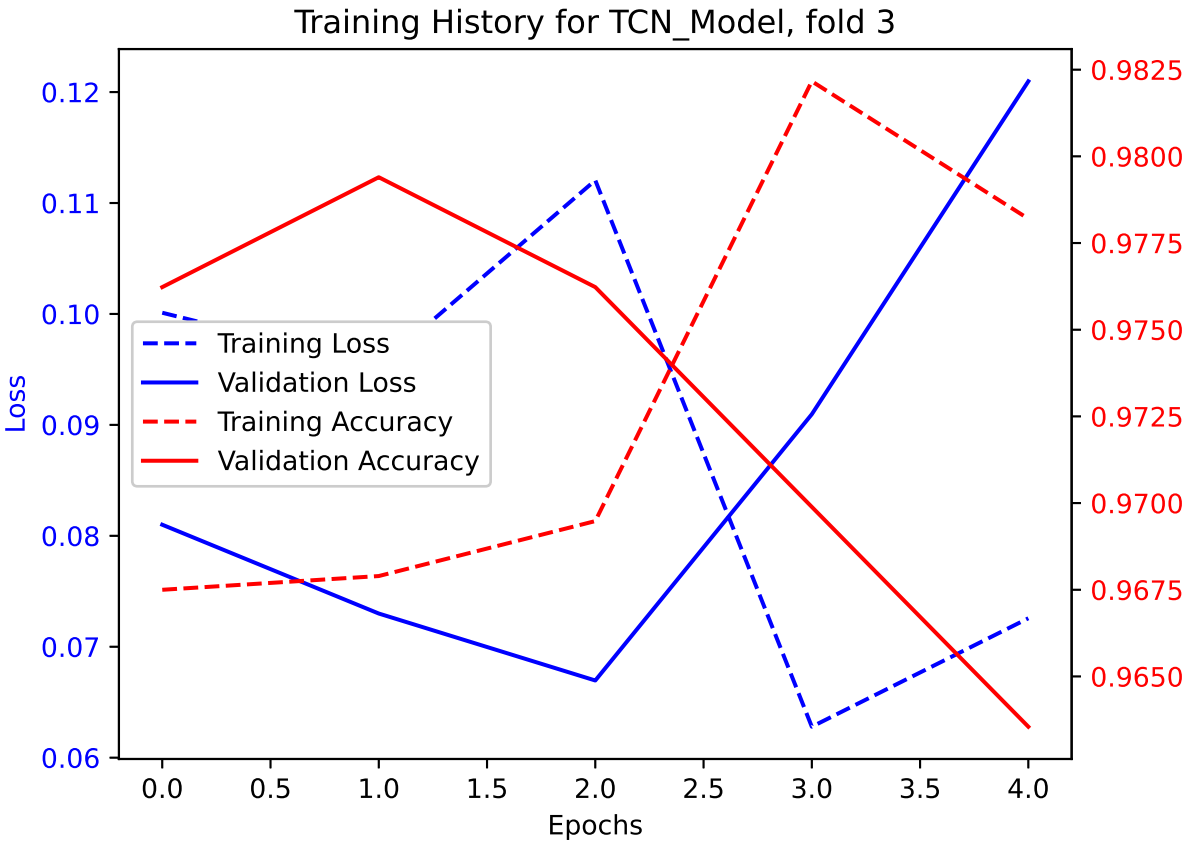
```
20/20 [==============================] - 1s 8ms/step - loss: 0.0203 - accuracy: 0.9952
Test Loss TCN_Model, fold 2: 0.02028728276491165, Test Accuracy TCN_Model, fold 2: 0.995245635509491
20/20 [==============================] - 1s 5ms/step
F1 score for TCN_Model, fold 2: 0.9960427719473249
20/20 [==============================] - 0s 5ms/step
```

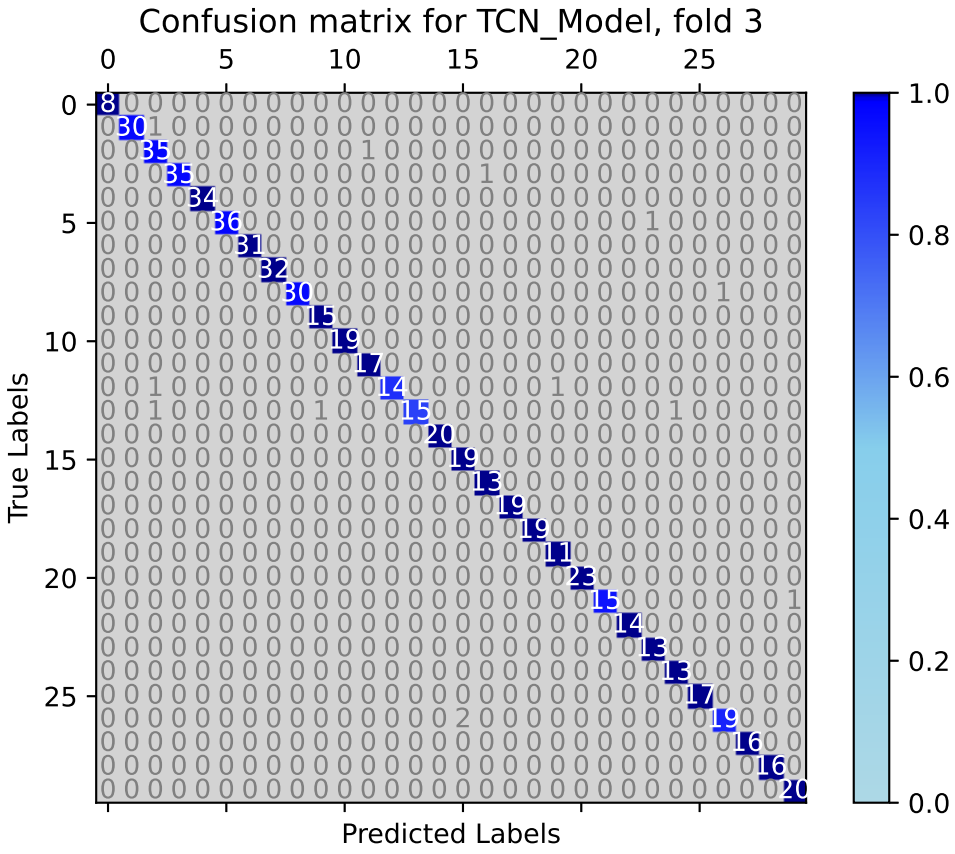

Training History for TCN_Model, fold 2
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
Loss
Accuracy
Epochs
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.99
0.98
0.97
0.96
0.95
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Confusion matrix for TCN_Model, fold 2
True Labels
Predicted Labels
0
5
10
15
20
25
1.0
0.8
0.6
0.4
0.2
0.0

```
Epoch 1/5
76/79 [============================>..] - ETA: 0s - loss: 0.1009 - accuracy: 0.9679
Epoch 1: val_accuracy improved from -inf to 0.97623, saving model to /kaggle/working/TCN_Model_Fold_3_Checkpoint.h5
79/79 [==============================] - 16s 31ms/step - loss: 0.1001 - accuracy: 0.9675 - val_loss: 0.0810 - ...
   val_accuracy: 0.9762
Epoch 2/5
 1/79 [..............................] - ETA: 1s - loss: 0.0522 - accuracy: 0.9688

77/79 [============================>.] - ETA: 0s - loss: 0.0969 - accuracy: 0.9675
Epoch 2: val_accuracy improved from 0.97623 to 0.97940, saving model to /kaggle/working/TCN_Model_Fold_3_Checkpoint.h5
79/79 [==============================] - 2s 20ms/step - loss: 0.0953 - accuracy: 0.9679 - val_loss: 0.0730 - ...
   val_accuracy: 0.9794
Epoch 3/5
79/79 [==============================] - ETA: 0s - loss: 0.1120 - accuracy: 0.9695
Epoch 3: val_accuracy did not improve from 0.97940
79/79 [==============================] - 1s 19ms/step - loss: 0.1120 - accuracy: 0.9695 - val_loss: 0.0670 - ...
   val_accuracy: 0.9762
Epoch 4/5
78/79 [============================>.] - ETA: 0s - loss: 0.0634 - accuracy: 0.9820
Epoch 4: val_accuracy did not improve from 0.97940
79/79 [==============================] - 1s 18ms/step - loss: 0.0628 - accuracy: 0.9822 - val_loss: 0.0910 - ...
   val_accuracy: 0.9699
Epoch 5/5
79/79 [==============================] - ETA: 0s - loss: 0.0726 - accuracy: 0.9782
Epoch 5: val_accuracy did not improve from 0.97940
79/79 [==============================] - 1s 18ms/step - loss: 0.0726 - accuracy: 0.9782 - val_loss: 0.1210 - ...
   val_accuracy: 0.9635
```

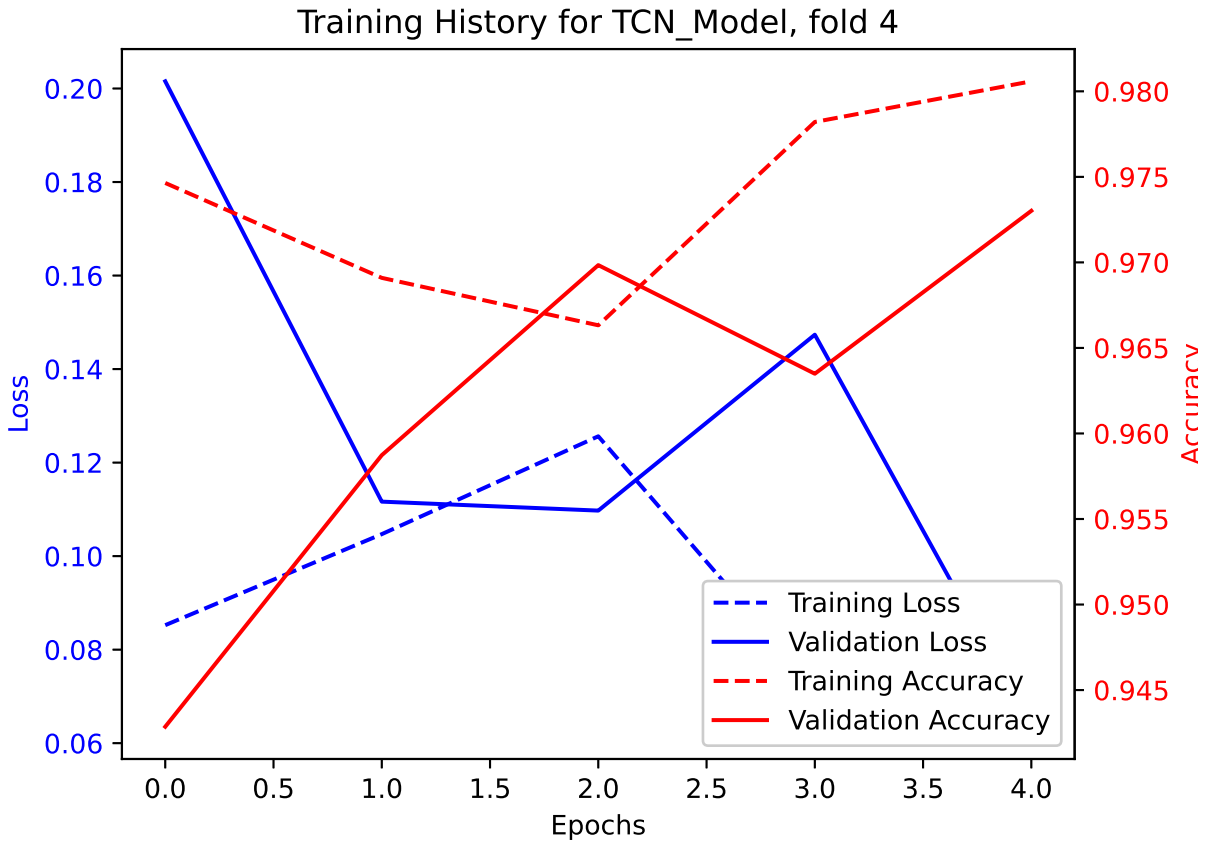
```
20/20 [==============================] - 1s 8ms/step - loss: 0.0730 - accuracy: 0.9794
Test Loss TCN_Model, fold 3: 0.07299331575632095, Test Accuracy TCN_Model, fold 3: 0.9793977737426758
20/20 [==============================] - 1s 6ms/step
F1 score for TCN_Model, fold 3: 0.9777520753583643
20/20 [==============================] - 0s 6ms/step
```

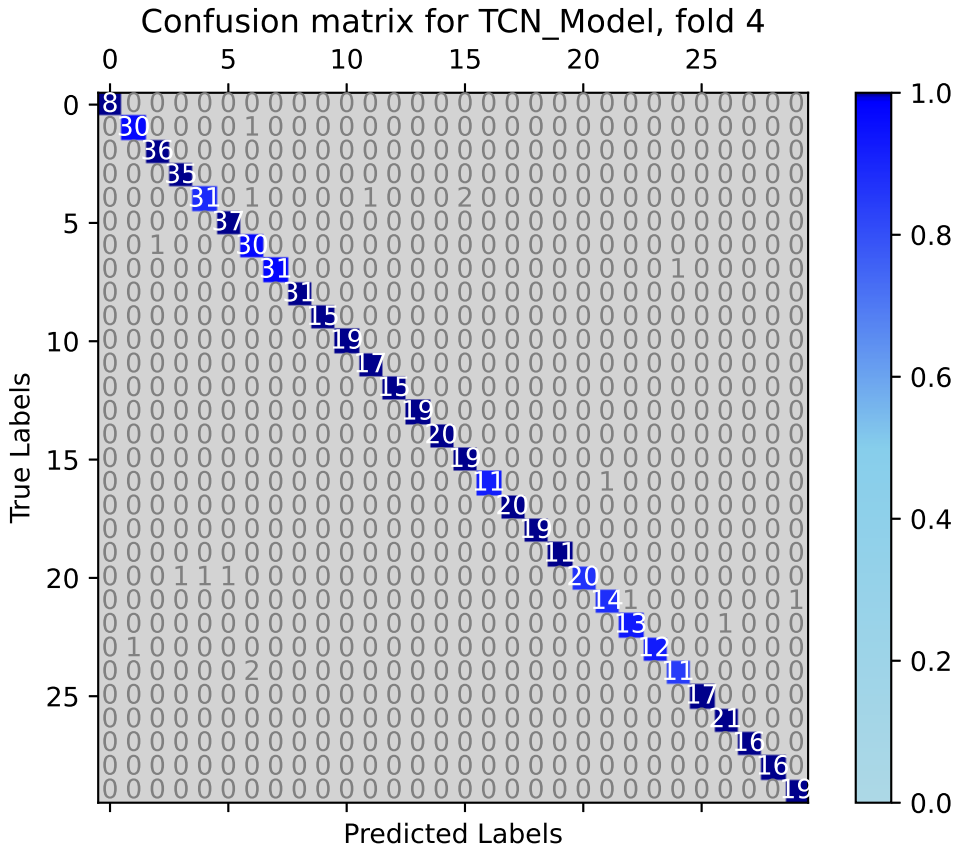

Training History for TCN_Model, fold 3
Loss
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
0.12
0.11
0.10
0.09
0.08
0.07
0.06
0.9825
0.9800
0.9775
0.9750
0.9725
0.9700
0.9675
0.9650
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Confusion matrix for TCN_Model, fold 3
True Labels
Predicted Labels
0
5
10
15
20
25
1.0
0.8
0.6
0.4
0.2
0.0

```
Epoch 1/5
79/79 [==============================] - ETA: 0s - loss: 0.0852 - accuracy: 0.9746
Epoch 1: val_accuracy improved from -inf to 0.94286, saving model to /kaggle/working/TCN_Model_Fold_4_Checkpoint.h5
79/79 [==============================] - 17s 36ms/step - loss: 0.0852 - accuracy: 0.9746 - val_loss: 0.2015 - ...
   val_accuracy: 0.9429
Epoch 2/5
 1/79 [..............................] - ETA: 1s - loss: 0.0201 - accuracy: 1.0000

76/79 [===========================>..] - ETA: 0s - loss: 0.1008 - accuracy: 0.9700
Epoch 2: val_accuracy improved from 0.94286 to 0.95873, saving model to /kaggle/working/TCN_Model_Fold_4_Checkpoint.h5
79/79 [==============================] - 2s 21ms/step - loss: 0.1047 - accuracy: 0.9691 - val_loss: 0.1117 - ...
   val_accuracy: 0.9587
Epoch 3/5
79/79 [==============================] - ETA: 0s - loss: 0.1256 - accuracy: 0.9663
Epoch 3: val_accuracy improved from 0.95873 to 0.96984, saving model to /kaggle/working/TCN_Model_Fold_4_Checkpoint.h5
79/79 [==============================] - 2s 20ms/step - loss: 0.1256 - accuracy: 0.9663 - val_loss: 0.1097 - ...
   val_accuracy: 0.9698
Epoch 4/5
76/79 [===========================>..] - ETA: 0s - loss: 0.0742 - accuracy: 0.9774
Epoch 4: val_accuracy did not improve from 0.96984
79/79 [==============================] - 1s 18ms/step - loss: 0.0719 - accuracy: 0.9782 - val_loss: 0.1474 - ...
   val_accuracy: 0.9635
Epoch 5/5
76/79 [===========================>..] - ETA: 0s - loss: 0.0804 - accuracy: 0.9803
Epoch 5: val_accuracy improved from 0.96984 to 0.97302, saving model to /kaggle/working/TCN_Model_Fold_4_Checkpoint.h5
79/79 [==============================] - 2s 20ms/step - loss: 0.0784 - accuracy: 0.9806 - val_loss: 0.0635 - ...
   val_accuracy: 0.9730
```

```
20/20 [==============================] - 1s 8ms/step - loss: 0.0635 - accuracy: 0.9730
Test Loss TCN_Model, fold 4: 0.06351867318153381, Test Accuracy TCN_Model, fold 4: 0.9730158448219299
20/20 [==============================] - 1s 6ms/step
F1 score for TCN_Model, fold 4: 0.9730095256876878
20/20 [==============================] - 0s 5ms/step
```


Training History for TCN_Model, fold 4
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy


Confusion matrix for TCN_Model, fold 4
True Labels
Predicted Labels

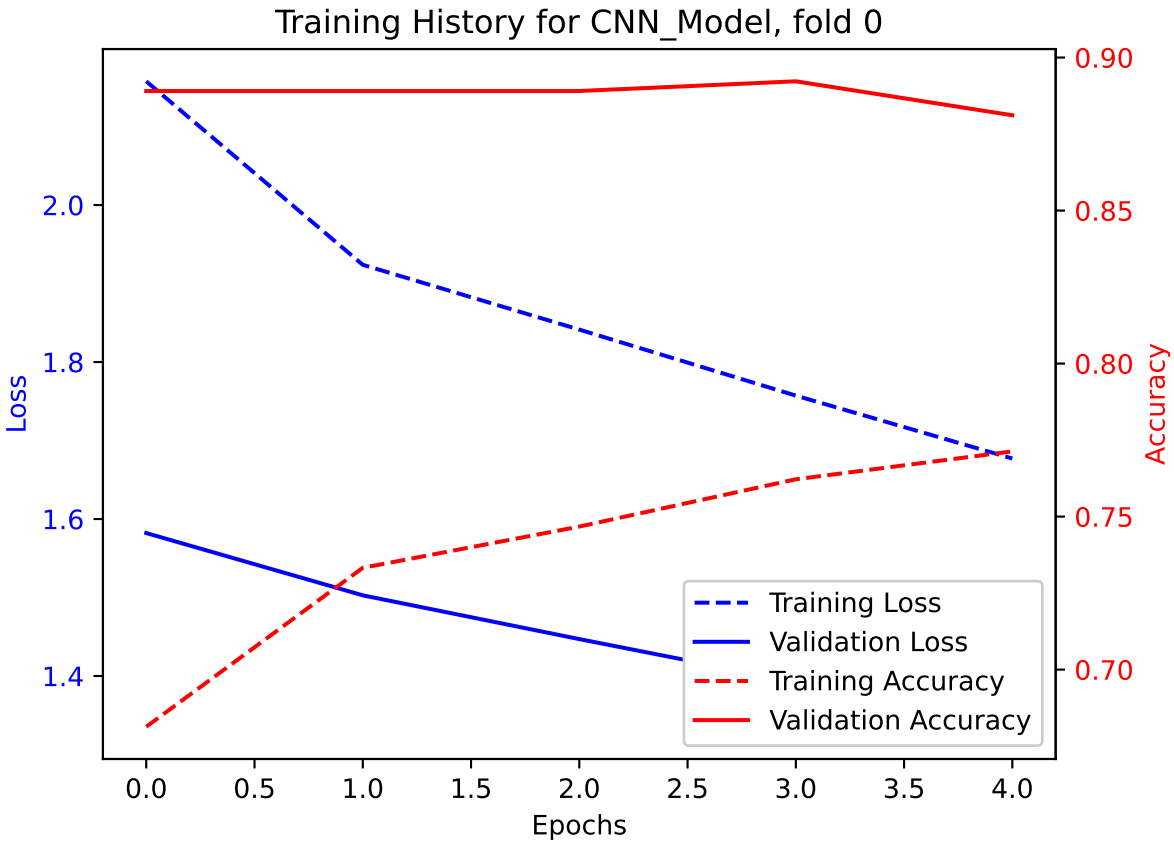
RESULTS for TCN model

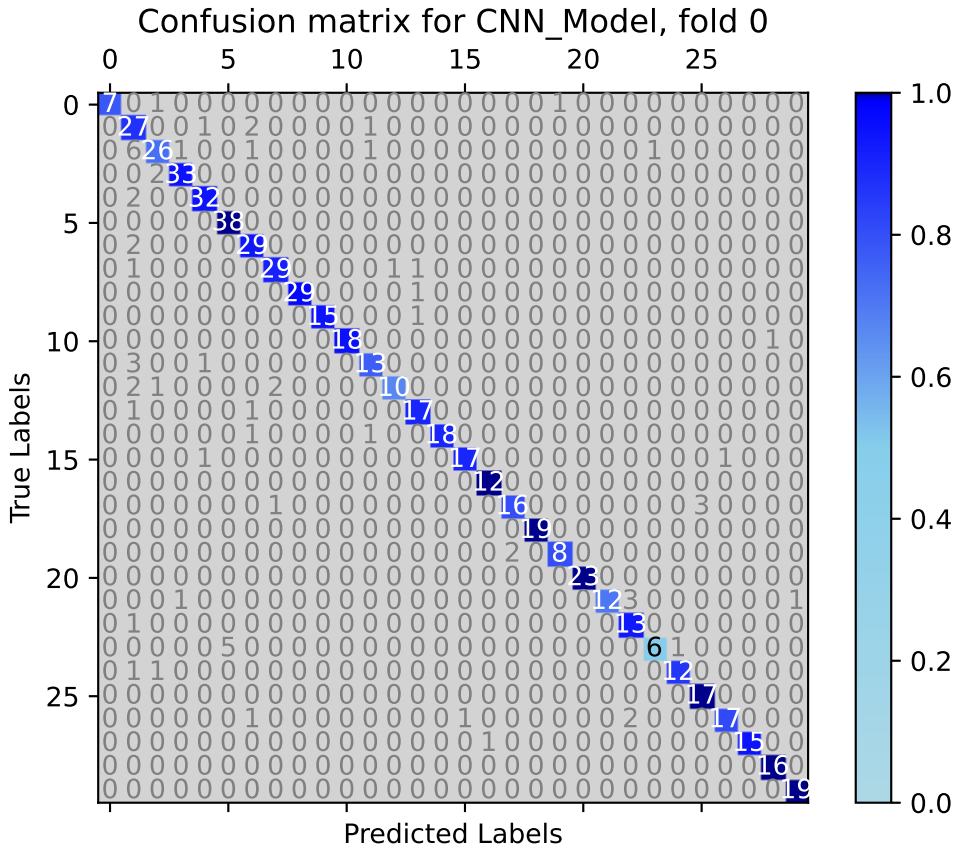
```
RESULTS for TCN_Model
Average F1 Score for TCN_Model: 0.980485511524033
F1 scores for all folds for TCN_Model: [0.9668776343899625, 0.9887455502368246, 0.9960427719473249, 0.9777520753583643, 0.9730095256876878
```

CNN Model

```
Epoch 1/5
79/79 [==============================] - ETA: 0s - loss: 2.1576 - accuracy: 0.6813
Epoch 1: val_accuracy improved from -inf to 0.88906, saving model to /kaggle/working/CNN_Model_Fold_0_Checkpoint.h5
79/79 [==============================] - 7s 19ms/step - loss: 2.1576 - accuracy: 0.6813 - val_loss: 1.5821 - ...
   val_accuracy: 0.8891
Epoch 2/5
 7/79 [=>............................] - ETA: 0s - loss: 2.0566 - accuracy: 0.6875

79/79 [==============================] - ETA: 0s - loss: 1.9239 - accuracy: 0.7333
Epoch 2: val_accuracy did not improve from 0.88906
79/79 [==============================] - 1s 10ms/step - loss: 1.9239 - accuracy: 0.7333 - val_loss: 1.5028 - ...
   val_accuracy: 0.8891
Epoch 3/5
79/79 [==============================] - ETA: 0s - loss: 1.8413 - accuracy: 0.7467
Epoch 3: val_accuracy did not improve from 0.88906
79/79 [==============================] - 1s 10ms/step - loss: 1.8413 - accuracy: 0.7467 - val_loss: 1.4470 - ...
   val_accuracy: 0.8891
Epoch 4/5
79/79 [==============================] - ETA: 0s - loss: 1.7572 - accuracy: 0.7622
Epoch 4: val_accuracy improved from 0.88906 to 0.89223, saving model to /kaggle/working/CNN_Model_Fold_0_Checkpoint.h5
79/79 [==============================] - 1s 11ms/step - loss: 1.7572 - accuracy: 0.7622 - val_loss: 1.3927 - ...
   val_accuracy: 0.8922
Epoch 5/5
79/79 [==============================] - ETA: 0s - loss: 1.6771 - accuracy: 0.7713
Epoch 5: val_accuracy did not improve from 0.89223
79/79 [==============================] - 1s 10ms/step - loss: 1.6771 - accuracy: 0.7713 - val_loss: 1.3354 - ...
   val_accuracy: 0.8811
```

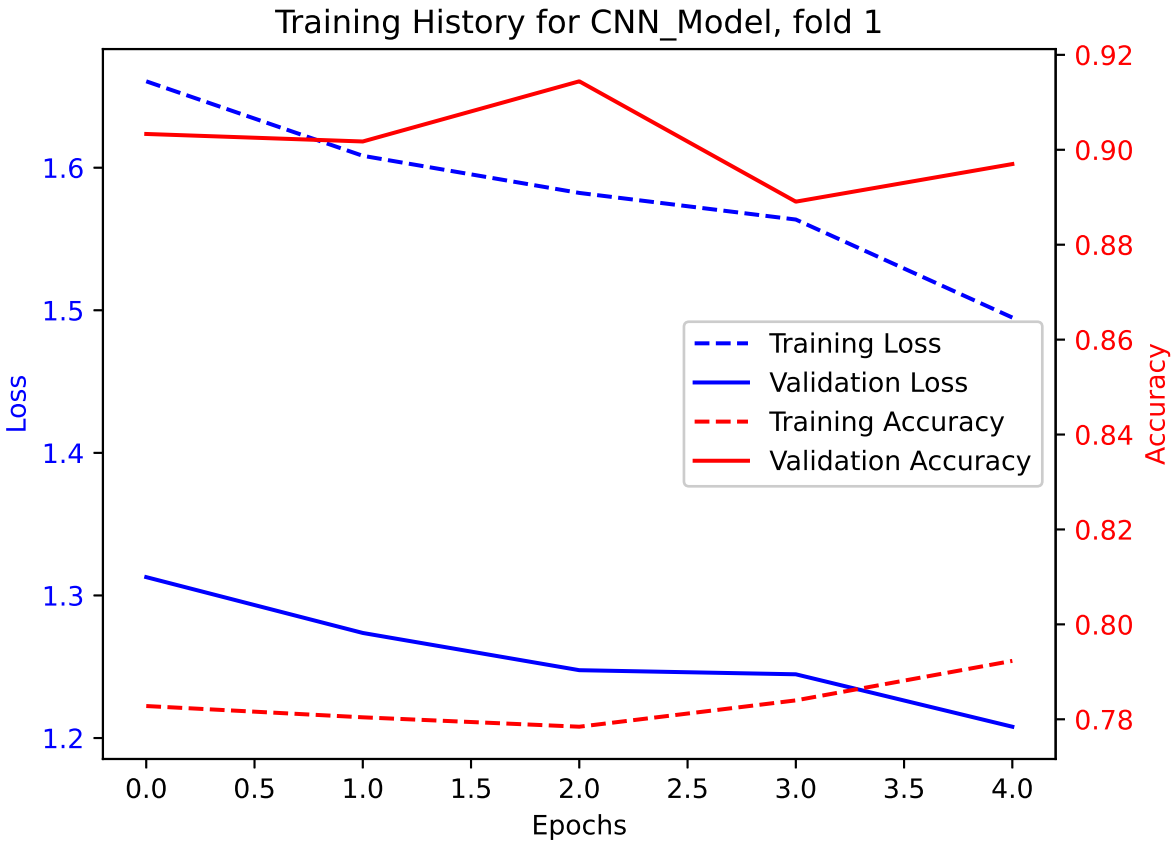
```
20/20 [==============================] - 0s 3ms/step - loss: 1.3927 - accuracy: 0.8922
Test Loss CNN_Model, fold 0: 1.392700433731079, Test Accuracy CNN_Model, fold 0: 0.8922345638275146
20/20 [==============================] - 0s 2ms/step
F1 score for CNN_Model, fold 0: 0.8890784909088278
20/20 [==============================] - 0s 2ms/step
```

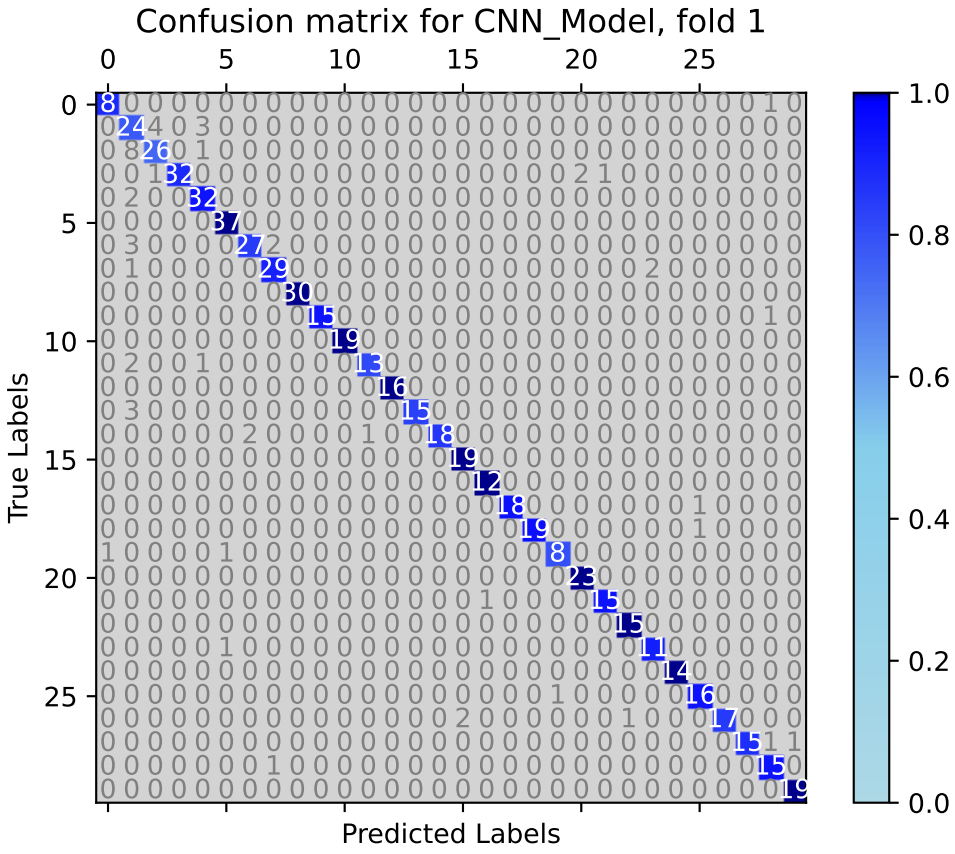

Training History for CNN_Model, fold 0
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy


Confusion matrix for CNN_Model, fold 0
True Labels
Predicted Labels

```
Epoch 1/5
79/79 [==============================] - ETA: 0s - loss: 1.6606 - accuracy: 0.7828
Epoch 1: val_accuracy improved from -inf to 0.90333, saving model to /kaggle/working/CNN_Model_Fold_1_Checkpoint.h5
79/79 [==============================] - 7s 16ms/step - loss: 1.6606 - accuracy: 0.7828 - val_loss: 1.3129 - ...
   val_accuracy: 0.9033
Epoch 2/5
 7/79 [=>............................] - ETA: 0s - loss: 1.6754 - accuracy: 0.7723

79/79 [==============================] - ETA: 0s - loss: 1.6083 - accuracy: 0.7804
Epoch 2: val_accuracy did not improve from 0.90333
79/79 [==============================] - 1s 10ms/step - loss: 1.6083 - accuracy: 0.7804 - val_loss: 1.2737 - ...
   val_accuracy: 0.9017
Epoch 3/5
79/79 [==============================] - ETA: 0s - loss: 1.5823 - accuracy: 0.7784
Epoch 3: val_accuracy improved from 0.90333 to 0.91442, saving model to /kaggle/working/CNN_Model_Fold_1_Checkpoint.h5
79/79 [==============================] - 1s 11ms/step - loss: 1.5823 - accuracy: 0.7784 - val_loss: 1.2476 - ...
   val_accuracy: 0.9144
Epoch 4/5
79/79 [==============================] - ETA: 0s - loss: 1.5636 - accuracy: 0.7840
Epoch 4: val_accuracy did not improve from 0.91442
79/79 [==============================] - 1s 10ms/step - loss: 1.5636 - accuracy: 0.7840 - val_loss: 1.2447 - ...
   val_accuracy: 0.8891
Epoch 5/5
79/79 [==============================] - ETA: 0s - loss: 1.4949 - accuracy: 0.7923
Epoch 5: val_accuracy did not improve from 0.91442
79/79 [==============================] - 1s 10ms/step - loss: 1.4949 - accuracy: 0.7923 - val_loss: 1.2079 - ...
   val_accuracy: 0.8970
```

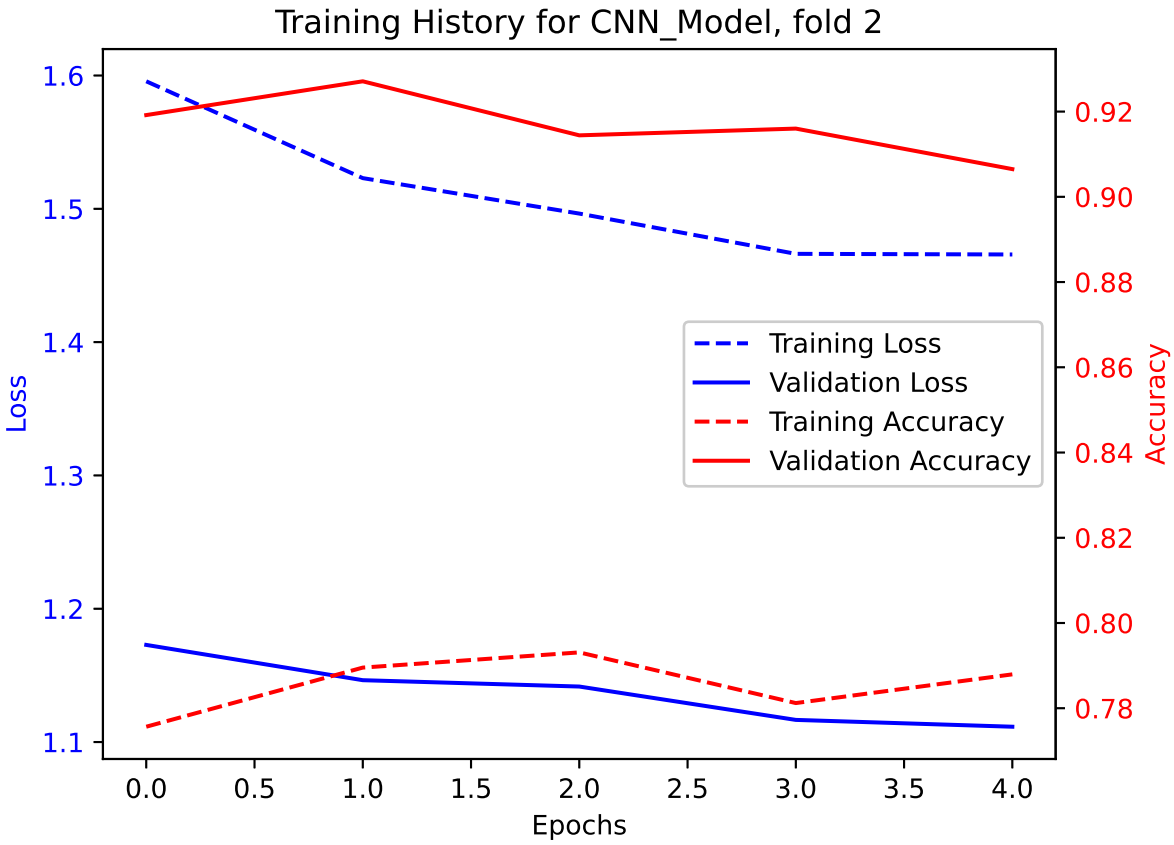
```
20/20 [==============================] - 0s 3ms/step - loss: 1.2476 - accuracy: 0.9144
Test Loss CNN_Model, fold 1: 1.2475864887237549, Test Accuracy CNN_Model, fold 1: 0.914421558380127
20/20 [==============================] - 0s 2ms/step
F1 score for CNN_Model, fold 1: 0.922339560707998
20/20 [==============================] - 0s 2ms/step
```

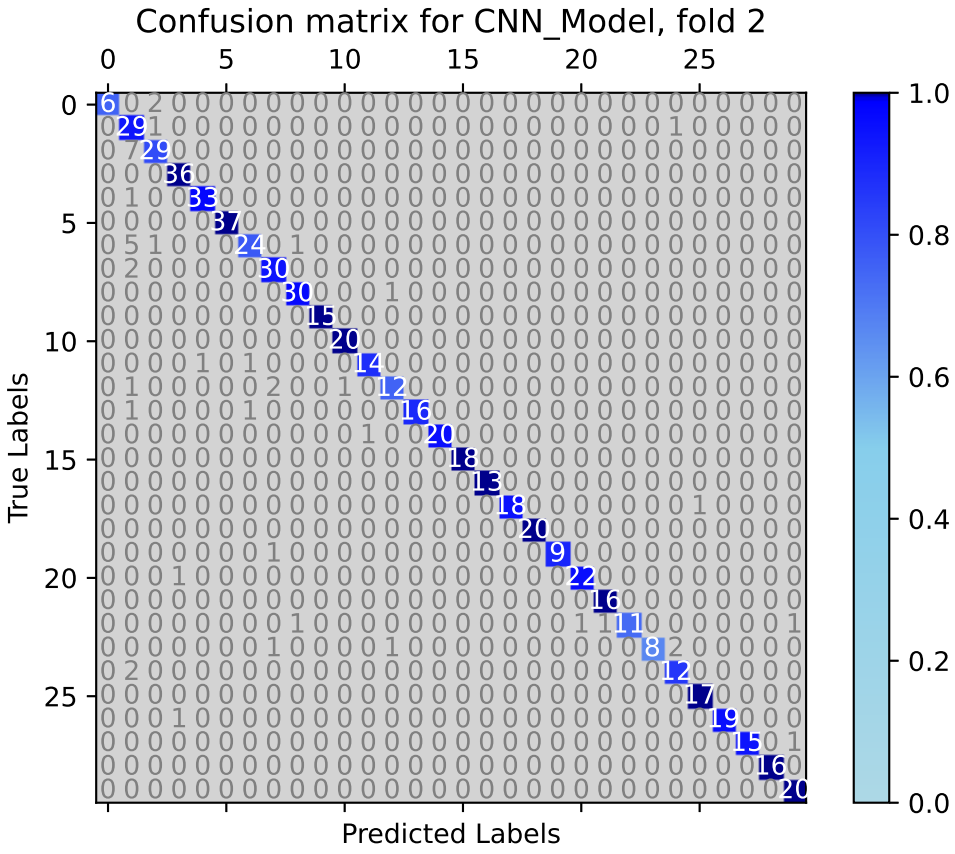

Training History for CNN_Model, fold 1
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
1.6
1.5
1.4
1.3
1.2
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0.78
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Confusion matrix for CNN_Model, fold 1
True Labels
Predicted Labels
0
5
10
15
20
25
1.0
0.8
0.6
0.4
0.2
0.0

```
Epoch 1/5
77/79 [============================>.] - ETA: 0s - loss: 1.5930 - accuracy: 0.7760
Epoch 1: val_accuracy improved from -inf to 0.91918, saving model to /kaggle/working/CNN_Model_Fold_2_Checkpoint.h5
79/79 [==============================] - 7s 16ms/step - loss: 1.5956 - accuracy: 0.7757 - val_loss: 1.1729 - ...
   val_accuracy: 0.9192
Epoch 2/5
 7/79 [=>............................] - ETA: 0s - loss: 1.6082 - accuracy: 0.7366

79/79 [==============================] - ETA: 0s - loss: 1.5230 - accuracy: 0.7895
Epoch 2: val_accuracy improved from 0.91918 to 0.92710, saving model to /kaggle/working/CNN_Model_Fold_2_Checkpoint.h5
79/79 [==============================] - 1s 11ms/step - loss: 1.5230 - accuracy: 0.7895 - val_loss: 1.1464 - ...
   val_accuracy: 0.9271
Epoch 3/5
79/79 [==============================] - ETA: 0s - loss: 1.4964 - accuracy: 0.7931
Epoch 3: val_accuracy did not improve from 0.92710
79/79 [==============================] - 1s 10ms/step - loss: 1.4964 - accuracy: 0.7931 - val_loss: 1.1416 - ...
   val_accuracy: 0.9144
Epoch 4/5
79/79 [==============================] - ETA: 0s - loss: 1.4662 - accuracy: 0.7812
Epoch 4: val_accuracy did not improve from 0.92710
79/79 [==============================] - 1s 15ms/step - loss: 1.4662 - accuracy: 0.7812 - val_loss: 1.1166 - ...
   val_accuracy: 0.9160
Epoch 5/5
79/79 [==============================] - ETA: 0s - loss: 1.4658 - accuracy: 0.7880
Epoch 5: val_accuracy did not improve from 0.92710
79/79 [==============================] - 1s 10ms/step - loss: 1.4658 - accuracy: 0.7880 - val_loss: 1.1115 - ...
   val_accuracy: 0.9065
```

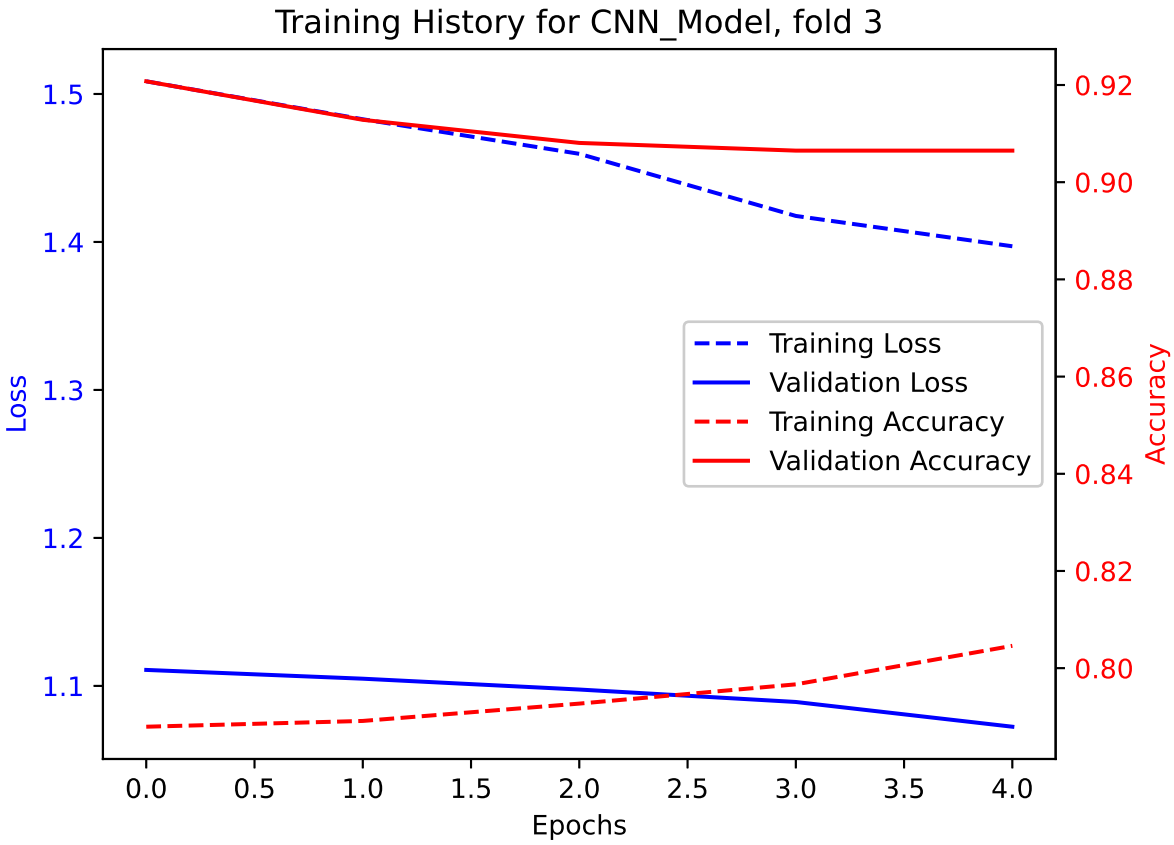
```
20/20 [==============================] - 0s 3ms/step - loss: 1.1464 - accuracy: 0.9271
Test Loss CNN_Model, fold 2: 1.1464018821716309, Test Accuracy CNN_Model, fold 2: 0.9270998239517212
20/20 [==============================] - 0s 2ms/step
F1 score for CNN_Model, fold 2: 0.9286956831388417
20/20 [==============================] - 0s 2ms/step
```

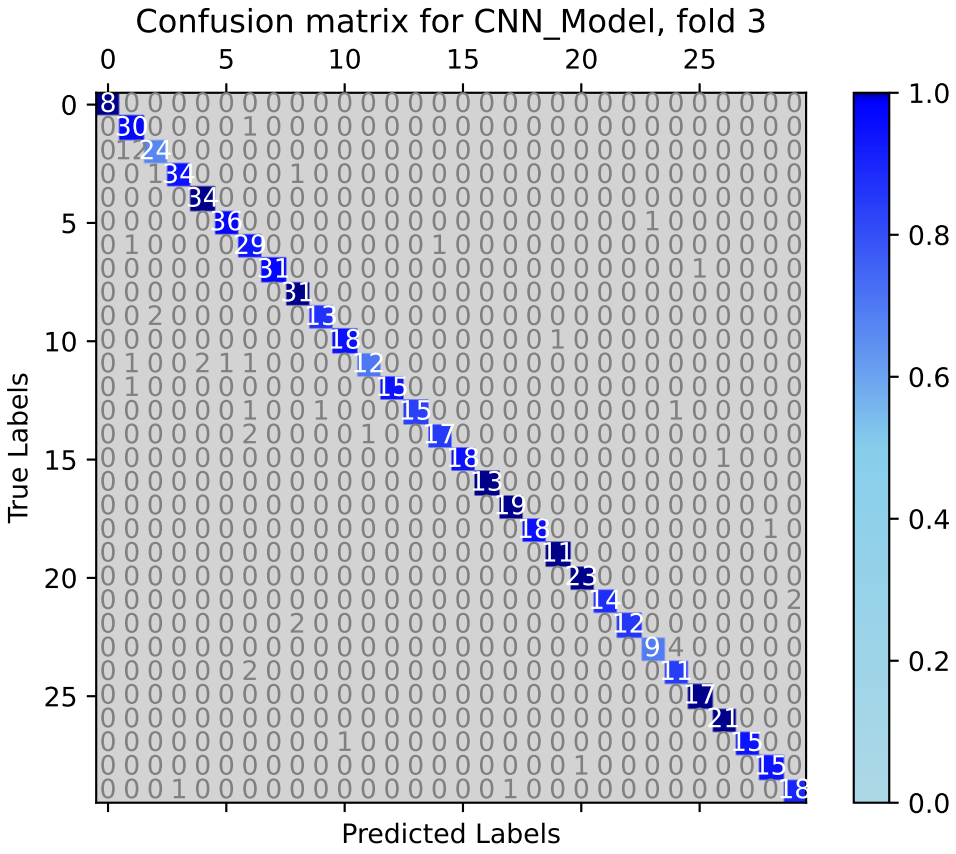

Training History for CNN_Model, fold 2
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
1.6
1.5
1.4
1.3
1.2
1.1
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0.78
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Confusion matrix for CNN_Model, fold 2
True Labels
Predicted Labels
0
5
10
15
20
25
1.0
0.8
0.6
0.4
0.2
0.0

```
Epoch 1/5
79/79 [==============================] - ETA: 0s - loss: 1.5086 - accuracy: 0.7880
Epoch 1: val_accuracy improved from -inf to 0.92076, saving model to /kaggle/working/CNN_Model_Fold_3_Checkpoint.h5
79/79 [==============================] - 7s 15ms/step - loss: 1.5086 - accuracy: 0.7880 - val_loss: 1.1108 - ...
   val_accuracy: 0.9208
Epoch 2/5
 7/79 [=>............................] - ETA: 0s - loss: 1.5697 - accuracy: 0.7455

79/79 [==============================] - ETA: 0s - loss: 1.4829 - accuracy: 0.7891
Epoch 2: val_accuracy did not improve from 0.92076
79/79 [==============================] - 1s 10ms/step - loss: 1.4829 - accuracy: 0.7891 - val_loss: 1.1049 - ...
   val_accuracy: 0.9128
Epoch 3/5
79/79 [==============================] - ETA: 0s - loss: 1.4596 - accuracy: 0.7927
Epoch 3: val_accuracy did not improve from 0.92076
79/79 [==============================] - 1s 10ms/step - loss: 1.4596 - accuracy: 0.7927 - val_loss: 1.0975 - ...
   val_accuracy: 0.9081
Epoch 4/5
79/79 [==============================] - ETA: 0s - loss: 1.4176 - accuracy: 0.7967
Epoch 4: val_accuracy did not improve from 0.92076
79/79 [==============================] - 1s 10ms/step - loss: 1.4176 - accuracy: 0.7967 - val_loss: 1.0891 - ...
   val_accuracy: 0.9065
Epoch 5/5
78/79 [============================>.] - ETA: 0s - loss: 1.3959 - accuracy: 0.8045
Epoch 5: val_accuracy did not improve from 0.92076
79/79 [==============================] - 1s 10ms/step - loss: 1.3971 - accuracy: 0.8046 - val_loss: 1.0724 - ...
   val_accuracy: 0.9065
```

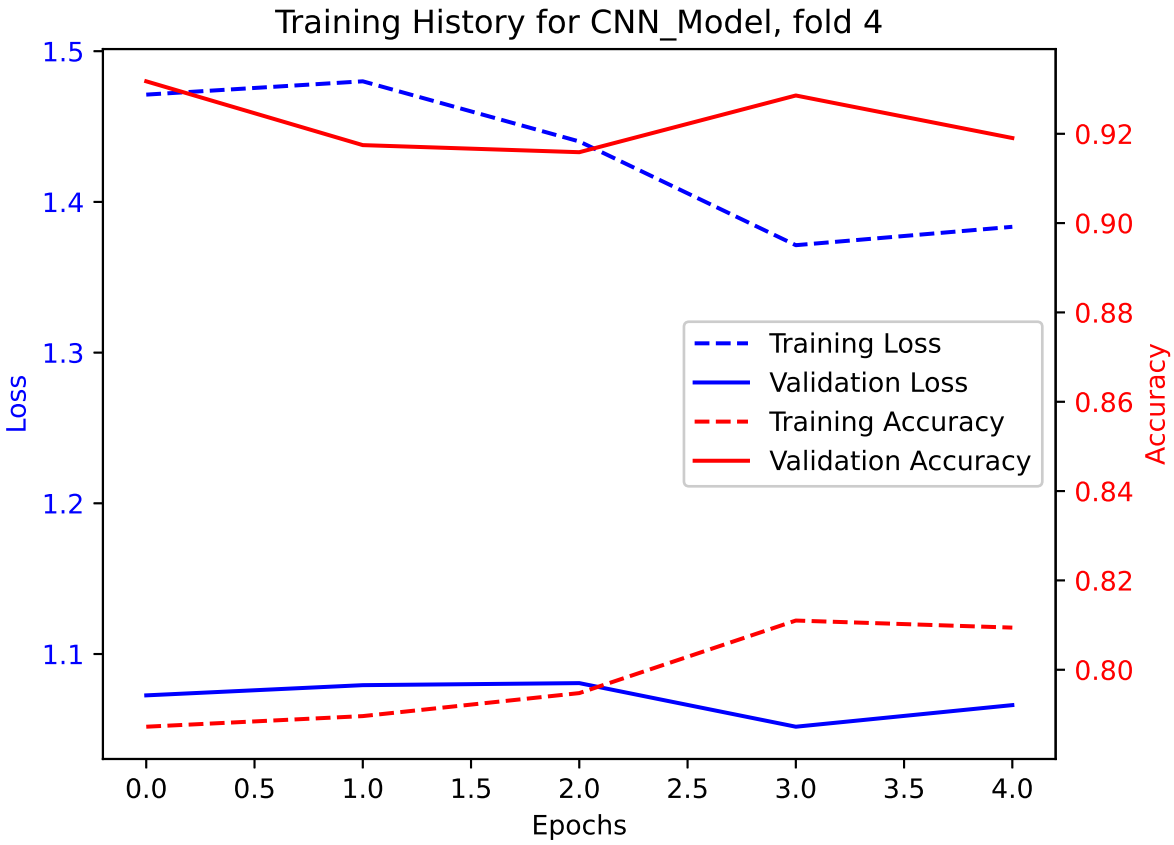
```
20/20 [==============================] - 0s 3ms/step - loss: 1.1108 - accuracy: 0.9208
Test Loss CNN_Model, fold 3: 1.1107983589172363, Test Accuracy CNN_Model, fold 3: 0.9207606911659241
20/20 [==============================] - 0s 2ms/step
F1 score for CNN_Model, fold 3: 0.922642182646403
20/20 [==============================] - 0s 2ms/step
```

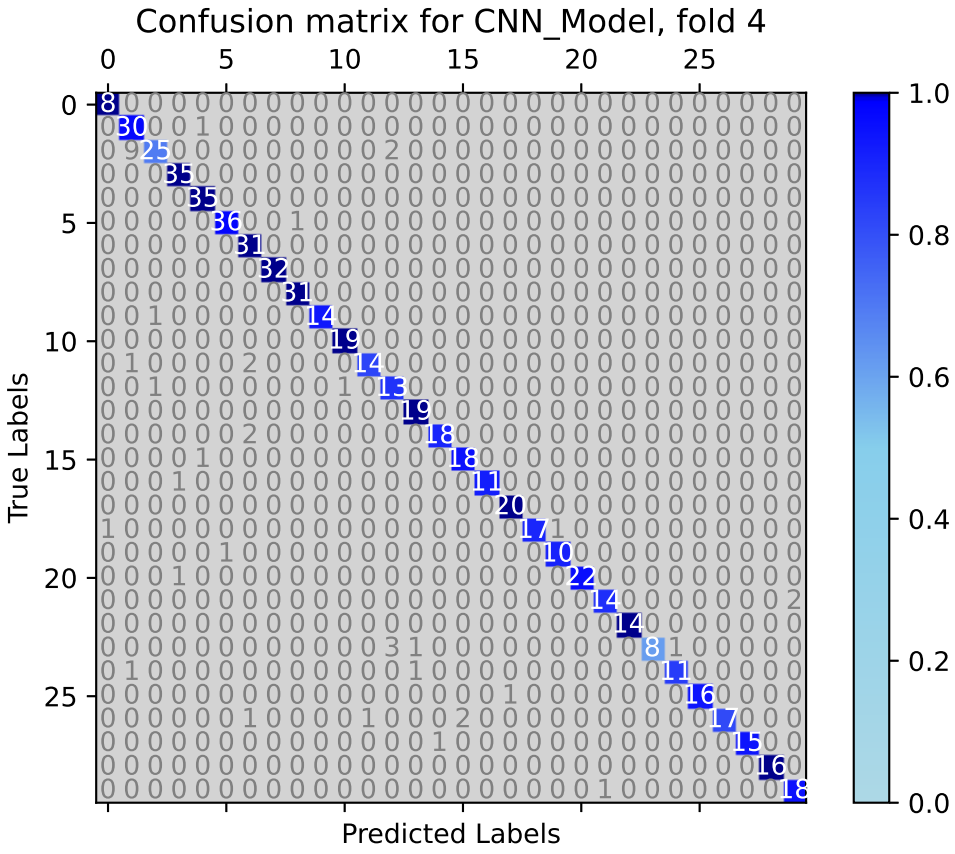

Training History for CNN_Model, fold 3
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
1.5
1.4
1.3
1.2
1.1
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Confusion matrix for CNN_Model, fold 3
True Labels
Predicted Labels
0
5
10
15
20
25
1.0
0.8
0.6
0.4
0.2
0.0

```
Epoch 1/5
79/79 [==============================] - ETA: 0s - loss: 1.4712 - accuracy: 0.7872
Epoch 1: val_accuracy improved from -inf to 0.93175, saving model to /kaggle/working/CNN_Model_Fold_4_Checkpoint.h5
79/79 [==============================] - 8s 20ms/step - loss: 1.4712 - accuracy: 0.7872 - val_loss: 1.0726 - ...
   val_accuracy: 0.9317
Epoch 2/5
 7/79 [=>............................] - ETA: 0s - loss: 1.5645 - accuracy: 0.7589

79/79 [==============================] - ETA: 0s - loss: 1.4800 - accuracy: 0.7896
Epoch 2: val_accuracy did not improve from 0.93175
79/79 [==============================] - 1s 10ms/step - loss: 1.4800 - accuracy: 0.7896 - val_loss: 1.0793 - ...
   val_accuracy: 0.9175
Epoch 3/5
77/79 [============================>.] - ETA: 0s - loss: 1.4384 - accuracy: 0.7946
Epoch 3: val_accuracy did not improve from 0.93175
79/79 [==============================] - 1s 10ms/step - loss: 1.4401 - accuracy: 0.7948 - val_loss: 1.0807 - ...
   val_accuracy: 0.9159
Epoch 4/5
79/79 [==============================] - ETA: 0s - loss: 1.3713 - accuracy: 0.8110
Epoch 4: val_accuracy did not improve from 0.93175
79/79 [==============================] - 1s 10ms/step - loss: 1.3713 - accuracy: 0.8110 - val_loss: 1.0518 - ...
   val_accuracy: 0.9286
Epoch 5/5
79/79 [==============================] - ETA: 0s - loss: 1.3835 - accuracy: 0.8094
Epoch 5: val_accuracy did not improve from 0.93175
79/79 [==============================] - 1s 10ms/step - loss: 1.3835 - accuracy: 0.8094 - val_loss: 1.0661 - ...
   val_accuracy: 0.9190
```

```
20/20 [==============================] - 0s 3ms/step - loss: 1.0726 - accuracy: 0.9317
Test Loss CNN_Model, fold 4: 1.0725630521774292, Test Accuracy CNN_Model, fold 4: 0.9317460060119629
20/20 [==============================] - 0s 2ms/step
F1 score for CNN_Model, fold 4: 0.9285938282825523
20/20 [==============================] - 0s 2ms/step
```

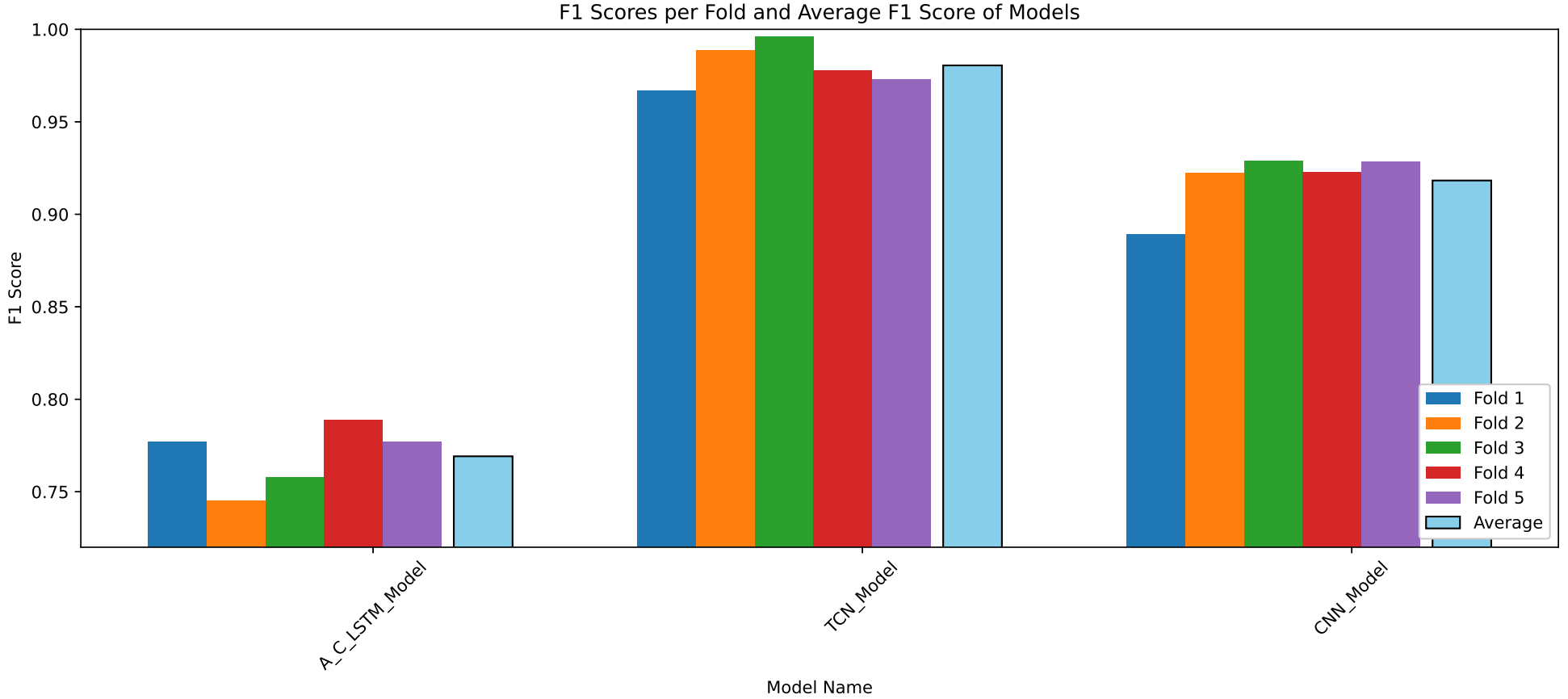

Training History for CNN_Model, fold 4
Loss
Accuracy
Epochs
Training Loss
Validation Loss
Training Accuracy
Validation Accuracy
1.5
1.4
1.3
1.2
1.1
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Confusion matrix for CNN_Model, fold 4
True Labels
Predicted Labels
0
5
10
15
20
25
1.0
0.8
0.6
0.4
0.2
0.0

RESULTS for CNN model

```
RESULTS for CNN_Model
Average F1 Score for CNN_Model: 0.9182699491369245
F1 scores for all folds for CNN_Model: [0.8890784909088278, 0.922339560707998, 0.9286956831388417, 0.922642182646403, 0.9285938282825523]
```

Results from Max F1-Averaged Stratified k-Fold of GI models



Model Name	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average F1
A_C_LSTM_Model	0.7771861305935289	0.7451250906981052	0.7576101883479243	0.7887851274015556	0.777085410226929	0.7691583894536087
TCN_Model	0.9668776343899625	0.9887455502368246	0.9960427719473249	0.9777520753583643	0.9730095256876878	0.980485511524033
CNN_Model	0.8890784909088278	0.922339560707998	0.9286956831388417	0.922642182646403	0.9285938282825523	0.9182699491369245

```
print(model_metrics)
```

```
{'A_C_LSTM_Model': {'F1_Scores': [0.7771861305935289, 0.7451250906981052, 0.7576101883479243, 0.7887851274015556, ...
   0.777085410226929], 'Average_F1': 0.7691583894536087},

'TCN_Model': {'F1_Scores': [0.9668776343899625, 0.9887455502368246, 0.9960427719473249, 0.9777520753583643, ...
   0.9730095256876878], 'Average_F1': 0.980485511524033},

'CNN_Model': {'F1_Scores': [0.8890784909088278, 0.922339560707998, 0.9286956831388417, 0.922642182646403, ...
   0.9285938282825523], 'Average_F1': 0.9182699491369245}}
```

```
!zip -r TopMan_ID_Models_Val.zip /kaggle/working
```

```
  adding: kaggle/working/ (stored 0%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_1_A_C_LSTM_Model.eps (deflated 98%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_0_CNN_Model.eps (deflated 98%)
  adding: kaggle/working/Training_History_fold1_TCN_Model.eps (deflated 72%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_2_A_C_LSTM_Model.eps (deflated 98%)
  adding: kaggle/working/Training_History_fold1_A_C_LSTM_Model.eps (deflated 73%)
  adding: kaggle/working/UIA_IMU_ID_Models_Val.zip (stored 0%)
  adding: kaggle/working/TCN_Model_Fold_2_Checkpoint.h5 (deflated 11%)
  adding: kaggle/working/CNN_Model_Fold_3_Checkpoint.h5 (deflated 40%)
  adding: kaggle/working/A_C_LSTM_Model_Fold_3_Checkpoint.h5 (deflated 9%)
  adding: kaggle/working/A_C_LSTM_Model_Fold_0_Checkpoint.h5 (deflated 9%)
  adding: kaggle/working/TCN_Model_Fold_3_Checkpoint.h5 (deflated 11%)
  adding: kaggle/working/A_C_LSTM_Model_Fold_2_Checkpoint.h5 (deflated 9%)
  adding: kaggle/working/Training_History_fold4_TCN_Model.eps (deflated 73%)
  adding: kaggle/working/f1_scores_grouped_performance.jpeg (deflated 48%)
  adding: kaggle/working/A_C_LSTM_Model_Fold_4_Checkpoint.h5 (deflated 9%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_0_TCN_Model.eps (deflated 98%)
  adding: kaggle/working/Prop_Confusion_Matrix_fold_1_CNN_Model.eps (deflated 98%)
  adding: kaggle/working/Training_History_fold3_TCN_Model.eps (deflated 73%)
  adding: kaggle/working/Training_History_fold2_A_C_LSTM_Model.eps (deflated 72%)
  adding: kaggle/working/CNN_Model_Fold_1_Checkpoint.h5 (deflated 39%)
```

```
adding: kaggle/working/Training_History_fold2_CNN_Model.eps (deflated 72%)
adding: kaggle/working/Training_History_fold4_A_C_LSTM_Model.eps (deflated 71%)
adding: kaggle/working/f1_scores_table.jpeg (deflated 22%)
adding: kaggle/working/Prop_Confusion_Matrix_fold_3_TCN_Model.eps (deflated 98%)
adding: kaggle/working/Training_History_fold3_CNN_Model.eps (deflated 72%)
adding: kaggle/working/CNN_Model_Fold_2_Checkpoint.h5 (deflated 40%)
adding: kaggle/working/Training_History_fold1_CNN_Model.eps (deflated 72%)
adding: kaggle/working/Prop_Confusion_Matrix_fold_0_A_C_LSTM_Model.eps (deflated 98%)
adding: kaggle/working/Prop_Confusion_Matrix_fold_3_A_C_LSTM_Model.eps (deflated 98%)
adding: kaggle/working/TCN_Model_Fold_1_Checkpoint.h5 (deflated 11%)
adding: kaggle/working/Training_History_fold3_A_C_LSTM_Model.eps (deflated 72%)
adding: kaggle/working/TCN_Model_Fold_4_Checkpoint.h5 (deflated 11%)
adding: kaggle/working/CNN_Model_Fold_0_Checkpoint.h5 (deflated 39%)
adding: kaggle/working/Prop_Confusion_Matrix_fold_2_CNN_Model.eps (deflated 98%)
adding: kaggle/working/f1_scores_table.eps (deflated 72%)
adding: kaggle/working/Prop_Confusion_Matrix_fold_2_TCN_Model.eps (deflated 98%)
adding: kaggle/working/Prop_Confusion_Matrix_fold_1_TCN_Model.eps (deflated 98%)
adding: kaggle/working/Training_History_fold0_CNN_Model.eps (deflated 71%)
adding: kaggle/working/.virtual_documents/ (stored 0%)
adding: kaggle/working/Training_History_fold0_A_C_LSTM_Model.eps (deflated 71%)
adding: kaggle/working/Training_History_fold4_CNN_Model.eps (deflated 72%)
adding: kaggle/working/Prop_Confusion_Matrix_fold_4_CNN_Model.eps (deflated 98%)
adding: kaggle/working/Prop_Confusion_Matrix_fold_4_TCN_Model.eps (deflated 98%)
adding: kaggle/working/f1_scores_grouped_performance.eps (deflated 69%)
adding: kaggle/working/Training_History_fold2_TCN_Model.eps (deflated 72%)
adding: kaggle/working/A_C_LSTM_Model_Fold_1_Checkpoint.h5 (deflated 9%)
adding: kaggle/working/Prop_Confusion_Matrix_fold_3_CNN_Model.eps (deflated 98%)
adding: kaggle/working/Training_History_fold0_TCN_Model.eps (deflated 72%)
adding: kaggle/working/Prop_Confusion_Matrix_fold_4_A_C_LSTM_Model.eps (deflated 98%)
adding: kaggle/working/CNN_Model_Fold_4_Checkpoint.h5 (deflated 40%)
adding: kaggle/working/TCN_Model_Fold_0_Checkpoint.h5 (deflated 11%)
```

Appendix: Project management

Team meetings

Project idea / formulation of thesis

The title and general idea of the project was formed during two meetings with supervisor Dr. Filippo and machine learning mentor Hamza Zafar. The meetings took place on the 22nd of May and 23rd of June 2023. The meeting minutes are provided below:

Team meeting, 22.05.2023

Participants: Dr. F , H and Martin

Agenda:

System arcitechture. Team review: proposed system arcitechture.

Idea: Two stage ML system made to compensate for IMU weakness (when used as position sensor): double integration, noise etc.

Feedback:

Such a system would require a large amount of manual work, labeling the different states. Supervised learning.

Proposed change:

Use "sensor fusion", i.e. combination of IMU and Qualisys data, to allow AI/ML-scheme to reckognize state of system. (State = walking gait stage, N = ?).

Task:

Preform preliminary study on similar works. Supervised learning vs. un-supervised learning (Walking gait phase detection). Sugested article: https://pure.manchester.ac.uk/ws/portalfiles/portal/213182312/FULL$_T$

Team$_m$*eeting*$_2$3.06.2023

meeting minutes Fri 23.06.2023

Participants: Dr. F , H and Martin

Unsupervised learning:

Input an array of data, possibly some kind of combination of two data sets, and allow the Learning algorithm to group the data in to clusters. The clusters will then work as outputs for the ANN.

Questions:

IMU measurements.

1. Budget? 2. MatLAb?

Send email to ROY.

IMU predictive system:s

https://www.nature.com/articles/s41467-020-19424-2.pdf?pdf=button

https://ieeexplore.ieee.org/document/10037621

https://fjfsdata01prod.blob.core.windows.net/articles/files/923164/pubmed-zip/.versions/1/.package-entries/fnbot-16-923164/fnbot-16-923164.pdf?sv=2018-03-28sr=bsig=jJjM33F6Jvprsr3P6E8yq4C

https://fjfsdata01prod.blob.core.windows.net/articles/files/923164/pubmed-zip/.versions/1/.package-entries/fnbot-16-923164/fnbot-16-923164.pdf?sv=2018-03-28sr=bsig=jJjM33F6Jvprsr3P6E8yq4C

https://pdf.sciencedirectassets.com/313346/1-s2.0-S2405896321X0002X/1-s2.0-S2405896320305656/main.pdf?X-Amz-Security-Token=IQoJb3JpZ2luX2VjEJL

Machine Learning:

Is it a good idea to take a data set we allready possess, and to some ML on this to get some actual work in? This could be used as a way of explaining different use of ANN's, and sort of a walk-through on feature-extraction.

Feature level based sensor fusion:

Page 85:

"Classification rules." How do we adjust these rules?

Meeting notes, meeting minutes

—

$\text{Team}_{m}eeting_{80}9_{2}023$

Data analytics / justification for application:

Skipped phases: Is it a mislabeled instance? Did the user skip a phase? Is the measurement faulty? (sensor failure).

LSTM: The LSTM uses memory, which in turn means that the last phases are known. Explain why the problem is still complex.

a) Plot the phases in a continous plot. (seaborn.pairplot (remeber hue) /

b) Count the number of the different phases. (Pandas Describe ++ look up analytic tools)

Fillippo:

From email $8_{92}3$[

https://inevitablehuman.com/the-best-way-to-identify-someone-is-by-the-way-they-walk-not-their-face/

https://arxiv.org/pdf/2203.04179.pdf: :text=Gait

https://www.sciencedirect.com/science/article/pii/S1574013721000721

https://www.ncbi.nlm.nih.gov/books/NBK557684/

https://recfaces.com/articles/what-is-gait-recognition

VERY IMPORTANT: https://www.frontiersin.org/articles/10.3389/frobt.2021.749274/full

]

Hamza:

Take a deep analysis of dataset. /

Skips? How often? / Done, in task a and b

Human activity reck.. /

LSTM:

Compare different model, justify your choice

Create new columns, that stores last 5 sensor readings.

Equipment:

Roy -

Thesis:

Investigate the possibility to identify different users based on the datasets. https://inevitablehuman.com/the-best-way-to-identify-someone-is-by-the-way-they-walk-not-their-face/

Make a table of different studies on gait phase recon.. Do a comparison, [technique, results, etc.] Make some sort of improvement / novel set-up

—- $\text{Team}_{m}eeting_{1}5_{0}9_{2}023$

Present work on the following task:

Data analytics / justification for application:

Skipped phases: Is it a mislabeled instance? Did the user skip a phase? Is the measurement faulty? (sensor failure).

LSTM: The LSTM uses memory, which in turn means that the last phases are known. Explain why the problem is still complex.

a) Plot the phases in a continous plot. (seaborn.pairplot (remeber hue) /

b) Count the number of the different phases. (Pandas Describe ++ look up analytic tools)

Comments on work, H and F:

Read, and write theory –> How should one deal with missing data in datasets?

Gait Phase identification:

Email Vu, ask for person id gait dataset.

Read Articles from Filiippo!

Paste from last meeting:

Fillippo:

From email $8_{92}3$[

https://inevitablehuman.com/the-best-way-to-identify-someone-is-by-the-way-they-walk-not-their-face/

https://arxiv.org/pdf/2203.04179.pdf: :text=Gait

https://www.sciencedirect.com/science/article/pii/S1574013721000721

https://www.ncbi.nlm.nih.gov/books/NBK557684/

https://recfaces.com/articles/what-is-gait-recognition

VERY IMPORTANT: https://www.frontiersin.org/articles/10.3389/frobt.2021.749274/full

]

—-

Team.meeting.4.10.2023

Email:

Gait phase labels based on acc_Z

Purchase sensor: https://www.moti.dk/shop

MOTI

Thesis outline:

1 . detection

2. identity

3. gait anomalies (FOG - Parkinson's)

—-

Project timeline

The following chart was used as a planning tool.

Figure D.1: Project timeline, GANT chart

Bibliography

[1] C. Bouvier et al. "Reduced and stable feature sets selection with random forest for neurons segmentation in histological images of macaque brain." In: *Scientific Reports* 11 (2021). URL: https://www.nature.com/articles/s41598-021-02344-6.

[2] Bosch Sensortec. *BMI160 - Inertial Measurement Unit*. 2024. URL: https://www.bosch-sensortec.com/products/motion-sensors/imus/bmi160/.

[3] Ottavio Calzone. *An Intuitive Explanation of LSTM*. https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c. Accessed: [Insert date here]. Year of publication.

[4] Rung-Ching Chen et al. "Selecting critical features for data classification based on machine learning methods." In: *Journal of Big Data* 7.1 (2020), p. 52. URL: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00327-4.

[5] DataCamp. *Curse of Dimensionality in Machine Learning*. Accessed: 2023-02-15. 2020. URL: https://www.datacamp.com/blog/curse-of-dimensionality-machine-learning.

[6] Scikit-Learn Developers. *sklearn.metrics.f1_score*. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html. Accessed: [Insert date here].

[7] Scikit-Learn Developers. *sklearn.model_selection.StratifiedKFold*. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html. Accessed: [Insert date here].

[8] EmotiBit. *EmotiBit - Wearable Emotional Intelligence*. EmotiBit Website. 2024. URL: https://www.emotibit.com/.

[9] EmotiBit. *EmotiBit GitHub Repository*. GitHub. 2024. URL: https://github.com/EmotiBit.

[10] *Feature Engineering*. https://domino.ai/data-science-dictionary/feature-engineering. Accessed: [access date]. 2024.

[11] M. Hartmann et al. "Effects of Juvenile Idiopathic Arthritis on Kinematics and Kinetics of the Lower Extremities Call for Consequences in Physical Activities Recommendations." In: *International Journal of Pediatrics* 2010 (2010), Article ID 835984. DOI: 10.1155/2010/835984. URL: https://www.researchgate.net/publication/46404365_Effects_of_Juvenile_Idiopathic_Arthritis_on_Kinematics_and_Kinetics_of_the_Lower_Extremities_Call_for_Consequences_in_Physical_Activities_Recommendations.

[12] IBM. *Convolutional Neural Networks*. https://www.ibm.com/topics/convolutional-neural-networks. Accessed: [Insert date here].

[13] IBM. *Neural Networks*. https://www.ibm.com/topics/neural-networks. Accessed: [Insert date here].

[14] First Author Javier Conte Alcaraz, Second Author Sanam Moghaddamnia, and Second Author Jürgen Peissig. "Efficiency of Deep Neural Networks for Joint Angle Modeling in Digital Gait Assessment." In: *EURASIP Journal on Advances in Signal Processing* 2020.1 (2020), p. 15. DOI: https://doi.org/10.1186/s13634-020-00715-1. URL: https://asp-eurasipjournals.springeropen.com/articles/10.1186/s13634-020-00715-1.

[15] Yunchuan Kong and Tianwei Yu. "A Deep Neural Network Model using Random Forest to Extract Feature Representation for Gene Expression Data Classification." In: *Scientific Reports* 8 (2018). URL: https://www.nature.com/articles/s41598-018-34833-6.

[16] Zhiqin Lu and Ruoxi Lu. "A Universal Approximation Theorem of Deep Neural Networks for Expressing Probability Distributions." In: *arXiv preprint* (2020). eprint: 2004.08867. URL: https://arxiv.org/abs/2004.08867.

[17] Sebastian O.H. Madgwick. "An efficient orientation filter for inertial and inertial/magnetic sensor arrays." In: (Apr. 2010). URL: https://www.samba.org/tridge/UAV/madgwick_internal_report.pdf.

[18] First Author Marion Mundt et al. "Estimation of Gait Mechanics Based on Simulated and Measured IMU Data Using an Artificial Neural Network." In: *Frontiers in Bioengineering and Biotechnology* 8 (2020), p. 41. DOI: https://doi.org/10.3389/fbioe.2020.00041.

[19] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[20] Dr. Barak Or. *Temporal Convolutional Networks: The Next Revolution for Time Series*. Published on Aug 12, 2020, 6 min read. 2020. URL: https://medium.com/metaor-artificial-intelligence/temporal-convolutional-networks-the-next-revolution-for-time-series-8990af826567.

[21] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors." In: *nature* 323.6088 (1986), pp. 533–536.

[22] Milind Sahay. *Neural Networks and the Universal Approximation Theorem*. Towards Data Science. Available online at https://towardsdatascience.com/neural-networks-and-the-universal-approximation-theorem-8a389a33d30a.

[23] scikit-learn developers. *Importance of Feature Scaling*. [Online; accessed 15-Feb-2023]. 2022. URL: https://scikit-learn.org/stable/auto_examples/preprocessing/plot_scaling_importance.html.

[24] TensorFlow. *Save, serialize, and export models | TensorFlow Core*. https://www.tensorflow.org/guide/keras/serialization_and_saving. Accessed: [Insert date here]. 2021.

[25] Luke Topham et al. *Limb Joint Movement Gait Dataset from IMU sensor*. June 2022. DOI: 10.21227/b5tp-y175. URL: https://dx.doi.org/10.21227/b5tp-y175.

[26] Huong Thi Thu Vu et al. "ED-FNN: A New Deep Learning Algorithm to Detect Percentage of the Gait Cycle for Powered Prostheses." In: *Sensors* 18.7 (2018), p. 2389. DOI: https://doi.org/10.3390/s18072389. URL: https://www.mdpi.com/1424-8220/18/7/2389.

[27] Tishya A.L. Wren et al. "Use of a Patella Marker to Improve Tracking of Dynamic Hip Rotation Range of Motion." In: *Gait Posture* 27 (2008), pp. 530–534. DOI: 10.1016/j.gaitpost.2007.07.006. URL: https://www.researchgate.net/publication/6135730_Use_of_a_patella_marker_to_improve_tracking_of_dynamic_hip_rotation_range_of_motion.

[28] Xiang Zhang, Shiqi Yu, and Yunhao Liu. "IMU-Based Gait Recognition Using Convolutional Neural Networks and Multi-Sensor Fusion." In: *Sensors* 17.12 (2017), p. 2735. DOI: https://doi.org/10.3390/s17122735. URL: https://www.mdpi.com/1424-8220/17/12/2735.

[29] Xiang Zhang, Shiqi Yu, and Yunhao Liu. "IMU-Based Gait Recognition Using Convolutional Neural Networks and Multi-Sensor Fusion." In: *Sensors* 17.12 (2017), p. 2735. DOI: 10.3390/s17122735. URL: https://www.mdpi.com/1424-8220/17/12/2735.