UNIVERSITY OF AGDER

Classification of Diabetes and Cardiac Arrhythmia using Deep Learning

Micheal Dutt

SUPERVISORS
Morten Goodwin
Vimala Nunavath

Master's Thesis
University of Agder, 2018
Faculty of Engineering and Science
Department of ICT

UiA
University of Agder
Master's thesis

Faculty of Engineering and Science
Department of ICT

Abstract

Deep Learning (DL) is a research area that has flourished significantly in the recent years and has shown remarkable potential for artificial intelligence in the field of medical applications. The reasons for success are the ability of DL algorithms to model high-level abstractions in the data by using automatic feature extraction property as well as significant amount of medical data that is available for training these algorithms. DL algorithms can learn features from a large volume of healthcare data, and then use the procured insights to assist clinical practice. We have implement DL algorithm for the classification of two diseases in the medical domain: Diabetes and Cardiac Arrhythmia.

Diabetes is often considered as one of the world's major health problems according to the World Health Organization. Recent surveys indicate that there is an increase in the number of diabetic patients resulting in the increase in serious complications such as heart attacks and deaths. This thesis presents a Multi-Layer Feed Forward Neural Networks (MLFNN) for the classification of diabetes on publicly available Pima Indian Diabetes (PID) dataset. A series of experiments are conducted on this dataset with variation in learning algorithms, activation units, techniques to handle missing data and their impact on classification accuracy have been discussed. Finally, the results are compared with other machine learning algorithms like Naïve Bayes, Random Forest, and Logistic Regression. The achieved classification accuracy by MLFNN (82.5%) is the best of all the other classifiers.

The term arrhythmia refers to any variation in the usual sequence of the heartbeat. There are many types of cardiac arrhythmia ranging in severity, including Premature Atrial Contractions (PACs), Atrial Fibrillation, and Premature Ventricular Contractions (PVCs). This thesis focuses on the use of DL algorithms: Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) to classify arrhythmia with minimum possible data pre-processing on MIT-BIH Arrhythmia Database (MIT dataset). Furthermore, we study the influence of different hyperparameters like L2 regularization and number of epochs on the classification accuracy of LSTM. We achieved a classification accuracy of 99.19% and 98.40% with CNN and LSTM models respectively. From our research, we believe that CNN model can assist the doctors in the classification of arrhythmia.

Preface

This thesis is based on the work performed for the IKT591 final year Master's thesis project which corresponds to 60 ECTS credits as part of the Master program in Information and Communication Technology at the University of Agder. The thesis work started from August 2017 and ended in June 2018. As a result of this study, a paper based on the 1st research question has been accepted by Eleventh International Conference on Developments in e-Systems Engineering - DeSE2018.

I would like to express my sincere thanks to my supervisor Associate Professor Morten Goodwin for his patient guidance and support throughout the project. His supervision leads me to open the door of doing research. I also would like to express my gratitude to Vimala Numavath, who helped me to understand the research questions better. Finally, I would like to thank all my friends and family for their help and encouragement during my Master studies.

Micheal Dutt

Table of Contents

List of Figures

List of Tables

Part I

Research Overview

Chapter 1

Introduction

Detecting a medical anomaly is usually considered to be a complex task and comes under the domain of medical experts and physicians. There are specific processes to detect medical abnormalities that are carried out by physicians. These processes tends to be time consuming and subjective [73] [60]. Classification of diabetes and arrhythmia are two such pathological cases, which usually requires many physicians with a wide range of experience in the respective domains.

Diabetes is one of the world's major health problems according to the World Health Organization report(WHO) [8]. According to the report, there were around 422 million in 2014 living with diabetes. This number had increased dramatically from 1980 when approximately 108 million people were suffering from this disease. Diabetes alone has caused about 1.5 million deaths in 2012. The cases of diabetes have grown faster in the low and middle-income countries as compared to high-income countries over the past decade. Diabetes can often be classified as type 1 or type 2. Type 1 diabetes is caused by β-cell destruction, usually leading to absolute insulin deficiency. Type 2 diabetes is due to progressive insulin secretory defect on the background of insulin deficiency [23]. Generally, when the amount of glucose in the blood is high, the performance in different body organ gets affected. If the disease in such condition is not timely classified, then it may lead to heart attacks, kidney failures and blindness [37] [47]. In many cases, the classification is generally based on patient test results.Thus, for classifying if a person is diabetic or non-diabetic is a complex task requiring high skills and knowledge.

Cardiac arrhythmia is also a significant healthcare problem in the developed world [25]. The term "arrhythmia" refers to any variation in the usual sequence of electrical impulses. Sometimes this impulse can be too fast, too slow, or unpredictable which can cause the heart to beat in the same order. If the heart does not beat correctly, it can not pump the blood effectively. If the blood is not pumped effectively then the other parts of the body like the lungs, the brain can not function properly. Then this body parts may shut down or get damaged causing a life-threatening situation for the patient. The American Heart Association 2017 statistical report indicates that the atrial fibrillation is the most common cardiac rhythm disorder [25]. It has an approximate 25% lifetime incidence and annual costs of around 26 billion dollars, whereas sudden cardiac affects fewer but has more dangerous consequences. Early classification and medical care can significantly reduce the problems of patients [32]. With technological advancements and a lot of research being done in the field of Artificial Intelligence (AI), it can be made possible to classify the disease in the early stages.

AI techniques are bringing a paradigm shift to healthcare [44]. The recent development of the big data analytics and availability of large healthcare data has made these AI techniques a success in the healthcare sector. Deep learning (DL), which is a sub-domain of AI, has been an active area of research in this sector. DL can save time for physicians and provide balanced, repeatable results by automating the manual processes used by them. Additionally, there is a significant amount of data in medicine combination of which with a small sample size of pathological cases makes essential use of deep learning techniques for diagnosing and classifying the disease [49]. Deep learning techniques can prove to be useful computer-based tool that model the expert's behaviour and can improve classification accuracy and can become universal standard among medical practitioners.

This thesis explores the use of DL for classification of two medical anomalies: Diabetes and Arrhythmia. We have developed a Multi-Layer Feed Forward Neural Network (MLFNN) for classification of diabetes, and the dataset that we have used is Pima Indian Diabetes (PID) dataset. For the classification of Arrhythmia, we have developed a Convolutional Neural Network (CNN) and a Long Short-Terms Memory (LSTM) network. We have used MIT-BIH Arrhythmia Database (MIT dataset) for this task.

1.1 Motivation and Problem Statement

Usually, for classifying if a person is diabetic or non-diabetic, several tests are required, out of which some are expensive and time-consuming [60] [73]. For categorizing cardiac arrhythmia, there is a need for careful analysis of ECG signal. In both the cases, physicians are required to do the task. In some cases, regardless of their expertise and experience, there can be some errors [44]. Few of such mistakes that a physician can make are anchoring bias or availability bias. In anchoring bias, the physicians are stuck on an initial impression of the problem. There can also be some cases where physicians take some decisions based on some recent incidents which are called availability bias. Sometimes, the number of such physicians is not enough, or there can be some disagreement among the experts. In a field which requires so much of human expertise and concentration, sometimes it becomes prone to some errors which can prove to be costly for patients and even life-threatening.

In this thesis, deep learning models are built for the classification of diabetes and cardiac arrhythmia at high accuracy and least error rate. We study the impact of different activation functions and learning algorithms on MLFNN model on the PID dataset. Furthermore, we evaluate the performance of CNN and LSTM on the time series MIT dataset.

1.2 Research Questions

In this section, we discuss the research questions which this thesis make an effort to answer. The first research question belongs to the classification of diabetes, whereas the second research question is related to the classification of arrhythmia.

1. Does MLFNN provide better classification accuracy when compared with other machine learning algorithms for classifying diabetes on the PID dataset?

   To answer this research question, in the thesis an MLFNN model is developed which can classify diabetes with higher accuracy rate than other machine learning algorithms like Naïve Bayes (NB), Random Forest (RF) and Logistic Regression (LR). The PID dataset has been used to create MLFNN, which has eight input features and one output (Whether diabetic or non-diabetic).

1.2.1 Diabetes

Along with the research question mentioned above, we have formulated subordinated research questions for the PID dataset.

(a) Which technique will perform better to handle the missing data and provide the highest accuracy in MLFNN for the PID dataset?

We use three techniques to handle the missing data to answer this question. There are a large number of values that are missing from the attributes in the PID dataset. In the first technique, we remove all the rows that have a missing value. In second, we replace the missing values with zero. In the third technique, we replace the missing value with the mean of all the values of the specific attribute.

(b) Which activation function provide higher classification accuracy in MLFNN for the PID dataset?

The activation functions chosen to answer this question are Rectified Linear Unit (ReLU), Leaky ReLU, Exponential Linear Unit (ELU) and Scaled Exponential Linear Units (SELU). We use all these activation functions in a pre-defined architecture and compare the training accuracy, testing accuracy and Mean Squared Error (For both testing and training).

(c) Which learning algorithm provide higher classification accuracy in MLFNN for the PID dataset?

To answer this question, we use two learning algorithms: Stochastic Gradient Descent (SGD) and ADAM. Same performance metrics are used to compare the classification performance.

2. Is it possible to develop different deep learning model for automated arrhythmia detection?

   To answer this question, we develop two deep learning models. We design a CNN and LSTM networks for classification of cardiac arrhythmia. Both the models use the time series data MIT dataset.

1.2.2 Arrhythmia

Along with the research question mentioned above, we have formulated subordinated research question for the MIT dataset.

(a) Will LSTM perform better than the CNN model in classifying arrhythmia for the MIT dataset?

   We answer this research question by evaluating the performance of CNN and LSTM model on classifying cardiac arrhythmia.

1.3 Solution Overview

The first task is to develop a MLFNN model which can classify if a person has diabetes or not with higher accuracy. We then compare the performance of MLFNN with Naïve Bayes, Logistic Regression and Random Forests algorithm. We also determine how changing the activation units and learning algorithms impact the testing and training accuracies. Also, the PID dataset suffers from missing data. We continue the process further by implementing different techniques to handle the missing data. We apply three methods, replacing the missing value with zero, mean, and removing the missing values from the dataset.

In the second task, we classify the different type of cardiac arrhythmia by using CNN and LSTM models on the MIT dataset. We compare the performance of the models developed in this research with other models developed by researchers for the same database. We try to achieve higher accuracy by doing minimum data pre-processing. We try to evaluate the performance of both CNN and LSTM models. Furthermore, we study the influence of different hyperparameters like L2 regularization and number of epochs on the classification accuracy of LSTM.

1.4 Contribution

The main contribution of the thesis are:

- A MLFNN to classify diabetes with higher accuracy than some other models that have been implemented before.
- Proficient architectural considerations while designing the MLFNN model to improve the accuracy.
- A CNN and LSTM network to classify arrhythmia with higher accuracy by doing minimum data pre-processing as compared with other models.
- Evaluation of CNN and LSTM models on the time series MIT dataset.

The focus of this thesis is to answer all the research questions. This thesis explain how the architectural design plays a crucial role in improving the accuracy of the model and needs to be chosen with careful experimentation analysis and according to the dataset. We evaluate the performance of two deep learning models: CNN and LSTM.

1.5 Thesis outline

The rest of the thesis is structured as follows:

- **Chapter 2** provides background research for Artificial Neural Networks (2.1), Multi Layer Feed Forward Neural Network (2.2), Convolutional Neural Networks (2.3) and Recurrent Neural Networks (2.4).
- **Chapter 3** presents the literature review for the previous work done in the field of machine learning for diagnosing or classification of different diseases, Diabetes (3.1) and Arrhythmia (3.2).

- **Chapter 4** introduces the dataset that we have used for the classification of Diabetes (4.1) and Arrhythmia (4.2).
- **Chapter 5** explains the pre-processing techniques that we have used before feeding the data into neural network for the classification of Diabetes (5.1) and Arrhythmia (5.2).
- **Chapter 6** provides the details of the experimental setup that we have implemented for the classification of Diabetes (5.1) and Arrhythmia (5.2).
- **Chapter 7** shows the experimental results we have achieved and the discussion regarding the results.
- **Chapter 8** concludes the thesis and provide the summary of the work done in the thesis. We have also mentioned potential work that we will apply to our research in future to achieve more desirable results.

Chapter 2

Background

Deep Learning (DL) is the sub-domain of Machine Learning (ML) which is the sub-domain of Artificial Intelligence (AI). DL algorithms can be implemented to solve any supervised learning or unsupervised learning problems. DL algorithms try to model high-level abstractions in data by using automatic feature extraction property [11]. DL uses feature hierarchy where each layer trains on the distinct set of features that are provided to it as a previous layer output. The deep layers can recognize the sophisticated features in the data as they aggregate and then again recombine the features they receive from the last layer. This makes DL algorithms suitable for handling the large dataset [10]. DL algorithms can discover the pattern and structures within the datasets which are not either categorized or have any structure. That makes DL algorithms a potent tool in today's world as the majority of the databases that are available are either unstructured or unlabelled.

2.1 Artificial Neural Network

The structure of Artificial Neural Networks (ANNs) is inspired from the structure of the human brain [62]. In the human brain, there is a network of electro chemical cells called neurons which takes the input and process it and then decide whether they will send any output or not. In this way, this whole network of neurons works in harmony to determine the output. Similarly, ANNs are comprised of neurons which are arranged in the layers. These neurons take the input from either input layer or the previous layer neurons, process them and then decides what output to send to either the next layer or to the output layer [2]. How neurons take the decision is an important design principle of ANNs. These neurons are connected with at least one another neuron in the network. The connection between them has some value assigned to it which is known as weight. Based on connection architecture the ANNs can be either classified into feed forward networks (in which there are no loops) and the recurrent neural network (in which there are loops because of feedback connections). This weight decides the importance of the connection between the neurons. In the starting, these weights are just randomly assigned to all the connections. In the supervised learning, we train the network on labelled data. So, the weights in such networks are adjusted in such a way that the difference between the actual output and the desired output is least. In unsupervised learning, the training data has no labels. In such problems, the neural network itself extract some facts or patterns from the data during the iterations. Then neural network becomes stable after specific iterations. In this way, ANNs can be implemented to solve any supervised learning, unsupervised learning and even semi-supervised learning problems [1].

2.2 Multi-Layer FeedForward Neural Networks

The multi-layer feed forward neural network is a fully connected network consisting of an input layer, one or more hidden layer(s), and the output layer [66]. The general MLFNN network is illustrated in Figure 2.1.

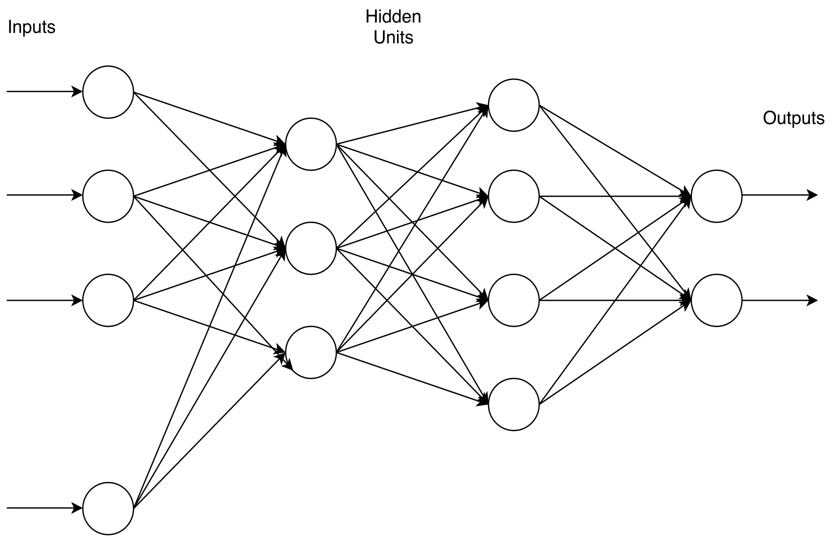

Figure 2.1: A General Multi-layer Feed Forward Neural Network with an Input Layer, 2 Hidden layers and an Output Layer.

The internal layers are called hidden layers because they are hidden from the outside world. They receive input from the internal processing units, process them and send the output to internal processing units in the next hidden layer. The real-valued input feature vector is fed in fan-out arrangement to all the neurons in the first hidden layer. Each neuron in the first hidden layer is connected with all the neurons in the next hidden layer. The weight coefficient characterizes the connection between two neurons. This weight reflects the importance of the connection in the neural network. Activation function determines the output of the neurons. The whole mechanism is illustrated in the Figure 2.2.

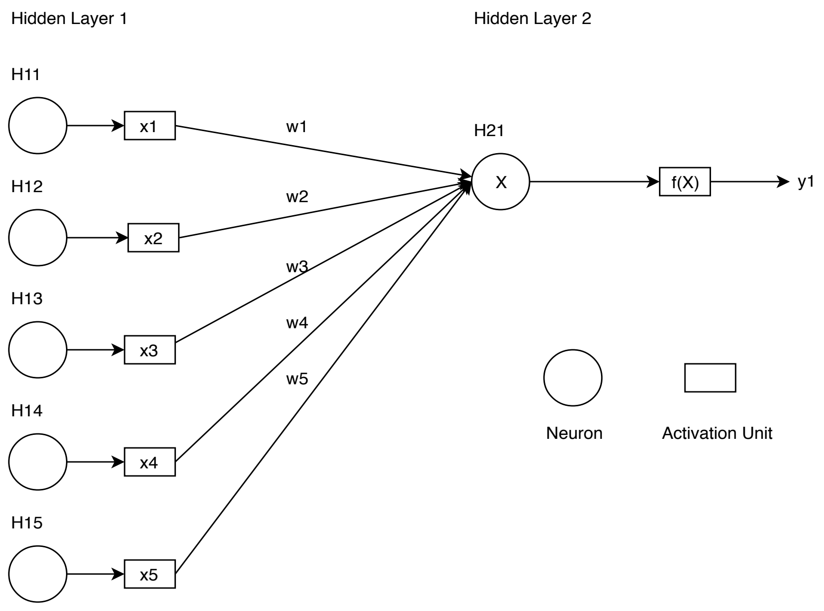

Figure 2.2: An example showing connections between the neurons and activation units

Let us see how the connections between neuron work. In Figure 2.2, we are considering two hidden layers: Hidden Layer 1 and Hidden Layer 2. First, hidden layer has five neurons, and we are considering only one neuron H21 in Hidden layer 2 to know how connections work. Firstly, Neuron H21 will compute X. Neurons from Hidden layer will pass their internal output values, which is multiplied with weight coefficient (the measure of the importance of the connection) as can be seen in equation 2.1 [7].

$$X = (w1x1 * w2x2 * w3x3 * w4x4 * w5x5) \tag{2.1}$$

This is the internal processing which neuron H21 will. After this process, X will pass through activation function, and the internal output of H21 will be created y1 as can be seen in equation 2.2.

$$y1 = f(X) + bias \tag{2.2}$$

2.2.1 Backpropagation Algorithm

In this subsection, we go through the concept of backpropagation. When the artificial neural networks were first implemented, they were very slow to train and does not produce the effective results. To sort out this problem, the concept of backpropagation was introduced [3]. The backpropagation algorithm comprises of two phases: propagation phase and the weight update phase.

When a neural network receives an input, there is generally random number weight that is assigned in the hidden layers. The neural network produces some output based on these weights. This is called the propagation phase. This output is then compared with the original output using loss function and the error is calculated. This error is then propagated backwards, and the weights are again updated in such a way that this error is reduced in the next round [9]. This updating of weight to reduce the error is called a weight update phase. These rounds are repeated until the error rate reduces to minimum values.

In this way, the backpropagation algorithm calculates the gradient of loss function using the error values with respect to the weight coefficients. After that, the gradients are updated in such a way that loss function is minimized in the optimization phase.

2.2.2 Activation Functions

Activation functions play a vital role while designing a neural network. To understand activation function we have to understand what artificial neurons do. As discussed in equation 2.2, it calculates the weighted sum of all the inputs it receives and adds bias and then decide whether it should fire or not [16]. So, this value can range between $+\infty$ to $-\infty$. Thus, activation functions bound this value produced by neurons and decide if the external connection is fired or not. Since artificial neurons are developed from the idea of biological inspiration brain neurons., we are using term fire which means neurons is active and sending a signal or in artificial neuron we say value. There is the number of activation functions that have been proposed to date.

We briefly discuss the activation functions we have used in this thesis for the experimentation. Below, we introduce the four kinds of activation unit: rectified linear unit (ReLU), leaky rectified linear (Leaky ReLU), exponential linear unit (ELU) and scaled exponential linear unit (SELU).

ReLU - The Rectified Linear Unit is first used in Restricted Boltzmann Machines [54]. It is the simplest nonlinear activation function and is defined as:

$$f(x) = \begin{cases} 0 & \text{for} \quad \text{x} < 0 \\ \text{x} & \text{for} \quad \text{x} \geqslant 0 \end{cases}$$

ReLU activation function gives an output x if x is greater then zero and 0 if x is less than zero. ReLU is considered inexpensive function since there are not any exponential values or other mathematical values to solve. So, it converges faster. But, when there are negative values, then ReLU activation function will bound it to zero and the neurons will not send any output values and will be considered as dead. In this way, ReLU units can die during the training process. ReLU activation can be seen in Figure 2.3.

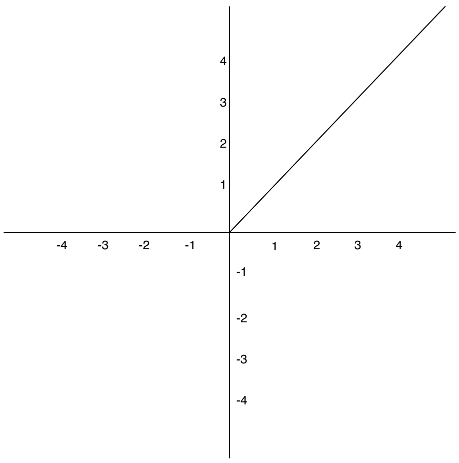

Figure 2.3: Graphical representation of ReLU Activation Function

Leaky ReLU activation was first introduced in acoustic model [51]. These are one attempt to fix the dying ReLU problem. Mathematically, it is defined as:

$$f(x) = \begin{cases} \alpha \text{x} & \text{for} \quad \text{x} < 0 \\ \text{x} & \text{for} \quad \text{x} \geqslant 0 \end{cases}$$

where $\alpha = 0.01$. Instead of function being zero for x less then zero, leaky ReLU have a small slope. Figure 2.4 shows the graphical representation of Leaky ReLU activation function.

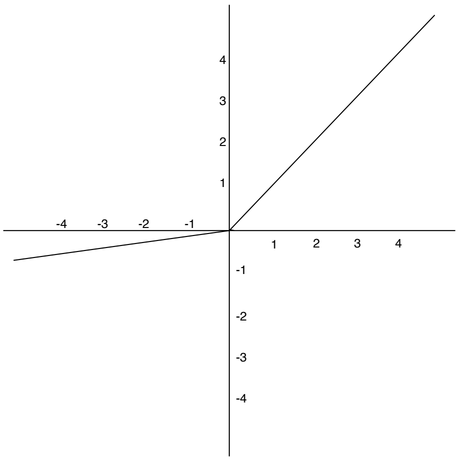

Figure 2.4: Graphical representation Leaky ReLU Activation Function

ELU - Exponential Linear Unit [30] with $0 < \alpha$ is

$$f(x) = \begin{cases} \alpha(\text{exp(x)-1}) & \text{for} \quad \text{x} < 0 \\ \text{x} & \text{for} \quad \text{x} \geqslant 0 \end{cases}$$

The hyperparameter α in ELU controls the saturation of the negative inputs. ELU makes the learning process faster in deep neural networks as compared with other activation units. ELU do not suffer from the dying or vanishing gradient problem as it has negative values which allow the mean activation to move closer to zero as can be seen in batch normalization. It does so at a lower computational cost. ELU does not quantitatively model the degree of absence of input but codes the degree of presence of specific characteristics in the data.

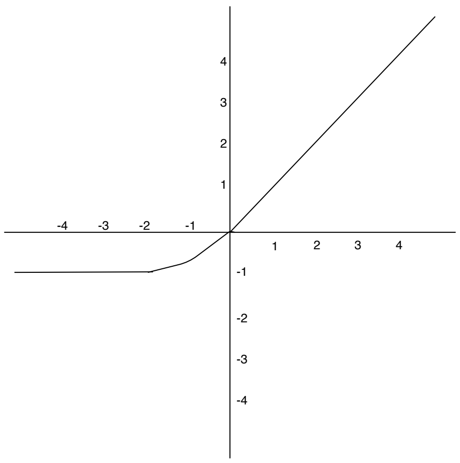

Figure 2.5: Graphical representation of ELU Activation Function

SELU - The SELU activation function [48] is given by:

$$f(x) = \lambda \begin{cases} \alpha(\text{exp(x)-1}) & \text{for} \quad \text{x} < 0 \\ \text{x} & \text{for} \quad \text{x} \geqslant 0 \end{cases}$$

SELU allows constructing a mapping with properties that lead to Self-Normalizing Neural Networks (SNNs). SNNs keeps normalization of activation's when propagating through layers. The activation's of the neural network is considered to be normalized if both their mean and their variance across samples are within predefined intervals. In SNNs the activation of the neuron automatically converges toward the zero mean and unit variance. It propagates through many layers even in the presence of noise and perturbation. In this way, SNN allows training deep neural networks with many layers by employing this powerful regularization technique and making the learning highly robust. If some activations are not close to zero, there is an upper and lower bound in the variance which makes it impossible to counter the dying and exploding gradients problems.

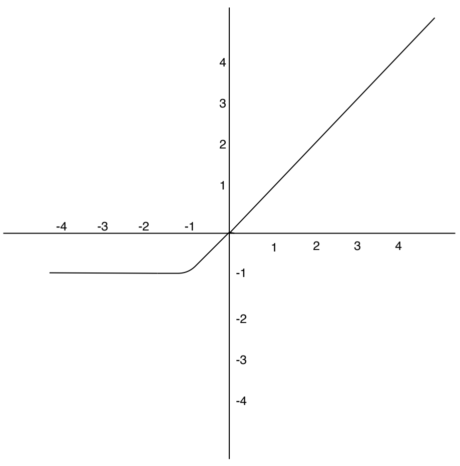

Figure 2.6: Graphical representation of SELU Activation Function

2.2.3 Learning Algorithms

In this section, we introduce two different types of learning algorithms: *Stochastic Gradient Descent (SGD) and Adam.* Both learning algorithms are explained below.

Stochastic Gradient Descent is probably the most used learning algorithm for machine learning and particularly for deep learning [36]. The standard gradient descent algorithms update the parameters approximated by evaluating the cost and gradient over the full training set. The SGD updates and computes the gradient of parameters using single or few training examples. Equation 2.3 gives the parameter update:

$$[P = P - LR * gradients] \tag{2.3}$$

P stands for Parameters, and LR stands for Learning Rate.

The benefit of updating the parameters based on few training examples is, it reduces the variance in the parameter update and leads to a stable convergence. The crucial setting of SGD is learning rate which must adjust with a lot of trial and error. In this thesis, we have used default learning rate which is 0.01.

Adam is an adaptive learning rate optimization algorithm which derives its name from the phrase “adaptive moments” [36]. Adam algorithm is based on the estimation of 1st and 2nd order moments. The algorithm estimates “the first moment” as the mean and the “second moment” as variance. So the update rule Adam is given by equation 2.4:

$$[P = P - LR * Means/Sqrt(Variance)] \tag{2.4}$$

Here, P stands for Parameters and LR stands for Learning Rate.

So if the variance of the gradient is high, it becomes unclear how parameter should be changed, so algorithm chooses the small step size in update rule. If the variance is low, the algorithm takes a more significant step.

2.2.4 Loss Functions

In most networks, the error is the difference in the desired output and the predicted output [5] and is given by equation 2.5.

$$J(W) = DesiredOutput - predictedOutput \tag{2.5}$$

The function which computes this error is known as Loss Function J(.). Different loss functions affect the models in different ways. They have a considerable impact on the performance of the model. MSE (mean squared error) is one of the most famous loss function, which calculates the square of the difference between desired value and the predicted value. There are different loss functions which we can use for classification and regression tasks [12].

Cross-entropy is most commonly used for the binary classification problem. Equation 2.6 computes Cross-entropy:

$$H_{y'}(y) = -\sum_i y'_i \log(y_i) \tag{2.6}$$

Cross-entropy measures the divergence between the two probabilities distribution. If the cross-entropy is significant, the difference between the two distribution is big. If the cross-entropy is small, the distributions are somewhat similar. Cross-entropy indicates the distance between what model believes output distribution should be, and what actual distribution is.

2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are the computationally efficient neural network which can be used to process any grid-like topology data. Some examples of such data are time series data which can be imagined as 1-D grid taking samples at the regular interval of time and images which can be imagined as 2-D grids of the pixel [36]. CNN is a mighty and efficient model which can achieve high accuracy by automatic feature extraction.

The architecture of CNN is modelled after the part of the brain where the visual images are processed called the visual cortex [4]. Neurons in the brain

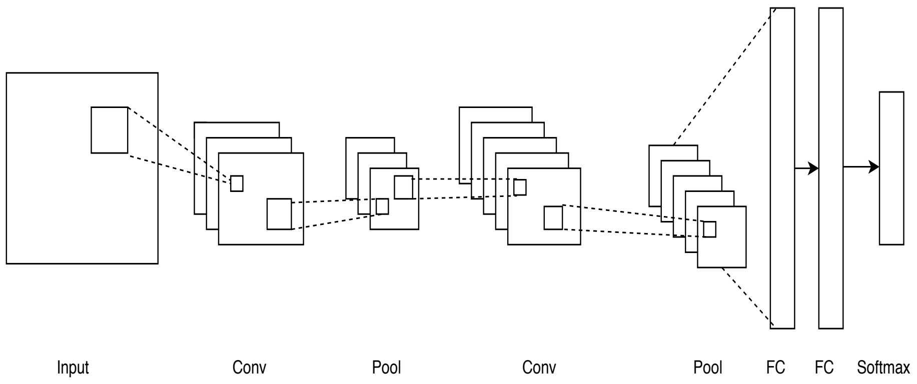

Figure 2.7: Basic Architecture of Convolutional Neural Architecture

process images only in the layers of increasing complexity. Neurons in the first layers only fire or send output when they are looking at a specific shape. Suppose some neurons just fire when they see curves and some when they look at straight lines. Neurons at the higher level recognizes the pattern of edges and colours. In the figure 2.7, we can see there is an input of any grid-like topology data. After that, there are several convolutions (Conv), and Pooling (Pool) layers followed by several fully connected layers. In case of multiclass classification, SoftMax is the output used.

2.3.1 Convolutional Layers

Convolution layer is the main feature of CNN. Convolution Layer often comprised of filter and feature maps [4]. Convolution is applied to the input using the filter to produce a feature map. Filter constitutes the parameter called weights and comes in specific shape matrix. We explain this process by taking an example.

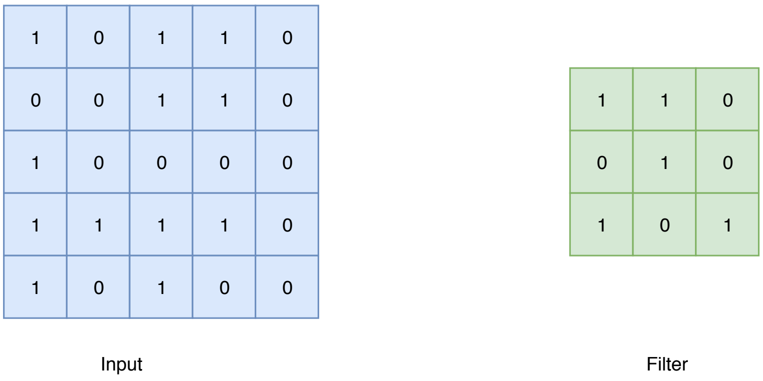

Figure 2.8: An example of Input and Filter

In Figure 2.8, there is input to convolution layer on the left side and on the right side is the filter. Due to the shape of the filter, it is known as 3x3 convolution. During the convolution operation this filter slides over the input, and at every point, we do the element-wise matrix multiplication. In the end, we sum the result, and this sum goes to feature map.

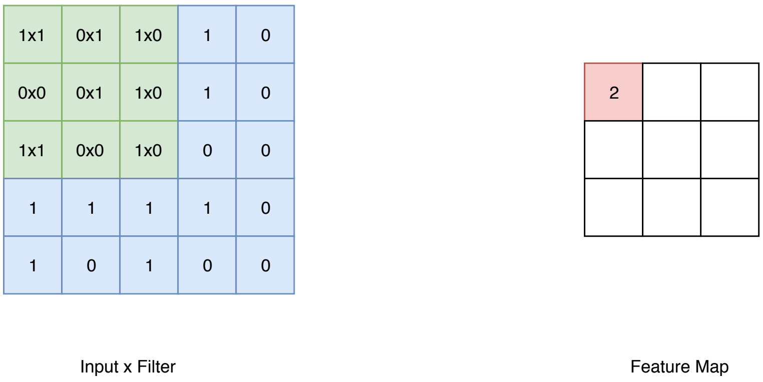

Figure 2.9: An example of building of Feature Map (Step 1)

In figure 2.9, the green area shows where the convolution operation takes place. The convolution operation is taking place at the top left, and the output is “2”. This result can be seen in the feature map also on the right side. In the same way, this convolution operation repeats by sliding this window to the right side by one step as can be seen in the figure 2.10. This value is again updated in the feature map. This process of sliding the filter over the input continues until the complete feature map is generated.

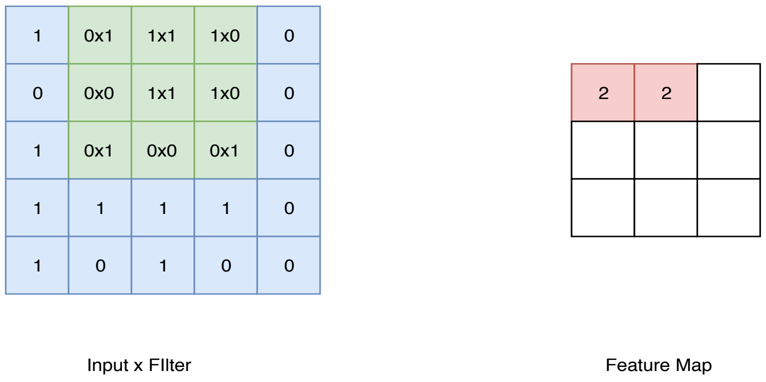

Figure 2.10: An example of building of Feature Map (Step 2)

In this example convolution operation are shown in 2D. In reality, convolutional layer takes the 3D volume of input and transforms it into the 3D volume of output. In addition to height and weight, there is depth also which corresponds to the colour channel (RGB).

2.3.2 Pooling Layers

After convolution operation, pooling is used to decrease the dimensionality by reducing the number of parameters [38]. Pooling layers down samples the feature maps independently by reducing the height and width, but it keeps the depth intact which means the depth is not changed. Since convolutional layer often generates a large number of feature maps making it slow to process by increasing the training time. Pooling layer thereby shortens the training time by reducing the dimensionality. Another ask of pooling is that it often helps to combat overfitting. In overfitting, the model generally learns the training data so well that it performs excellently on training data but fails to produce the same result on testing data.

In contrast with convolution operation, pooling operation has no parameters. The most common pooling operation that has proven to very useful is max pooling operation. It takes maximum value by sliding a window over the input. Max Pooling is further explained by taking an example where input 4x4 matrix and max-pooling using 2x2 window shown in Figure 2.11.

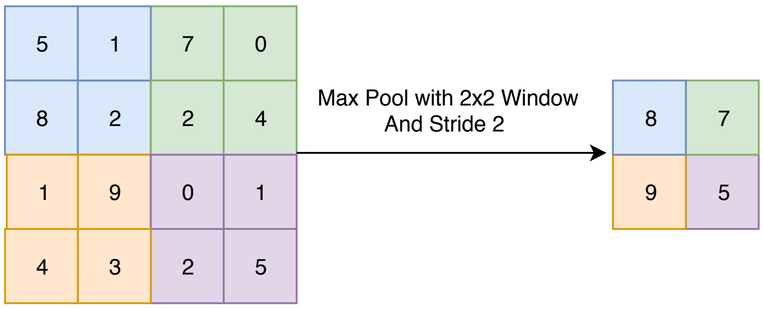

Figure 2.11: An example of Max Pooling Operation

2.3.3 Dropout Layers

Another layer that is not necessarily one of the primary layers, but can be categorized as important, is the Dropout layer [64]. Dropout is a common regularization technique for deep neural networks. It is used to combat overfitting by temporarily dropping some neurons during training time, at each iteration.

2.3.4 Fully Connected Layers

We add a fully connected layer(s) at the end of the CNN architecture. The convolutional and pooling layer do the feature extraction in the dataset, and the output they produce is in the form of 3D. Since it has three different axes each representing height, width, and the depth. The full connected layers are used to convert these 3D outputs into 1D and process is known as flattening [45].

2.4 Recurrent Neural Networks

Recurrent neural networks (RNN) [36] have become a prevalent choice for solving sequence prediction or time series problems. RNNs have been successfully implemented for the various task such as language modelling, learning word embeddings, online hand-written recognition and speech recognition. In RNNs, there are networks with loops, allowing information to persist. RNN takes decisions by considering the current input and from the previous inputs which it has received during the past and is in the memory. A normal RNN has a short-term memory which can store the copy of output it produces and loopback in into the network. Figure 2.12 illustrates the flow of information in RNN.

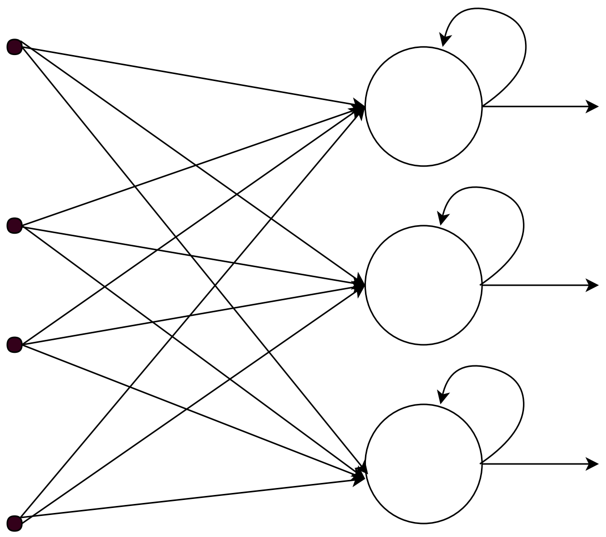

Figure 2.12: An example of Recurrent Neural Network (RNN)

In figure 2.13, we can see a chain like structure when we unroll RNN. This structure helps us to understand the concept the Backpropagation Through Time (BPTT). In this structure the error is propagated in more than two layers, to capture more past information. There are several timesteps each having input, output, and a copy of network that shares same parameters. Then the error is calculated at each time-step. After that, we roll the RNN, and the weight is updated.

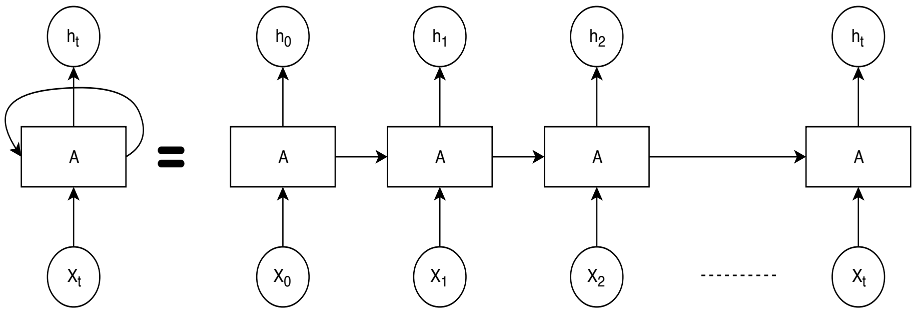

Figure 2.13: An example of an Unrolled RNN

2.4.1 Long Short Term Memory (LSTM) Networks

LSTMs are the extension of RNNs that are capable of learning long-term dependencies [42]. They have the extended memory which enables them to remember their input over an extended period. In an LSTM memory, it can read, write, and delete the information. This memory can be considered as gated cells, where gated means that the cell takes the decision. The decision of storing or removing the data depends on the importance it assigns to information. This assignment is done based on the weights learned by the algorithms. All RNNs are in shape of the chain of repeating modules of the neural network. As can be seen in figure 2.14, In standard RNN, the replicated module has a straightforward structure such as single tanh layer [17].

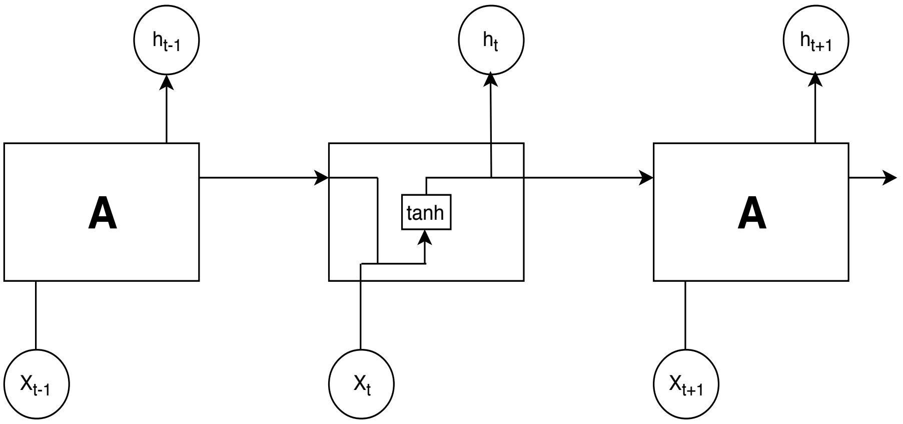

Figure 2.14: The repeating module in a standard RNN

LSTM repeating module does not have single neural network layer. There are four layers interacting in a very special way. Figure 2.15 shows the repeating module in an LSTM with four interacting layers.

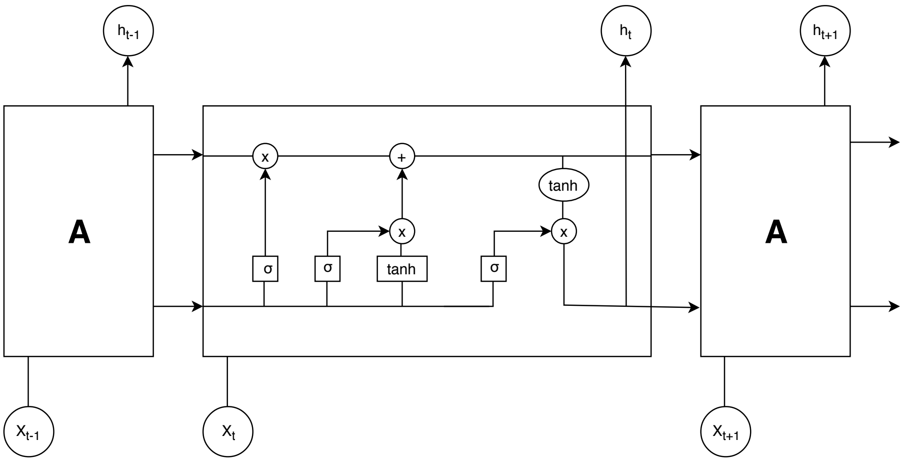

Figure 2.15: The repeating module in LSTM

The horizontal line through the top runs straight down the entire chain has only a few minor linear interactions. Information can flow very easily along it without getting changed. The structure of LSTM is constructed in such a way that it can add or remove the information to the cell state. Gates carefully regulate it. The first step by the LSTM is to decide which information it is going to cast out from the cell state. This is the decision taken by the sigmoid layer called “forget gate layer”. It gives output 0 or 1 where 0 corresponds to completely get rid of the information and 1 to keep the information completely.

The next step is to decide which information to store in the cell state. This step has two parts. The decision to create an update to the state is taken by the combination of the sigmoid layer called “input gate layer” and the tanh layer. Sigmoid layer contributes by deciding which values to update and tanh layer performs the creation of a new values vector which can be added to the state.

Finally, the output will be based on filtered cell state. First, a sigmoid layer will decide which parts of the cell state are going to output. Then the cell state is passed from the tanh, which pushes the values between -1 and 1. Then it is multiplied by the output of the sigmoid gate and only decided parts will go as output [6]

Chapter 3

Literature Review

There is significant research that is going on in the areas of developing machine learning algorithms for medical applications. In this chapter, we discuss some of the algorithms that have been developed by researchers in this field. In this thesis, our focus is to design such machine learning algorithms which can classify diabetes and cardiac arrhythmia at a high accuracy rate. So first we discuss some research done in the field of medical science, and after that, we discuss some of the algorithms which have been designed for diabetes and cardiac arrhythmia.

The author in [58] used Oxford Parkinson's Disease Detection Dataset for classification of Parkinson's disease (PD) using Multi-Layer Feed Forward Neural Network (MLFNN) with backpropagation algorithm. The performance metrics used in the paper are sensitivity, specificity, and accuracy. They achieved 83.3% for sensitivity, 63.3% for specificity, and 80% for accuracy in diagnosing and detection of PD using an MLFNN.

The authors in [39] used collected database from the Cleveland database from UCI repository and Ibn Al-Bitar Hospital Cardia Surgery and Baghdad Medical City to build heart disease diagnosis system for classifying two cases of heart conditions (Normal, Abnormal). They proposed two classifiers: MLP and Support Vector Machine (SVM) on the dataset consisting of thirteen medical factors to diagnose heart disease. The MLP classifier achieved 98% accuracy when evaluated on collected database whereas SVM attained an accuracy of 96%.

The authors in [70] used OASIS dataset (cross-sectional MRI data) to develop Alzheimer's disease (AD) detection system from magnetic resonance images. The dataset included 416 right-handed men and women aged between 18 to 96 years. Out of these samples, only 126 were picked (28 Ads and 98 HCs). They used inter-class variance criterion for selecting single slice from 3D volumetric data. The classification system they proposed was based on three components: wavelet entropy (WE), multi-layer perceptron (MLP) and biogeography-base optimisation (BBO). Their approach gave 92.40% accuracy, 92.14% sensitivity and 92.47% specificity.

The authors in [22] proposed a Convolutional Neural Network (CNN) for the classification of interstitial lung diseases (ILD). The network they recommended consist of 5 convolutional layers with 2x2 filters and activation layer (LeakyReLU), followed pooling layer with size equal to final feature map and three dense layers, similar to the size of the nal feature maps and three dense layers. The last dense layer has seven outputs, equivalent to the classes considered in the task: healthy, ground glass opacity (GGO), micronodules, consolidation, reticulation, honeycombing and a combination of GGO/reticulation. The dataset used for training and evaluating the proposed method was made using two databases: publicly available multimedia database of ILDs from the University Hospital of Geneva and the Bern University Hospital, Inselspital. For training and evaluating the CNN, they used 1496 images from the dataset, derived from the 120 CT scans. They achieved the classification performance of approximately 85.5% in analyzing the lung patterns.

The authors in [20] proposed a Convolutional Neural Network (CNN) for automated detection of coronary artery disease (CAD). The ECG signal (Normal and CAD) were retrieved from the physionet databases: Fantasia (Normal) and St-Petersburg Institute of Cardiology Technics 12-lead arrhythmia (CAD). The ECG signal has taken from 40 healthy subjects and 7 CAD subjects segmented (2s and 5s). The CNN architecture they proposed comprised of four convolutional. Four max-pooling layers and three fully connected layers for diagnosis of CAD using two and five seconds duration. The model differentiated between normal and abnormal ECG with an accuracy of 94.95%, 93.72% sensitivity, and 95.18% specificity for two-second duration and 95.11% accuracy, 91.13% sensitivity and 95.88% specificity for five seconds ECG segment.

The authors in [27] proposed a new convolutional neural network based multimodal disease risk prediction (CNN-MDRP) algorithm using structures and unstructured real-time hospital data, and the data stored in the data centre. The three-year data from 2013 to 2015 is used containing 31919 patients with 20320848 records. The latent factor model is used to construct the missing data. The experiment is performed on the regional chronic disease of the cerebral infarction. The CNN-MDRP algorithm reaches 94.8% prediction accuracy with convergence speed faster than CNN- based unimodal risk prediction algorithm.

Authors in [21] a developed a Recurrent Neural Network (RNN) to learn the encounter of patients in Pediatric Intensive Care Unit (PICU) of a major tertiary care centre. About 12000 patient's data extracted from Electronic Medical Records (EMR) who were in PICU over the period of ten years. Authors leveraged anonymized EMR from the PICU at Children's Hospital Los Angeles between December 2002 and March 2016. The data for each patient included information like demographics, diagnoses, and alive or not, at the end of an encounter with ICU. The RNN-generated scores achieved signicantly higher accuracy [AUROC higher than 93%] than the clinically used systems: Pediatric Index of Mortality (PIM 2) and the Pediatric Risk of Mortality (PRISM 3). Also, it also provided dynamic tracking of patient condition. The RNN model also outperformed logistic regression and multilayer perceptron models.

In the following sections, we go through some of the algorithms developed by researchers for the classification of diabetes and cardiac arrhythmia.

3.1 Diabetes

Substantial research has been done till now in the context of classifying diabetes using an artificial neural network. In this section, we look into some of these studies and get an overview of the algorithms that have been designed. All of these studies use diabetes database either the PID dataset or some other database.

In [46] used General Regression Neural Networks (GRNN) and Pima Indian Diabetes data set for identifying diabetes. In this paper, GRNN model was assumed to be four layers: input layer with eight features form data set, two hidden layers with 32 and 16 neurons, respectively. Finally, there was a single neuron in output layer, which determines whether the patient has diabetes or not. Out of total 768 samples, 75% and 25% of samples were used for training and testing process. The accuracy achieved with training and testing phase is 82.99% and 80.21%, respectively.

In [57] used Multilayer Artificial Neural Network with back propagation for diagnosing diabetes. In backpropagation, neural network compares computed output value with actual value and calculates the error. The weights are adjusted in each iteration in such a way, that the estimated error to be always less than the previous round. In this paper, the network consisted of an input layer with eight neurons, a hidden layer with six neurons and two neurons in output layer. The data set contains a total 768 samples out of which 500 samples were used during training and remaining 268 during the testing phase. The achieved diagnosis accuracy after 2000 rounds of dataset training becomes 82%.

In [63] used Probability Neural Network (PNN) for diagnosing diabetes. In this paper, PNN model consisted of an input layer with eight neurons representing each feature, single hidden layer, and an output layer with two neurons to diagnose whether a patient has diabetes or not. The dataset which consists of 768 samples, 90% of samples are used in training and remaining 10% during the testing phase. The achieved training, and testing accuracy is 81.49% and 89.56% respectively after 200 rounds.

In [61] used Naïve Bayes, J48 and Radial Based Artificial Neural Network for diagnosing diabete. Pima Indian dataset with 768 data samples out of which 268 samples were used during the testing phase. Naïve Bayes proved to be more efficient with 76.95% accuracy followed by J48 with 76.5% and RBF with 74.34% accuracy's, respectively.

In [50] used a data set with 250 data samples consisting of 27 features. These features also include blood pressure, creatinine, urine PH, fasting glucose. The average age of patients was between 25 to 78 years. They used Multi-layer feed-forward artificial neural networks with backpropagation for diagnosis. Three training functions were applied in backpropagation algorithm namely BFGS Quasi-Newton, Bayesian Regulation and Levenberg-Marquardt. Bayesian Regulation function achieved highest of 88.8% accuracy more than BFGS Quasi-Newton and Levenberg-Marquardt functions.

3.2 Arrhythmia

Significant research have been done till now in the context of classifying cardiac arrhythmia using an artificial neural network. We look into some of the studies and get an overview of the algorithms that have been designed. All of these studies use ECG database either the MIT dataset or ECG dataset extracted from the various data sources.

Authors in [40] proposed a hybrid structure of fuzzy clustered neural networks to classify different types of ECG data with the parametric AR model coefficients and spectral entropy as the parameters. In this study, a total of 20 records were randomly selected from the Massachusetts Institute of Technology- Beth Israel Hospital (MIT-BIH) arrhythmia database with different types of arrhythmias. Between the non-parametric strategies, the model PNN they suggested provided a better output in the performance. Comparison of the proposed method with the other existing approaches revealed that PNN with the clustered features has the higher classification accuracy as compared with the other techniques as well as in the time taken for the classification. In the study, the PNN showed an accuracy of 99.05%, and the MLFFN showed 97.14% respectively for classification of the eight types of ECG beats.

The authors in [24] applied artificial metaplasticity multilayer perceptron to classify cardiac arrhythmias. The MIT-BIH Arrhythmia Database was used to train and test AMMLPs. From 109,871 annotated ECG beats examined by specialists in MIT-BIH, 1000 were selected for this study, which contains four different waveforms related to cardiac arrhythmias target. For test results to be more valuable K-Fold cross-validation was used as minimises the bias associated with the random sampling of the training. The obtained AMMLP classification accuracy of 98.25%.

The authors in [69] employed a Probabilistic Neural Network (PNN) for the classification of normal heart beat and heart beat with some type arrhythmia. They have used MIT-BIH database. A lot of pre-processing was done on the data before feeding it into the PNN. They used 150 records with both the category to train the model and 50 records for testing the PNN model. The achieved classification accuracy was 96.5%.

The authors in [19] proposed an automated classification system for cardiac arrhythmia by using Artificial Neural Network (ANN). For the classification of normal and abnormal classes, ANN with backpropagation was used. Networks models were trained and tested on MIT-BIH arrhythmia dataset and MIT-BIT NSR database. The mixture of arrhythmic and non-arrhythmic data patient was trained the different structures of ANN. Since the numbers of neurons in the hidden layer is an effective parameter for improvement in results, the number of neurons were chosen to achieve the optimum number based on output results. So one hidden layer had three neurons and five neurons in the second layer. The classification performance was evaluated using sensitivity, specificity, classification accuracy and also mean squared error (MSE). The classification accuracy achieved on MIT database was 96.77% and on combination of MIT and NSR database was 96.21%.

Part II

Dataset and Pre-Processing

Chapter 4

Datasets

The datasets that have been used in this thesis are the PID dataset [18] and the MIT dataset [13]. The remainder of this chapter is organized in the following way. In the first subsection, we discuss the PID dataset and the techniques which we have used to make data more refined before feeding into MLFNN. In the next subsection, we discuss the MIT dataset and the pre-processing methods we have implemented before feeding the data into CNN and LSTM.

4.1 Diabetes

In this thesis, the PID dataset is taken from the University of California, Irvine (UCI) repository of machine learning database [18]. This data set is initially from the National Institute of Diabetes and Digestive and Kidney Diseases. The test was conducted according to world health organization criteria and woman inducted in the analysis were 21 years or older age belonging from PIMA Indian heritage. This dataset is used by various researchers to build classification system. It is the main reason for choosing this data set as we can compare our study with various other researchers for Pima Indian Diabetes diagnosing problem. This dataset is composed of a total of 768 instances. Eight features characterize each instance in the data set. All the features have numerical values as seen in Table 4.1.

Table 4.1: Attributes/Features of Pima Indian Data set

Attribute/Features	Types/Values
Number of times pregnant	Numerical Values
Plasma Glucose Concentration	Numerical Values
Diastolic Blood Pressure	Numerical Values (in mmHg)
Triceps skin fold thickness	Numerical Values (in mm)
2-Hour Serum insulin	Numerical Values (in μU/ml)
Body mass index	Numerical Values (in kg/m^2)
Diabetes pedigree function	Numerical Values
Age	Numerical Values (in years)

The last value is binary and is used for classification as it is divided into two classes: Class Zero (Non-diabetic) and Class One (Diabetic). The first eight features are used as input, and the last value is the ground truth. This dataset also contains a large portion of missing data as shown in Table 4.2.

Table 4.2: Missing Data in Pima Indian Data set

No.	Attribute/Features	Types/Values
1	Number of times pregnant	-
2	Plasma Glucose Concentration	5
3	Diastolic Blood Pressure	35
4	Triceps skin fold thickness	227
5	2-Hour Serum insulin	374
6	Body mass index	11
7	Diabetes pedigree function	1
8	Age	63

The total number of Diabetic cases are 268 which corresponds to 34.90% of total cases. The number of Non-diabetic instances is 500 (65.10%).

4.2 Arrhythmia

First, we discuss the selection criteria for the MIT dataset [13]. The MIT dataset database contains a total of 48 ECG records, each of which is slightly over 30 minutes. This data was collected between 1975 and 1979 by the Beth Israel Hospital Arrhythmia Laboratory. The information was obtained from the patients who were under treatment in the hospital. The records were chosen at random which included several rare phenomena which are clinically significant.

In this part, we discuss ECG lead configuration. There are two electrodes which are placed on the chest collecting upper signal and lower signal. Upper signal is modified limb lead two signal, and the lower signal is V1. The signals were reversed in record 114. Due to surgical dressing, the ML2 was not used in the records 102 and 104, but the modified lead V5 was used to collect the upper signal.

In this part, we discuss the annotations. Initially, all the beats were labelled normal by a simple QRS detector. Then the 30 minutes records chart was provided to two different cardiologists for labelling abnormal beats. After that, any disagreement between the cardiologists was resolved by auditing program which analysed the annotations. Initially, the MIT dataset contains around one million beat labels, and few of the beat labels were changed with time. Tables 4.3 and 4.4 shows the standard set of annotation codes defined for ECGs.

Table 4.3: The standard set of annotation codes for Beat Annotations

Symbol	Meaning
N	Normal Beat
L	Left bundle branch block beat
R	Right bundle branch block beat
B	Bundle branch block beat (unspecified)
A	Atrial premature beat
a	Aberrated atrial premature beat
J	Nodal (junctional) premature beat
S	Supraventricular premature beat
V	Premature ventricular contraction
F	Fusion of ventricular and normal beat
r	R-on-T premature ventricular contraction
n	Supraventricular escape beat (atrial or nodal)
e	Atrial escape beat
j	Nodal (junctional) escape beat
E	Ventricular escape beat
/	Paced beat
f	Fusion of paced and normal beat
Q	Unclassifiable beat
?	Beat not classified during learning

Table 4.4: The standard set of annotation codes for Non-Beat Annotations

Symbol	Meaning
[	Start of ventricular flutter/fibrillation
!	Ventricular flutter wave
]	End of ventricular flutter/fibrillation
x	Non-conducted P-wave (blocked APC)
(	Waveform onset
)	Waveform end
p	Peak of P-wave
t	Peak of T-wave
u	Peak of U-wave
‘	PQ junction
’	J-point
—	Isolated QRS-like artifact
+	Rhythm change
s	ST segment change
T	T-wave change
*	Systole
D	Diastole
=	Measurement annotation
”	Comment annotation
@	Link to external data

Chapter 5

Data Pre-processing

The real-world databases are highly prone to missing, inconsistent and noisy data likely due to its origin from heterogeneous sources and typically large size [34]. Therefore data pre-processing is an important aspect that needs to be taken into consideration before developing any machine learning model. There are different pre-processing techniques such as data cleaning, data integration, data transformation and data reduction. The noise and inconsistencies can be removed by implementing data cleaning. The data can be merged from the various source into a single data warehouse by the use of data integration. Data reduction can be applied to remove any correlation between the features, eliminating redundancies in the data (same data appears more than once in the file) and clustering (to make a group of data with the same properties). Data transformation can be used to scale between a predefined range [28] [41].

Time series data is comprised of real-valued measurements of multiple parameters at equal intervals of time. Time series prediction tasks take into account that the future values in the series are a function of past values of the same series. Electrocardiogram (ECG) analysis in medicine, stocks prediction in finance, energy usage prediction in power grids, weather prediction in meteorology and solar activity prediction in space weather are a few examples of real-life applications of time series modelling and forecasting.

Normalization rescales the data from the original range so that all values are within the specified range of 0 and 1. It can be required in some machine learning algorithms when your time series data has input values with differing scales. It needs that you can accurately estimate the minimum and maximum observable values. You may be able to determine these values from your available data. Standardization is a process of rescaling the dataset in which the mean of the values is 0, and the standard deviation is 1. It can be assumed by subtracting the mean value or centring the data. It can be very efficient and is required in some machine learning algorithms where the time series data has input values within different scales [29].

ANNs cannot interpret missing values in the data if any and when database is highly skewed. Missing data becomes a standard issue when working with medical databases [43]. It can be due to costly medical test which a patient cannot afford, or the values were taken but not recorded due to time constraints.

5.1 Diabetes

The process of training in the neural network can be made more effective by applying some pre-processing techniques on the network inputs before feeding them into the system. In the PID dataset, we first checked if there are any correlated features. Correlation can be defined as a measure of how strongly one input feature depends on another. By removing any correlated feature, we can increase the speed of learning of an algorithm. Since deletion of such feature will decrease the curse of dimensionality. It can also reduce the bias in the neural network. Random forest and Logistic regression classifier can show a decline in performance if there are any correlation bias [68]. We check if there are any correlated features in the data which can impact the classification accuracy of the classifiers. Figure 5.1 indicates that there are no correlated features in the PID dataset.

A few functions can transform inputs data into an improved form for the usage in the network. The normalization can process the data to be appropriate for the training process. In this process, the data is scaled in some specific range for every input feature to reduce the bias in the neural network. It also speeds up training time by starting the training process for each feature within the same scale. It is very effective when the difference between the two input features is on the very large scale. On the PID

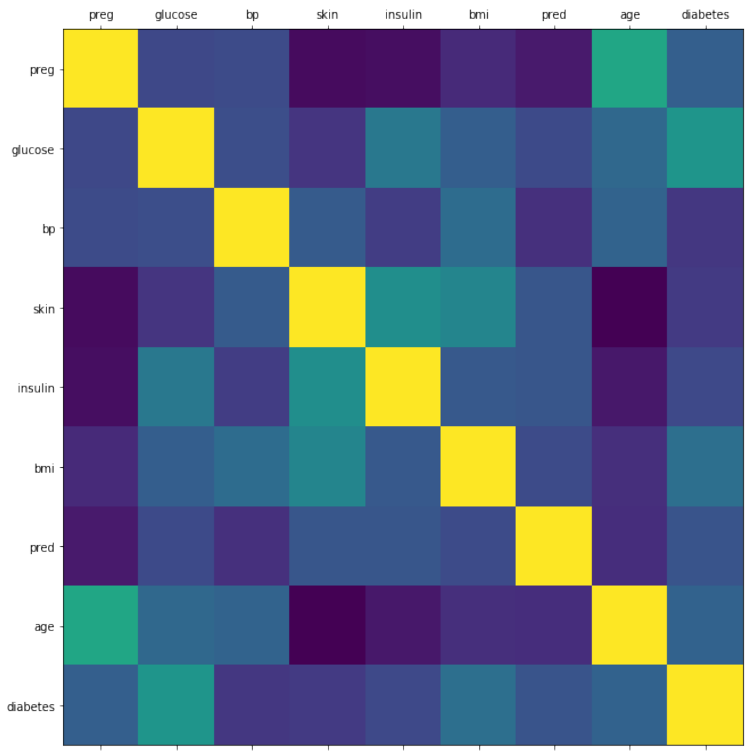

Figure 5.1: Figure showing correlation between the features present in the PID dataset

dataset, we have used MinMaxScaler for the process of normalization. It transforms all the features by scaling them into a given range.

The equation 5.1 provides the formula how normalization converts the values.

$$MinMaxScaler(feature\_range = (0, 1), copy = True) \tag{5.1}$$

It transforms all the features by scaling them in to a given range. The transformation is given by the following equations 5.2 and 5.3.

$$X_{std} = \frac{X - X.min}{X.max - X.min} \tag{5.2}$$

$$X_{scaled} = X_{std} * max - min + min \tag{5.3}$$

Here max and min represents the feature range.

The PID dataset suffers from a lot of missing values. They can impact the classification accuracy of the neural network model. So, to handle the missing the data we have used three different techniques. We first removed all the rows which have some missing attributes. In second, we replaced all the missing values with zero. In the third technique, we replace all the missing values with the mean of the other values in the attribute. For this, we have used imputer from the sklearn library [15]. We replaced all the missing values in the attribute by mean strategy.

$$Imputer(missing\_values =' NaN', strategy =' mean', axis = 0)$$

Axis here can have two values: 0 (to impute along the columns) and 1 (to impute along the rows). When we need to predict from the finalized model, the same imputing strategy must be implemented on the new data which we have applied on the training dataset.

These all are the pre-processing techniques which we have implemented before feeding the PID dataset into the networks. By visualizing the data, we have checked for any correlation among the features. There is a vast difference between the values of attributes in the PID dataset. So, we applied normalization to rescale the values between a specific range. In the end, for handling the missing values, we use Imputer from the skearn library to fill them with the mean of the attribute.

5.2 Arrhythmia

The MIT dataset [13] contains 48 two channel ECG recordings, each recording is little more than 30 minutes, digitized at 360 HZ per channel with 11-bit resolution over a range of 10mV. Our primary purpose is to perform the minimum pre-processing to the data so that any algorithm can process and classify the heartbeats in the format as they were recorded. In the repository, the data files and annotation files are kept in the Waveform Database (WFBD) format. There are a total of three extensions for each recording. The first one is .dat extension; these are Signal Files. These are the binary files which contain the sample of digitized signal. For the proper interpretation of these files we need associated header files, and these files store the waveforms. The second extension is .hea; these are header files. These are the small files which describe the content of the related signal files. The third extension is .ann; these are annotation files. These are the binary files which contain the annotations referring to specific samples in the associated files. For extracting and processing these WFDB files, we have used WFDB software utility. The rdann and rdsamp commands from WFDB software toolbox are used for converting signal and annotation files into text files.

We have trained out the network with the individual heartbeats to avoid dependencies between the class labels of consecutive heartbeats. If the network is trained on original sequence, then implementing heartbeat to heartbeat independence would not have been possible. The reason is most of the data in the all sequences have same class label (normal or abnormal). To train the network on randomized sequence, we need to apply normalization on all heartbeats so that they can have the same baseline. Here randomization corresponds to sampling each heartbeat uniformly from all heartbeats in the original sequence.

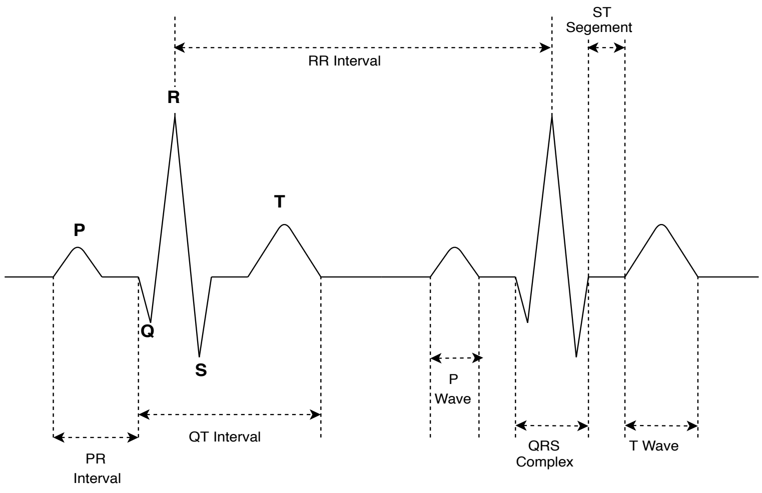

Figure 5.2: An ECG of Normal Sinus Rhythm

In Figure 5.2, we can see different intervals and waves that are present in an ECG signal. Since all the relevant features are contained around the R peak, so we extracted a series of samples centred around this peak of the heartbeats. In this way, in a continuous ECG sequence, we defined and extracted a single heartbeat.

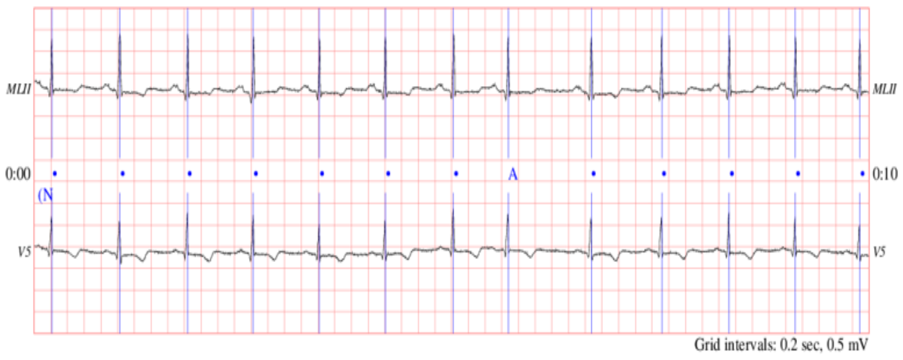

Figure 5.3: Waveform Plot with Annotation (10 seconds) obtained from PhysioBank ATM [14]

In Figure 5.3 we can see every heartbeat has been annotated at R peaks. We calculate the interval between the consecutive R peaks, and it came out as 0.75 seconds approximately. The window that we have used to extract the sample has a size of 1 second. With this size of the window, almost all examples contain three hundred and sixty-one samples. Some cases included three hundred and sixty samples. In those cases, we duplicated the sample at the end to make sure same sample size for each heartbeat. We also removed some irrelevant examples such as: in which there were not complete 361 samples, in which annotation was neither normal or some type of arrhythmia. In the end, we have left with around ninety-three thousand five hundred examples approximately. Out of these around 20% examples had an annotation with some arrhythmia.

Part III

Experiments and Results

Chapter 6

Experimental Setup

In this chapter we look into the experimental setup we have implemented for classification of diabetes and arrhythmia. In the first section, we discuss the experimental setup of MLFNN for classifying diabetes and explains briefly the crucial design considerations which we have considered and evaluation techniques we have used for the experiments. In the second section, we have described the experimental setup of CNN and LSTM for the task of classifying arrhythmia. After that, we have explained the evaluation techniques we have considered for this task.

6.1 Diabetes

In this section, We discuss the network architecture and design consideration of MLFNN for classification of diabetes that are considered for the experiments. Firstly, we present the network architecture that is used. In the second part, we discuss the evaluation metrics that are used.

6.1.1 Network Architecture

Key design consideration of a feed-forward neural network is to determine the overall structure of the network. It includes choosing the depth of the network and the width of each layer. The depth of network means the number the hidden layer and the width corresponds to the number of neurons in the layer [36]. A single layer feed-forward network is able to represent any function. The problem with such network is that the layer can be infeasible and not be able to learn and generalize correctly. In contrast, the deep models can significantly reduce the number of units per layer and the generalization error. Another key consideration in designing the network is the connections between the layers. If we decrease the number of connections, we can reduce the number of parameters and the amount of computation that is required to evaluate the network. These all architectural considerations are highly problem dependent. The ideal network architecture for a task must be found by doing repeated experimentation and monitoring the evaluation metrics.

We have designed the network with input layer (8 Neurons), three hidden layers with 50 neurons each and an output layer with a single neuron. The network architecture can be seen in Figure 6.1

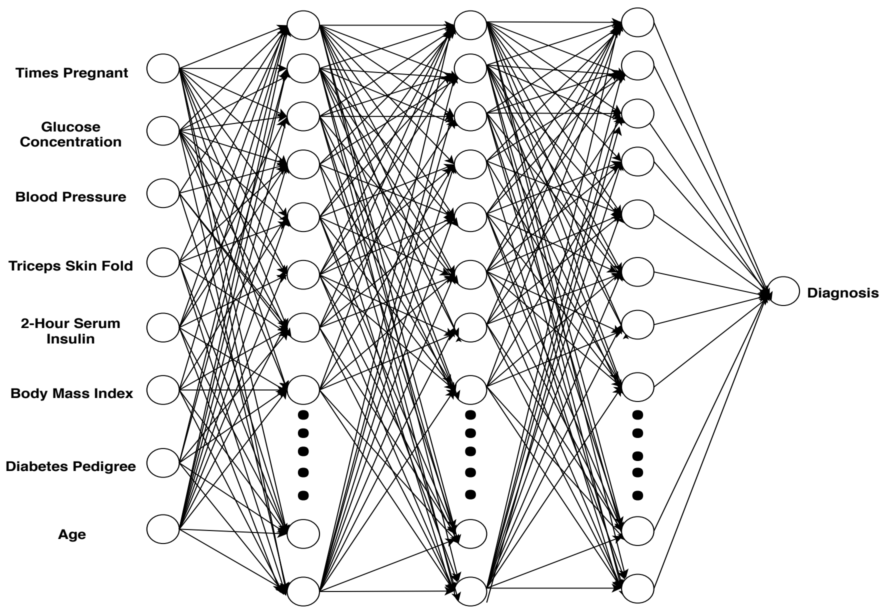

Figure 6.1: The MLFNN Network Architecture with Input Layer (8 Neurons), Three Hidden Layers (50 neurons each) and an output layer (1 neuron)

For the MLFNN model evaluation, we have used repeated holdout validation technique which is also known as Monte Carlo Cross-Validation.

This method can be better illustrated by the following equation 6.1.

$$ACC_{average} = \frac{1}{k}\sum_{j=1}^{k} ACC_j \tag{6.1}$$

We have split the PID dataset into 90% training and 10% testing. Then we repeated this experiment 150 times with different seed and then compute the average performance. In this way, every time the model is evaluated on the randomly selected test set. It gives us a better idea about the model stability and how well it can perform on random test data.

6.2 Arrhythmia

In this section, we discuss the network architecture and other design consideration that are considered for the experimentation. We have developed a CNN and LSTM for classifying cardiac arrhythmia. Firstly, we discuss the network architectures that are used. In the second part we discuss about the evaluation metrics are used.

6.2.1 Network Architecture

Firstly, we discuss the architectural design of CNN we have used to perform the experimentation. Generally, a CNN is comprised of an input layer, convolutional layers, pooling layers and one or more dense layers. A convolutional layer is specified by the number of filters it has, size, stride, padding of each filter. CNN network usually contains one or more convolutional layer, followed by pooling layers and a dense layer which receives the full input of the previous layer. After a lot of experimentation, we settle down with a final architecture of CNN. The CNN architecture we have used is comprised of a convolutional layer with a total of 50 filters of size 30x1. Then we used a maxpool layer, and it has a size of 5x1. Then there is a fully connected layer with a total of hundred units followed by a sigmoid layer which has a single unit. Figure 6.2 provides the in-depth details of the structure of the used CNN. We used an input layer feeding the data into the network. After that, there is a Convolution Layer (50 Filters 30x1), Max-Pooling Layer (5x1). In the end, there is a fully connected layer with a total of 100 units followed by sigmoid unit presenting the output.

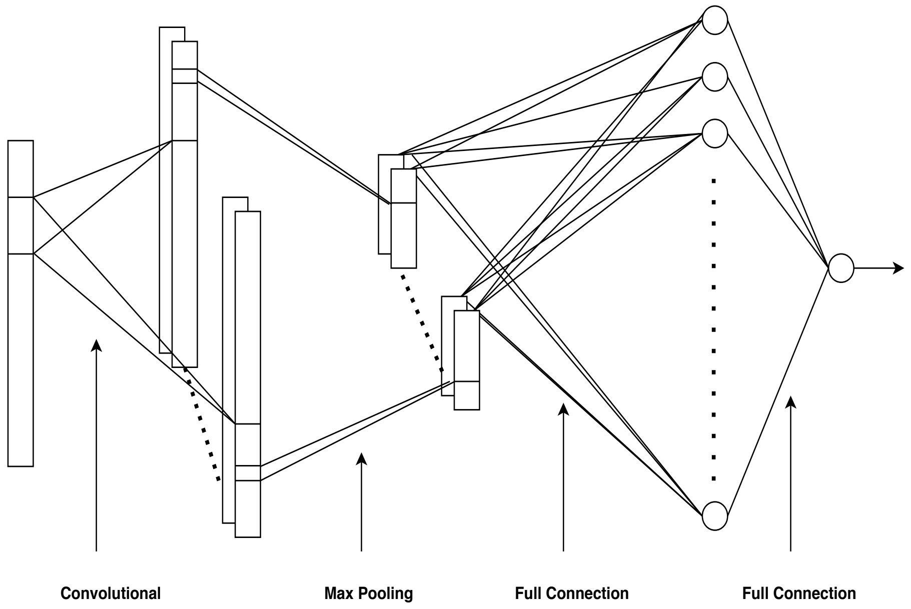

Figure 6.2: CNN Network Architecture with Convolution Layer (50 Filters 30x1), Max-Pooling Layer (5x1), Fully Connected Layer (100 Units) and Sigmoid Layer (1 Unit)

Now we discuss the architecture of LSTM we have implemented for classifying cardiac arrhythmia. The LSTM RNNs have an internal memory which is also known as cells. They consist of three gates which control the flow of the information through the network. At each timestep, the input and output gates control the flow of the data, and a forget gate gives an ability to LSTM to clear the memory. In the experimental setup we have implemented, the network has an LSTM layer with a total of thirty units, and after that, there is fully connected layer a total of hundred units. In the end, there is a sigmoid layer with a single unit.

For CNN and LSTM model evaluation, we have used k-fold cross-validation techniques. In this technique, we split the data into k parts. One part of this data is used for validation, and other remaining parts act as training data for model evaluation. For the experiments we have used 4 -fold cross validation which is illustrated in Figure 6.3.

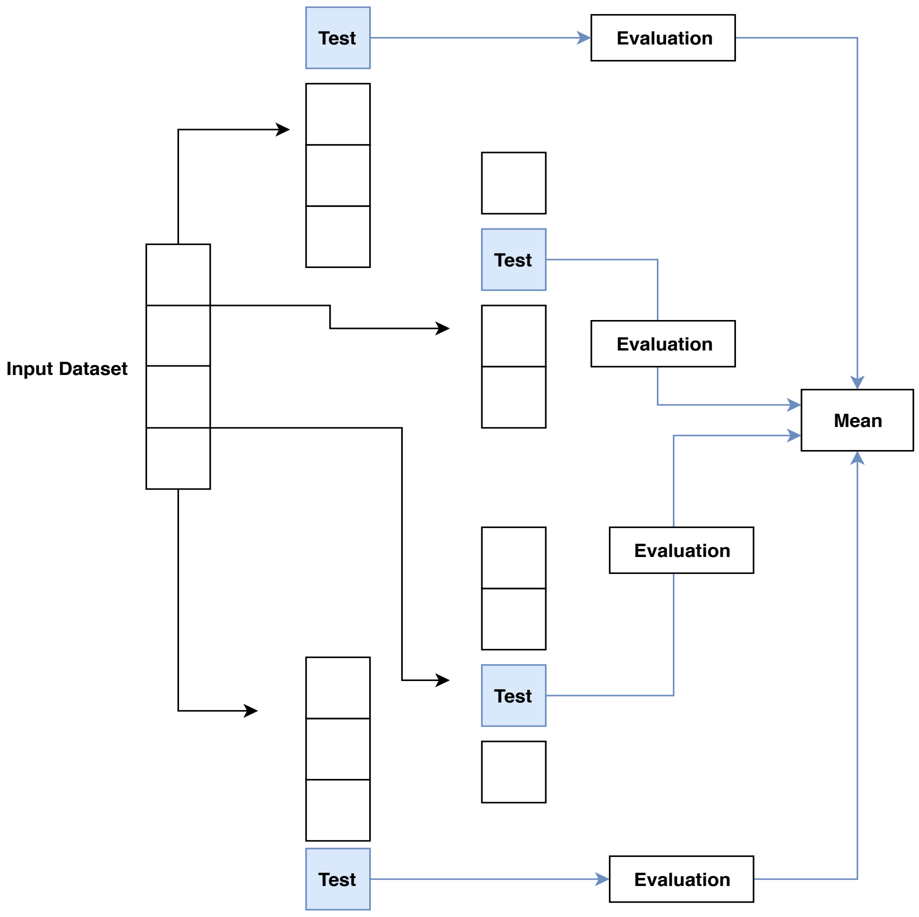

Figure 6.3: K-fold cross-validation scheme, with K=4

Chapter 7

Results and Discussions

In this chapter, we discuss the results we have achieved by conducting various experiments on the PID dataset and the MIT dataset. In the first section, we focus on the result obtained from experiments performed for the classification of diabetes on the PID dataset. In the second subsection, we focus on results we achieved from the experiments conducted for the classification of arrhythmia on the MIT dataset. All the results try to provide the answer to the research questions and subordinated research questions we have discussed in chapter 1 in the research question section 1.2.

7.1 Diabetes

In this section, we present and discuss the results that are obtained for answering the research question 1 (see 1.2) and subordinated research questions (see 1.2.1) by implementing MLFNN model for the PID dataset.

In the first experiment, we develop an MLFNN model and then compare its performance with different classifiers. In the second experiment, we present and discuss the results obtained by various techniques for handling the missing data. In the third experiment, we discuss the result obtained by different activation function on the classification accuracy in the MLFNN model. In our last experiment, we discuss the results obtained by different learning algorithm on classification accuracy the MLFNN model.

7.1.1 Comparison with other classifiers

In this section, we compare the performance of different machine learning classifier with MLFNN model proposed by us. We are using Naïve Bayes (NB), Random Forest (RF), Logistic Regression (LR) for the classification of diabetes. We first discuss about the classifiers we have used for evaluation and then compare the results with MLFNN model proposed by us.

MLFNN has been explained in theoretical background 2.2. The network architecture for the model proposed by us has been explained in the section 6.1. So here we provide the result that we achieve with MLFNN. We have achieved the training accuracy of 80.23% and the testing accuracy of 79.05% on the PID dataset for the classification of diabetes. This result is achieved by imputing the missing values with the mean. More details about the MLFNN model design and experimentation can be found in the paper which is attached at the end of this thesis in part IV.

Naïve Bayes (NB) classifier is a very straightforward and robust algorithm for the classification task [52]. To understand the NB classifier, we need to understand the Bayes theorem. So, let's first discuss the Bayes theorem. Bayes theorem works on the concept of conditional probability. Conditional probability is the probability that an event will happen, given that another event has already occurred. Using the conditional probability, we can calculate the probability of an event using its prior knowledge.

Equation 7.1 represents the Bayes' theorem [35]. A and B are two events.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{7.1}$$

- P(A|B): The conditional probability that an event A occurs, given that B has already happened. It is also known as the posterior probability.
- P(A) and P(B): The probability of event A and event B respectively.
- P(B|A) : the conditional probability that event B occurs, given that A has already happened.

NB is a classifier which uses the Bayes theorem. NB predicts membership probabilities for each class like the probability that a given data belongs to a particular class. After that, the probabilities are counted, and the class which has the highest probability is the most likely class. This concept is also known as Maximum A Posteriori (MAP).

For NB classifier, we have achieved the training accuracy of 75.11% and the testing accuracy of 67.53%.

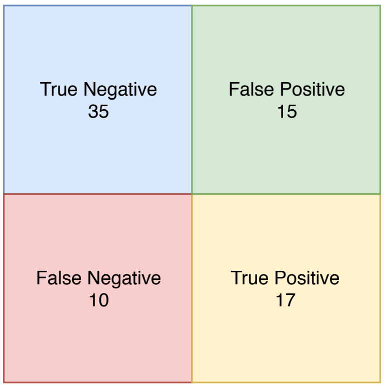

Figure 7.1: Confusion Matrix for Naïve Bayes

Table 7.1: Classification Report for Naïve Bayes

	precision	recall	f1-score	support
0	0.78	0.70	0.74	50
1	0.53	0.63	0.58	27
avg / total	0.69	0.68	0.68	77

Figure 7.1 shows the confusion matrix by NB classifier. True negative are the cases which are not diabetes and model predicts it in the same way. False positive are the cases which are not diabetes, but model predicts it to be diabetes. In our test data, there are a total of 50 cases which are non-diabetic. The NB classifier is predicting 35 of them correctly but 15 cases which are non-diabetic it is predicting them wrong by classifying them as diabetic. True positive are the cases which are diabetes and model predicts it the same way. False negative which are diabetes, but model predicts it to be not diabetes. In the same way, there are a total of 27 diabetic cases

out of which NB classifier is predicting 17 cases correctly but rest 10 cases incorrectly.

Table 7.1 shows the classification report of NB classifier. The classification report generates statistics based on the confusion matrix values. Recall here is the true positive rate and sensitivity. It shows how well the model is classifying diabetes when the result is actually diabetes. For NB classifier we achieve 63% recall. Precision is also positive predictor value. It shows how often the patient had diabetes and model said they would. NB classifier has a precision of 53%.

Random Forest (RF) algorithm is a type of supervised classification algorithm [59]. RF create a forest with many trees. There is a direct relationship between the number of trees in the forest and the results it can get. The higher the number of trees, the more accurate the result. There are many advantages of using RF. We can use it for both classification and regression tasks. RF algorithm can counter one critical problem that can make the results worse like by overfitting (when a model learns the training data to such a extent that it negatively impacts the performance of the model on test data), but for RF algorithm, if there are enough trees in the forest, the classifier won't overfit the model. The third advantage is RF classifier can handle missing values, and the last advantage is that the RF classifier can be modelled for categorical values. There are two stages in RF algorithm; in the first stage, the creation of the random forest, the other is to predict the random forest classifier built in the first stage.

For RF classifier, we have achieved the training accuracy of 99.42% and the testing accuracy of 71.43%.

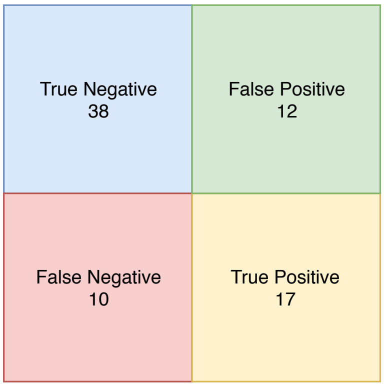

Figure 7.2: Confusion Matrix for Random Forest

Table 7.2: Classification Report for Random Forest

	precision	recall	f1-score	support
0	0.79	0.76	0.78	50
1	0.59	0.63	0.61	27
avg / total	0.72	0.71	0.72	77

Figure 7.2 shows the confusion matrix by RF classifier. In our test data, there are a total of 50 cases which are non-diabetic. The RF classifier is predicting 38 of them correctly but 12 cases which are non-diabetic it is predicting them wrong by classifying them as diabetic. In the same way, there are a total of 27 diabetic cases out of which RF classifier is predicting 17 cases correctly but rest 10 cases incorrectly. So there is sight improvement as compared to NB classifier in the RF classifier when it is classifying the cases which are not diabetic.

Table 7.2 shows the classification report of RF classifier. For RF classifier we achieves same 63% recall as we achieved for NB classifier. There is slight improvement in the precision. RF classifier achieves 59% precision.

Logistic Regression (LR) model is a type of supervised classification model involving a linear discriminant [56]. Given a set of inputs, LR does not try to predict the value of a numeric variable. Instead, it predicts that the output is a probability that the given input point belongs to a particular class. The central principle of Logistic Regression is the assumption that the input space can be separated into two regions, one for each class, by using a linear boundary. This dividing plane is called a linear discriminant, because, its linear function, and it helps the model classify between points belonging to different classes. The logistic regression models are categorized based on the number of target classes and use the functions like sigmoid or softmax functions to predict the target class. LR model uses the sigmoid function when there is binary classification task and softmax function when there is multi-classification task.

For LR classifier, we have achieved the training accuracy of 77.71% and the testing accuracy of 72.73%.

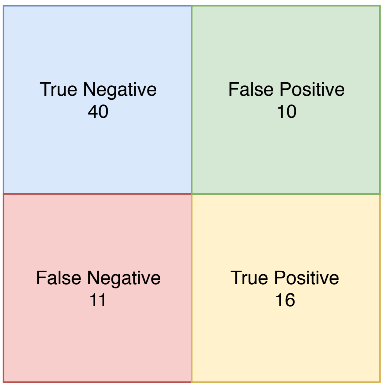

Figure 7.3: Confusion Matrix for Logistic Regression

Table 7.3: Classification Report for Logistic Regression

	precision	recall	f1-score	support
0	0.78	0.80	0.79	50
1	0.62	0.59	0.60	27
avg / total	0.73	0.73	0.73	77

Figure 7.3 shows the confusion matrix by LR classifier. The LR classifier is predicting 40 non-diabetic cases correctly but 10 cases which are non-

diabetic it is predicting them wrong by classifying them as diabetic. In the same way, there are a total of 27 diabetic cases out of which LR classifier is predicting 16 cases correctly but rest 11 cases incorrectly. So there is sight improvement in LR classifier as compared to NB and RF classifier when it is classifying the cases which are not diabetic cases.

Table 7.3 shows the classification report of LR classifier. For LR classifier we achieves same 59% recall which is less then from both NB and RF classifier. There is improvement in the precision. LR classifier achieves 62% precision.

Table 7.4: Performance of different classifiers

	NB	LR	RF	MLFNN
Training Accuracy%	75.11	77.71	99.42	80.23
Testing Accuracy%	67.53	72.73	71.43	79.05

Table 7.4 shows that the MLFNN model performs better than all other classifiers giving the highest accuracy of 79.05%. All the experiments are performed on the PID dataset by imputing the missing values with mean. The RF classifier gives the highest accuracy during training (99.42%), but the testing accuracy is 71.43%. The least testing accuracy has been provided by NB classifier (67.53%). This can be explained by the fact that MLFNN can model the highly skewed data with its complex structure. The depth of the neural network and backpropagation algorithm helps the model to adjust the weights in such a way that it can provide the highest accuracy with the testing data.

7.1.2 Handling Missing Data

In this experiment, we have used three different approaches to handle the missing data.

The first approach and easiest way to deal with the missing values are to remove the instances with missing values. This technique may lead to the loss of highly valuable information but is very useful when the database is highly skewed. There is a total of 768 instances, but after removing cases the missing values, we are left with only 392 cases.

The second approach is to replace all missing values with zeros. We fill all the values that are missing in the dataset with zero. The problem with this technique is that sometimes it does not make any sense when the attribute has zero value. In this dataset, let us suppose that some data from blood pressure attribute is missing. It does not seem natural that a person has a zero blood pressure. Similarly, there are other attributes, replacing them with zero does not make sense.

The third technique is to impute all missing values with mean value. In this method, the missing value of an attribute is replaced by the average of all the available values of the same attribute in the data. Table 7.5 shows the classification performance of all the three approaches.

Table 7.5: Performance with variation in techniques to handle missing data

	Remove samples	Replace (mean)	Replace (zero)
Training Accuracy%	83.91	80.23	81.00
Testing Accuracy%	82.50	79.05	78.93
Training MSE	0.111	0.131	0.130
Testing MSE	0.131	0.145	0.149

The testing accuracy after removing the sample (82.5%) is superior then replacing the missing values with the mean (79.05%) followed by replacing with zero (78.93%). The reason behind this behaviour is that sometimes replacing values make it hard for the neural network to properly adjust the impact of certain features which normally plays a crucial role in solving the problem.

In Figure 7.4, it can be seen that loss is decreasing with the number of epochs for the testing set. In Figure 7.5 we can see that after 100 epochs there is an increase in the loss for the testing set but the model is working well on training data. This is clear indication that the model is overfitting the data. In Figure 7.6 it can be seen that loss on testing data becomes stable after 100 epochs and remain almost same even after 500 epochs.

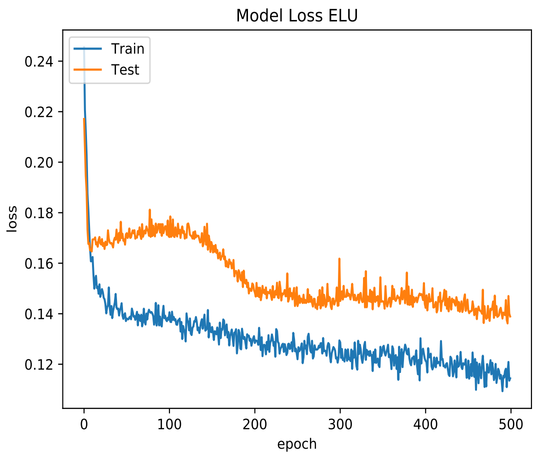

Figure 7.4: Figure showing change in the Loss per epoch after removing the missing values. The figure shows one example out of 150 randomly selected test sets.

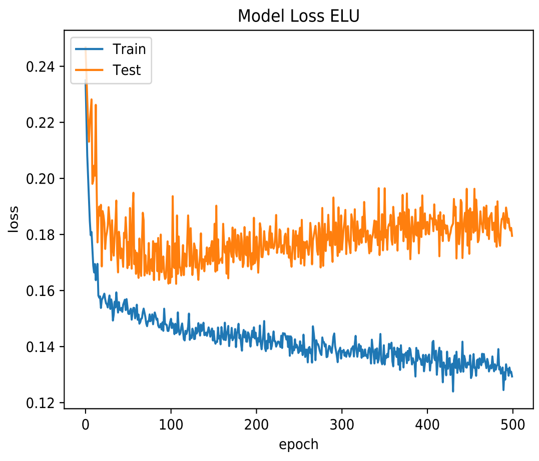

Figure 7.5: Figure showing change in the Loss per epoch after Replacing the missing values with zero. The figure shows one example out of 150 randomly selected test sets.

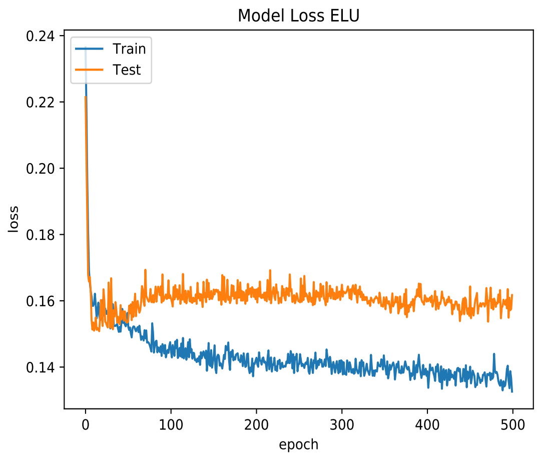

Figure 7.6: Figure showing change in the Loss per epoch after imputing the missing values with mean. The figure shows one example out of 150 randomly selected test sets.

7.1.3 Activation Units

In this experiment, we compare classification performance of four different activation units with Adam as a learning algorithm. Table 7.6 shows the performance results of four activation units we have considered for the experiment.

Table 7.6: Performance with variation in activation functions

	ELU	SELU	Leaky ReLU	ReLU
Training Accuracy%	83.91	87.11	92.63	98.98
Testing Accuracy%	82.50	82.05	81.15	78.73
Training MSE	0.111	0.094	0.057	0.011
Testing MSE	0.131	0.132	0.143	0.178

The results show that the ELU gives the highest accuracy of 82.5% followed by SELU (82.05%), Leaky ReLU (81.15%) and ReLU (78.73%). ELU and SELU have improved learning characteristics as compared with other two activations for the PID dataset. In contrast with ReLU, ELU does not merely discard the neurons with negative values, which allows them to push mean unit activation closer to zero like batch normalization. ELU activation is more robust to noise whereas Leaky ReLU, which also consider negative values do not ensure noise robust deactivation state. SELU allows constructing self-normalizing neural networks. It pushes activation's to zero mean and unit variance, leading to the same effect as batch normalization, avoiding exploding and vanishing gradient.

On the training set, the loss for ReLU (0.011362) is lowest followed by Leaky ReLU (0.057298). The loss in the other two activations ELU and SELU are 0.111245 and 0.094413 respectively. Figure 7.7 shows that the loss in the testing set is decreasing with increase in the number of epochs. Figure 7.8 indicates that the after 210 epochs the loss in testing set starts increasing. It is a clear sign of overfitting. Figure 7.9 shows that the loss in the testing set for SELU activation function becomes almost stable after 200 epochs. Figure 7.10 shows the overfitting issue for ReLU activation function after 70 epochs declining the performance of the model on testing set.

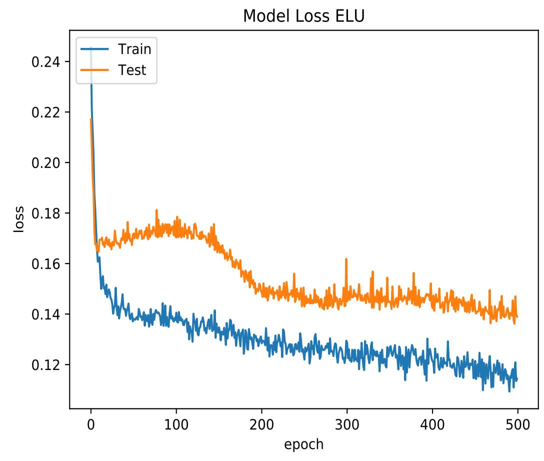

Figure 7.7: Figure showing change in the Loss per epoch ELU activation function. The figure shows one example out of 150 randomly selected test sets.

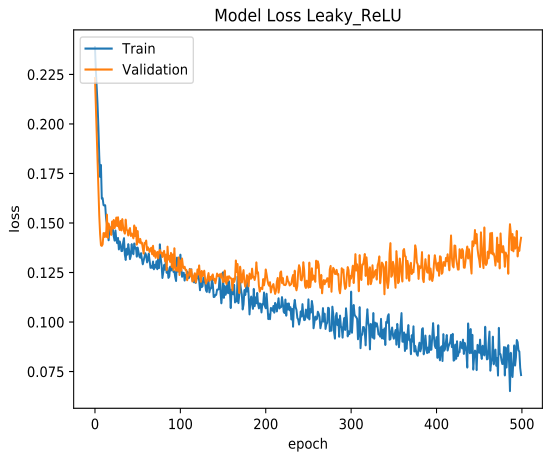

Figure 7.8: Figure showing change in the Loss per epoch Leaky ReLU activation function. The figure shows one example out of 150 randomly selected test sets.

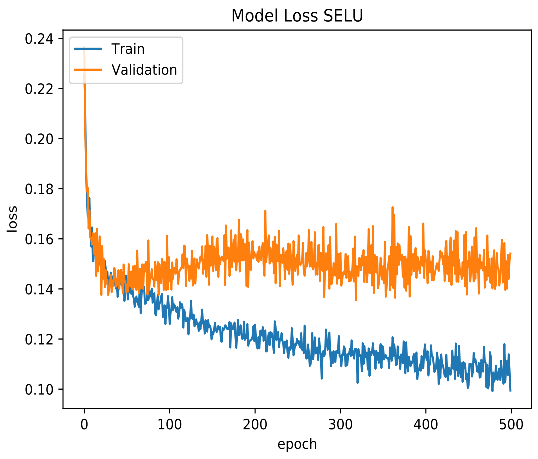

Figure 7.9: Figure showing change in the Loss per epoch SELU activation function. The figure shows one example out of 150 randomly selected test sets.

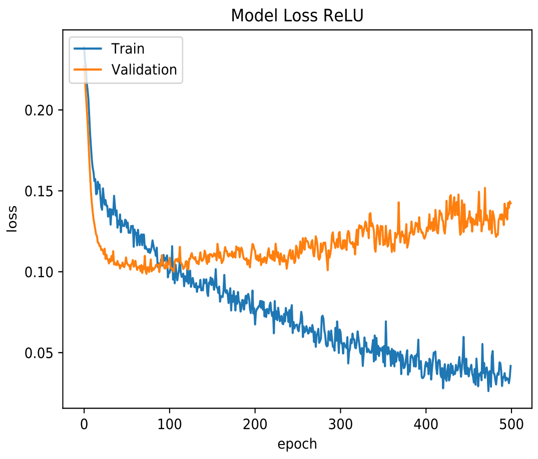

Figure 7.10: Figure showing change in the Loss per epoch ReLU activation function. The figure shows one example out of 150 randomly selected test sets.

7.1.4 Learning Algorithms

In our last experiment, we study the impact of two different learning SGD and Adam algorithm on classification performance on the PID dataset.

Table 7.7: Performance with variation in learning algorithms

	Adam	SGD
Training Accuracy%	83.91	78.93
Testing Accuracy%	82.5	78.43
Training MSE	0.111	0.140
Testing MSE	0.131	0.146

Table 7.7 shows the classification performance of SGD and Adam algorithm using ELU as an activation function on the PID dataset.

Not surprisingly, we find the performance of Adam (82.5%) is better than SGD (78.4%). Since for the SGD, adjusting the learning rate is one of the crucial parameters, and we are using the default learning rate, and it is taking predefined steps to reduce the error. Whereas, Adam is cleverly adjusting the learning rate according to the variance in gradient values it encounters.

In Figure 7.14 it can be seen that loss in testing data is decreasing with the increase in the number of epochs. In Figure 7.15 it can be seen that the loss in the testing data becomes stable after 100 epochs. After 500 epochs the loss for testing data is approximately 0.15 for Adam whereas it is 0.17 for SGD.

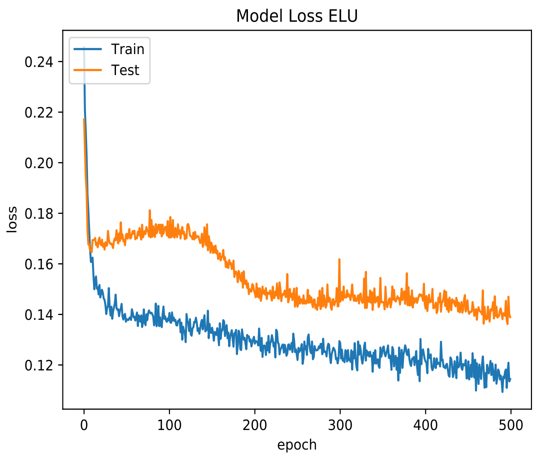

Figure 7.11: Figure showing change in the Loss per epoch for Adam learning algorithm. The figure shows one example out of 150 randomly selected test sets.

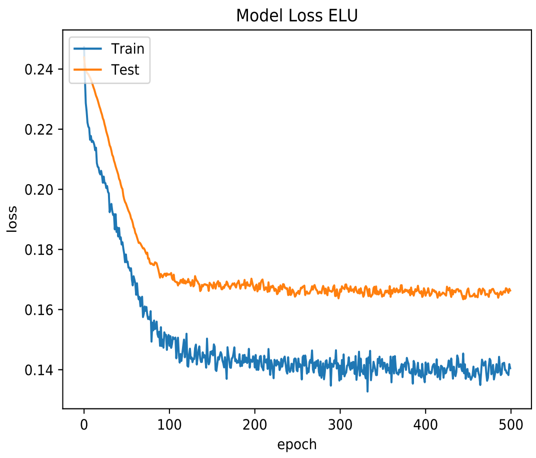

Figure 7.12: Figure showing change in the Loss per epoch for SGD learning algorithm. The figure shows one example out of 150 randomly selected test sets.

7.2 Arrhythmia

In this section, we present and discuss the results obtained by CNN and LSTM architectures on the MIT dataset. For training the networks, we have used stochastic gradient descent algorithm (SGD) with batch size 256. Cross-entropy is the loss function which has been used in all the experiments. In the first experiment, we evaluated the classification performance of CNN and LSTM architectures. In the second experiment, we discuss, how the L2 regularization lambda impacts the classification performance of LSTM architecture. In the third experiment, we have discussed the impact of the number of epochs on classification accuracy in the LSTM architecture. All the results try to provide the answer to the research questions and subordinated research questions (see 1.2.2) we have discussed in chapter (see 1).

7.2.1 Comparison of CNN with other algorithms

In this section, we present the comparison of our results with existing literature. Table 7.8 and 7.9 provides the information regarding the modelling techniques and performance achieved by models developed by other researchers on the MIT dataset. Our CNN model provides the higher classification accuracy (99.19%) as compared with other models.

Table 7.8: Comparison of classification accuracy achieved by CNN model developed in this thesis with other algorithm developed by researchers for ECG classification

Researchers	Modeling Technique	Performance Measures	Accuracy (%)
R.Ceylan et al. (2008) [26]	3-layered FFNN, T2FCNN, fuzzy clustering neural network	Sensitivity, Specificity, Average detection rate	96.70 100 98.35
A.Dallali et al. (2011) [31]	FCM and heart rate variability (HRV)	Accuracy	99.05

Table 7.9: Comparison of classification accuracy achieved by CNN model developed in this thesis with other algorithm developed by researchers for ECG classification

Researchers	Modeling Technique	Performance Measures	Accuracy (%)
M.Vijayavan an et al. (2014) [69]	Feed forward PNN classifier Trained with extracted features	Accuracy	96.5
S.Yu et al. (2008) [72]	PNN (radial basis layer and competitive layer), 3-layer FFNN with back propagation algorithm	Sensitivity, Specificity, Overall accuracy	98.508 99.906 98.710
M.Das et al. (2014) [33]	MLPNN classifier ST+MLPNN, ST+WT+MLPNN	Sensitivity, Accuracy	69.38 97.5
X.Tang et al. (2014) [67]	QNN trained using gradient descent method	Accuracy	91.7
J.Nasiri et al. (2009) [55]	Genetic algorithm-SVM	Accuracy	93.46
A.Muthuchu dar et al. (2013) [53]	Feed forward network with back propagation algorithm as training algorithm	Accuracy	96
Z.Zidelmal et al. (2013) [74]	SVM with rejection	Average accuracy with no rejection, Minimal classification cost	97.2 98.8
Our Work	**CNN**	**Accuracy**	**99.19**

7.2.2 Evaluation of CNN and LSTM models

In this experiment, we present the evaluation of the performance of CNN and LSTM model on the MIT dataset. The learning rate is initialized to 0.01 and decremented by 6 x 10^{-6} at each epoch. There are a total of 1500 epochs used for both LSTM and CNN model. For updating the weights, Nesterov momentum [65] has been used.

Table 7.10 and 7.11 provides the detail of the classification accuracy and time taken by single epoch (in seconds) for both the LSTM and CNN architecture. As mentioned in the network architecture for Arrhythmia we have used 4-fold cross-validation. So, the classification of all the folds are provided and, in the end, the average has been taken. The similar procedure has been followed for the time taken by single epoch. We can notice that the classification performance of CNN architecture (99.19%) is more than the LSTM architecture (98.41%). This variation in results shows that CNN is performing better then LSTM for the MIT dataset. Time taken by LSTM for single epoch is 0.19 seconds (average) which is much less than the average time taken by CNN model for single epoch (4.5 seconds). The performance of CNN and LSTM are mostly problem dependent. LSTM models with their ability to store data in their memory generally perform well in the prediction task. Also, they perform better when the input is of arbitrary length [71]. There are also a lot of hyperparameters, tuning them could have provided a different result. So, for the network architecture and hyperparameter tuning we have used for this experiment with the MIT dataset CNN has shown the better result as compared to LSTM.

Table 7.10: Performance of CNN and LSTM models with 4-Fold cross validation

$\lambda = 0.0$					
	CV1	CV2	CV3	CV4	Average
CNN Accuracy	99.48	99.1	99.19	98.98	99.19
LSTM Accuracy	98.98	98.66	98.06	97.93	98.41
CNN Time(s)	5.297	4.458	4.458	4.459	4.668
LSTM Time(s)	0.190	0.190	0.190	0.190	0.190

Table 7.11: Performance of CNN and LSTM models

$\lambda = 0.001$					
	CV1	CV2	CV3	CV4	Average
CNN Accuracy	99.49	99.24	98.96	98.96	99.16
LSTM Accuracy	98.63	98.4	97.92	97.79	98.18
CNN Time(s)	4.559	4.558	4.559	4.558	4.559
LSTM Time(s)	0.192	0.192	0.192	0.192	0.192
$\lambda = 0.002$					
	CV1	CV2	CV3	CV4	Average
CNN Accuracy	99.58	99.3	99.03	98.8	99.18
LSTM Accuracy	98.52	98.22	97.82	97.76	98.08
CNN Time(s)	4.559	4.559	4.559	4.559	4.559
LSTM Time(s)	0.195	0.194	0.194	0.194	0.194
$\lambda = 0.003$					
	CV1	CV2	CV3	CV4	Average
CNN Accuracy	99.51	99.28	98.87	98.75	99.10
LSTM Accuracy	98.63	98.25	97.73	97.62	98.056
CNN Time(s)	4.559	4.558	4.558	4.558	4.558
LSTM Time(s)	0.195	0.194	0.194	0.194	0.194
$\lambda = 0.004$					
	CV1	CV2	CV3	CV4	Average
CNN Accuracy	99.49	99.28	98.86	98.59	99.05
LSTM Accuracy	98.64	98.27	97.89	97.69	98.12
CNN Time(s)	4.559	4.570	4.558	4.558	4.561
LSTM Time(s)	0.201	0.200	0.200	0.200	0.200

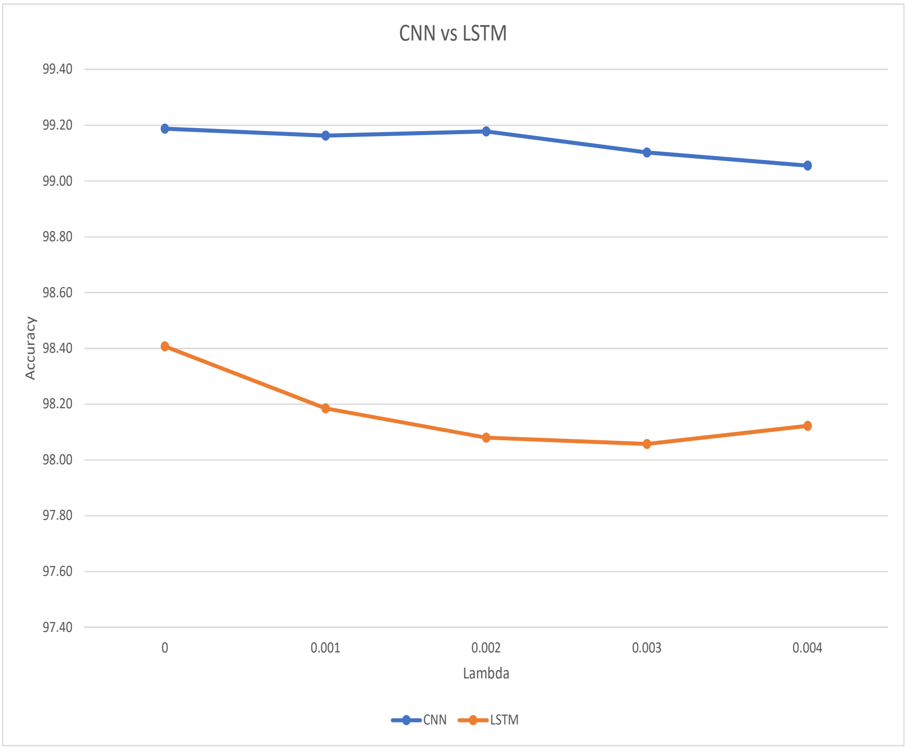

Figure 7.13: Graphical representation of performance of CNN and LSTM models

As can be seen in Figure 7.13 the performance of CNN simply outperforms the performance of LSTM model. The variation in value L2 regularization λ degrades the classification accuracy in both the models.

7.2.3 Evaluation of LSTM model with Variation in Regularization

In this experiment, we present the evaluation of the performance of LSTM model with the variation in the L2 regularization coefficient λ. The regularization is added to loss function which prevents the model from overfitting the training data. L2 regularization of the parameters, encourages all the parameters to be small, instead of being peaky [36]. So the network to pay equal attention to all dimensions of the input vector which leads to smooth network mapping in a neural network.

As can be seen in tables 7.12, 7.13 and 7.14 the classification performance of the LSTM networks falls with the increase in the regularization coefficient λ. We are performing the testing on the uniformly spaced value between 0.0 to 0.1 (0.0, 0.001, 0.002, ... , 0.1). We have used 4-fold cross-validation and show the average accuracies per fold over all the values of λ. When λ is 0.0, then the classification accuracy is maximum which is 98.4%. After that with slight variation, it is decreasing with the increase in the value of λ. At λ 0.1 the classification accuracy is 98.02% which is less than the accuracy at λ 0.0. There is also not so much significant change in the time taken for a single epoch. So we can conclude that increasing the value of λ leads to a significant drop in the performance of the LSTM network.

Table 7.12: Performance of LSTM model with Variation in L2 Regularization Parameter λ

$\lambda = 0.0$					
	CV1	CV2	CV3	CV4	Average
LSTM Accuracy	98.98	98.66	98.06	97.93	98.40
LSTM Time(s)	0.190	0.190	0.190	0.190	0.190
$\lambda = 0.001$					
	CV1	CV2	CV3	CV4	Average
LSTM Accuracy	98.63	98.4	97.92	97.79	98.18
LSTM Time(s)	0.192	0.192	0.192	0.192	0.192

Table 7.13: Performance of LSTM model with Variation in L2 Regularization Parameter λ

$\lambda = 0.002$					
	CV1	CV2	CV3	CV4	Average
LSTM Accuracy	98.52	98.22	97.82	97.76	98.08
LSTM Time(s)	0.192	0.192	0.192	0.192	0.192
$\lambda = 0.003$					
	CV1	CV2	CV3	CV4	Average
LSTM Accuracy	98.63	98.25	97.73	97.62	98.05
LSTM Time(s)	0.195	0.194	0.194	0.194	0.194
$\lambda = 0.004$					
	CV1	CV2	CV3	CV4	Average
LSTM Accuracy	98.64	98.27	97.89	97.69	98.12
LSTM Time(s)	0.201	0.200	0.200	0.200	0.200
$\lambda = 0.005$					
	CV1	CV2	CV3	CV4	Average
LSTM Accuracy	98.7	98.29	97.79	97.61	98.09
LSTM Time(s)	0.201	0.201	0.201	0.200	0.201
$\lambda = 0.006$					
	CV1	CV2	CV3	CV4	Average
LSTM Accuracy	98.61	98.27	97.58	97.65	98.02
LSTM Time(s)	0.203	0.203	0.202	0.202	0.202
$\lambda = 0.007$					
	CV1	CV2	CV3	CV4	Average
LSTM Accuracy	98.67	98.36	97.72	97.55	98.07
LSTM Time(s)	0.201	0.201	0.201	0.201	0.201

Table 7.14: Performance of LSTM model with Variation in L2 Regularization Parameter

$\lambda = 0.008$					
	CV1	CV2	CV3	CV4	Average
LSTM Accuracy	98.64	98.39	97.78	97.51	98.08
LSTM Time(s)	0.201	0.201	0.201	0.201	0.201
$\lambda = 0.009$					
	CV1	CV2	CV3	CV4	Average
LSTM Accuracy	98.64	98.25	97.71	97.53	98.03
LSTM Time(s)	0.202	0.202	0.202	0.202	0.202
$\lambda = 0.1$					
	CV1	CV2	CV3	CV4	Average
LSTM Accuracy	98.66	98.35	97.58	97.5	98.02
LSTM Time(s)	0.202	0.201	0.201	0.201	0.201

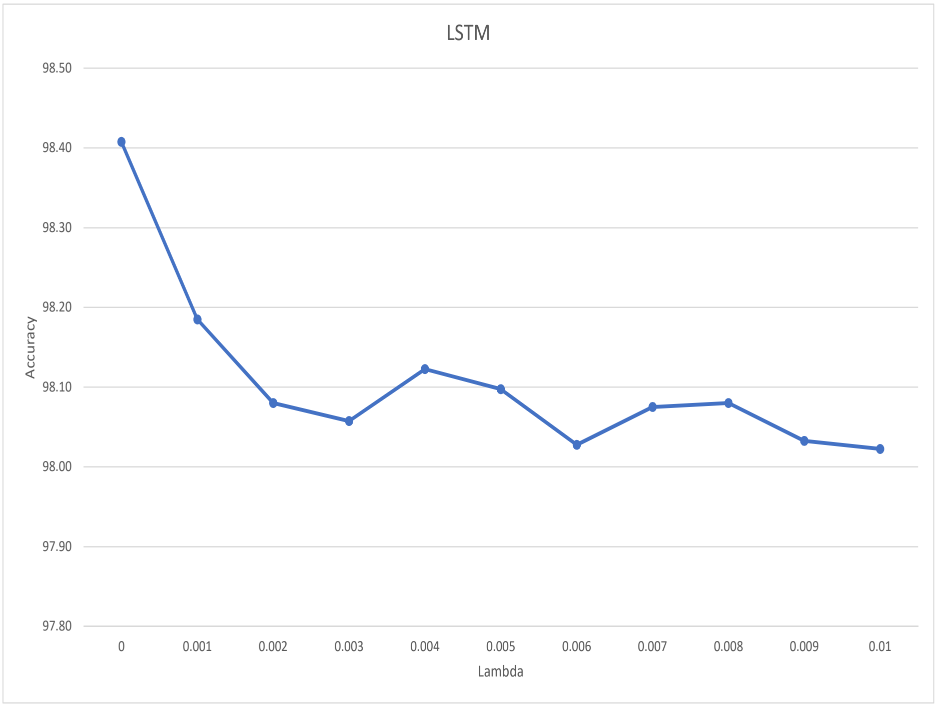

Figure 7.14: Graphical representation of performance of LSTM model with Variation in L2 Regularization Parameter

As can be seen in Figure 7.14 the performance of LSTM can be seen downgrading with the variation in the L2 regularization λ. The best accuracy of LSTM (98.40%) comes when λ is 0.0. After that, the classification accuracy starts decreasing with the increase in the value of λ.

7.2.4 Evaluation of LSTM Model with Variation in Number of Epochs

In the last experiment, we present the evaluation of LSTM network by changing the number of epochs. The previous analyses were performed with a total of 1500 epochs. An epoch can be defined as one forward pass and one backward pass for all training examples. So basically, it describes the total number of time the algorithm has seen the entire dataset.

Table 7.15: Performance of LSTM model with variation in the number of epochs

Epochs = 500					
	CV1	CV2	CV3	CV4	Average
LSTM Accuracy	98.63	98.49	97.22	97.32	97.91
LSTM Time(s)	0.064	0.064	0.064	0.064	0.06
Epochs = 1000					
	CV1	CV2	CV3	CV4	Average
LSTM Accuracy	98.64	98.38	97.8	97.76	98.14
LSTM Time(s)	0.151	0.150	0.148	0.149	0.149
Epochs = 1500					
	CV1	CV2	CV3	CV4	Average
LSTM Accuracy	98.98	98.66	98.06	97.93	98.41
LSTM Time(s)	0.190	0.190	0.190	0.190	0.190
Epochs = 2000					
	CV1	CV2	CV3	CV4	Average
LSTM Accuracy	98.8	98.68	98.14	98.11	98.43
LSTM Time(s)	0.257	0.256	0.256	0.256	0.256
Epochs = 2500					
	CV1	CV2	CV3	CV4	Average
LSTM Accuracy	98.92	98.66	98.2	98.14	98.48
LSTM Time(s)	0.384	0.383	0.382	0.382	0.383

Table 7.15 shows the detail of classification accuracy and average time taken by single epoch for the different number of epochs. We have achieved a classification accuracy of 98.40% with 1500 epochs. When we check the result from 500 to 1500 epochs by uniformly incrementing them by 500 we can to see a small margin of increase in the classification accuracy as can be seen in Figure 7.15. At 500 epochs the accuracy was 97.91% which increased to 98.48% at 2500 epochs. There is a significant increase in time taken by the network to train. At 500 epoch network took an average of 0.064 seconds per epoch to train which increased to 0.383 seconds per epoch at 2500 epochs.

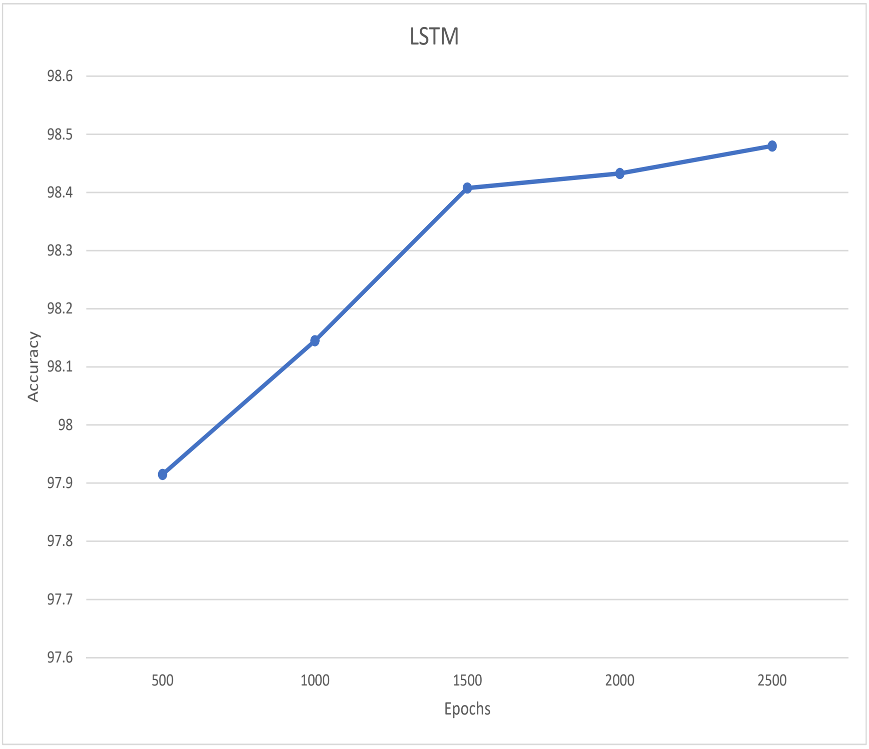

Figure 7.15: Graphical representation of performance of LSTM model with variation in the number of epochs

Chapter 8

Conclusion and Future Work

This thesis has focused on implementing the deep learning algorithms in the medical domain. We have built deep learning models for the classification of the two medical anomalies: diabetes and cardiac arrhythmia.

In the first task, we present a multi-layer feed forward neural network (MLFNN) for classification of diabetes type 2 using the Pima Indian Diabetes (PID) dataset. The MLFNN model outperforms Naïve Bayes (NB), Random Forest (RF) and Logistic Regression(LR) in the classification accuracy. We also discussed how different techniques to handle the missing data plays a crucial role in improving the classification accuracy of the diabetes type 2. The accuracy with removing the instances with missing values outperforms the techniques where we replaced these instances with mean or zero. Furthermore, a comparative study is carried out using different activation units and learning algorithms, which plays a crucial role in solving the problem in the neural network. Since activation units like ELU and SELU do not merely discard negative outputs by neurons, they proved to work better for the PID dataset as compared to ReLU and Leaky ReLU. We also go through learning algorithms like Adam and SGD impact on the performance of MLFNN for the PID dataset. Adam with its adaptive nature outperforms SGD algorithm in the classification accuracy. We conclude that the architectural design plays a crucial role in improving the efficiency of the MLFNN model and needs to choose with careful experimentation, analysis and according to the dataset.

For the future work, we will try to improve accuracy by applying various feature selection techniques. We would also like to test the model with a larger dataset.

In the second task, we classify different types of cardiac arrhythmia using Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) models. We evaluated the performance of CNN and LSTM on classifying arrhythmia. We have avoided any extensive pre-processing in work to gain the in-depth visions of capabilities of CNN and LSTMs. CNN has outperformed LSTM in the classification accuracy. Furthermore, we continued our experiments by incrementing the value L2 regularization coefficient λ from 0.0 to 0.1 at uniform intervals. This resulted in down gradation of classification accuracy of the LSTM network. We also showed how the number of epochs in LSTM impact the classification accuracy. The model we have developed has 1500 epochs and increasing the number increments the performance of LSTM with a very small margin. All-over we have achieved test accuracy that surpasses the other work done by researchers on the MIT dataset.

Since we have implemented minimum preprocessing, for the future work, we will try to apply various pre-processing techniques which can produce the better results. We will also try to extend our work and see if the model can predict any future cardiac arrhythmia. As an extension of this work, we will try to combine CNN and LSTM layers and check if this architectural design can produce a even more accurate result.

References

1. Artificial intelligence neural networks- available at - http://www.w3ii.com/artificial_intelligence/artificial_intelligence_-neural_networks.html.

2. Artificial neural networks-deep learning for java-available at-https://prateekvjoshi.com/2012/08/14/artificial-neural-networks/.

3. Backpropagation - from wikipedia, available at - https://en.wikipedia.org/wiki/backpropagation.

4. A beginner's guide to understanding convolutional neural networks - adit deshpande - available - https://adeshpande3.github.io/adeshpande3.github.io/a-beginner's-guide-to-understanding-convolutional-neural-networks/.

5. Classification and loss evaluation - softmax and cross entropy loss, available at - https://deepnotes.io/softmax-crossentropy.

6. Essentials of deep learning : Introduction to long short term memory - pranjal srivastava , december 10, 2017 - - available - https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/.

7. Everything you need to know about neural networks, https://hackernoon.com/everything-you-need-to-know-about-neural-networks-8988c3ee4491.

8. Global report on diabetes - published by world health organization - available at - https://www.who.int/diabetes/global-report/en/.

9. Implementation of xor gate using multi-layer perceptron/error back propogation, http://vlabs.iitb.ac.in/vlabs-dev/labs/machine_learning/labs/exp2/theory.php.

10. Introduction to deep neural networks (deep learning) - deep learning for java open-source, distributed, deep learning library for the jvm - available at-https://deeplearning4j.org/neuralnet-overview.

11. Log analytics with deep learning and machine learning - by jagreet kaur gill — may 28, 2017 - available at-https://www.xenonstack.com/blog/data-science/log-analytics-with-deep-learning-and-machine-learning.

12. Loss functions and optimization algorithms. demystified, available at - https://medium.com/data-science-group-iitr/loss-functions-and-optimization-algorithms-demystified-bb92daff331c.

13. Mit-bih arrhythmia database - moody gb, mark rg. the impact of the mit-bih arrhythmia database. ieee eng in med and biol 20(3):45-50 (may-june 2001). (pmid: 11446209) - available at - https://www.physionet.org/physiobank/database/mitdb/.

14. Physiobank atm - https://physionet.org/cgi-bin/atm/atm.

15. scikit learn library - sklearn.preprocessing.imputer- - available - http://scikit-learn.org/stable/modules/generated/ sklearn.preprocessing.imputer.html.

16. Understanding activation functions in neural networks, https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0.

17. Understanding lstm networks - posted on august 27, 2015 - colah's blog - available - http://colah.github.io/posts/2015-08-understanding-lstms/.

18. Pima indian dataset, https://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes, 2017.

19. MK Abhinav-Vishwa, SD Lal, and P Vardwaj. Clasification of arrhythmic ecg data using machine learning techniques. *International Journal of Interactive Multimedia and Artificial Intelligence*, 1(4), 2011.

20. U Rajendra Acharya, Hamido Fujita, Oh Shu Lih, Muhammad Adam, Jen Hong Tan, and Chua Kuang Chua. Automated detection of coronary artery disease using different durations of ecg segments with convolutional neural network. *Knowledge-Based Systems*, 132:62–71, 2017.

21. Melissa Aczon, David Ledbetter, L Ho, Alec Gunny, Alysia Flynn, Jon Williams, and Randall Wetzel. Dynamic mortality risk predictions in pediatric critical care using recurrent neural networks. *arXiv preprint arXiv:1701.06675*, 2017.

22. Marios Anthimopoulos, Stergios Christodoulidis, Lukas Ebner, Andreas Christe, and Stavroula Mougiakakou. Lung pattern classification for interstitial lung diseases using a deep convolutional neural network. *IEEE transactions on medical imaging*, 35(5):1207–1216, 2016.

23. American Diabetes Association et al. Standards of medical care in diabetes—2015 abridged for primary care providers. *Clinical diabetes: a publication of the American Diabetes Association*, 33(2):97, 2015.

24. Y Benchaib, Alexis Marcano-Cedeno, Santiago Torres-Alegre, and Diego Andina. Application of artificial metaplasticity neural networks to cardiac arrhythmias classification. In *International Work-Conference on the Interplay Between Natural and Artificial Computation*, pages 181–190. Springer, 2013.

25. Emelia J Benjamin, Michael J Blaha, Stephanie E Chiuve, Mary Cushman, Sandeep R Das, Rajat Deo, J Floyd, M Fornage, C Gillespie, CR Isasi, et al. Heart disease and stroke statistics-2017 update: a report from the american heart association. *Circulation*, 135(10):e146–e603, 2017.

26. Rahime Ceylan, Yüksel Özbay, and Bekir Karlik. A novel approach for classification of ecg arrhythmias: Type-2 fuzzy clustering neural network. *Expert Systems with Applications*, 36(3):6721–6726, 2009.

27. Min Chen, Yixue Hao, Kai Hwang, Lu Wang, and Lin Wang. Disease prediction by machine learning over big data from healthcare communities. *IEEE Access*, 5:8869–8879, 2017.

28. Haibin Cheng, Pang-Ning Tan, Christopher Potter, and Steven Klooster. Detection and characterization of anomalies in multivariate time series. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 413–424. SIAM, 2009.

29. Bertrand Clarke, Ernest Fokoue, and Hao Helen Zhang. *Principles and theory for data mining and machine learning.* Springer Science & Business Media, 2009.

30. Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

31. A Dallali, A Kachouri, and M Samet. Classification of cardiac arrhythmia using wt, hrv, and fuzzy c-means clustering. *Signal Processing: An Int. J.(SPJI)*, 5(3):101–109, 2011.

32. Chaitrali S Dangare and Sulabha S Apte. Improved study of heart disease prediction system using data mining classification techniques. *International Journal of Computer Applications*, 47(10):44–48, 2012.

33. Manab Kumar Das and Samit Ari. Ecg beats classification using mixture of features. *International scholarly research notices*, 2014, 2014.

34. Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, 1996.

35. Dennis G Fryback. Bayes' theorem and conditional nonindependence of data in medical diagnosis. *Computers and Biomedical Research*, 11(5):423–434, 1978.

36. Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

37. UK Prospective Diabetes Study (UKPDS) Group et al. Intensive blood-glucose control with sulphonylureas or insulin compared with conventional treatment and risk of complications in patients with type 2 diabetes (ukpds 33). *The lancet*, 352(9131):837–853, 1998.

38. Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 2017.

39. Tabreer T Hasan, Manal H Jasim, and Ivan A Hashim. Heart disease diagnosis system based on multi-layer perceptron neural network and support vector machine. 2017.

40. Hassan H Haseena, Paul K Joseph, and Abraham T Mathew. Classification of arrhythmia using hybrid networks. *Journal of medical systems*, 35(6):1617–1630, 2011.

41. Douglas M Hawkins. *Identification of outliers*, volume 11. Springer, 1980.

42. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

43. T Jayalakshmi and A Santhakumaran. A novel classification method for diagnosis of diabetes mellitus using artificial neural networks. In *Data Storage and Data Engineering (DSDE), 2010 International Conference on*, pages 159–163. IEEE, 2010.

44. Fei Jiang, Yong Jiang, Hui Zhi, Yi Dong, Hao Li, Sufeng Ma, Yilong Wang, Qiang Dong, Haipeng Shen, and Yongjun Wang. Artificial intelligence in healthcare: past, present and future. *Stroke and Vascular Neurology*, pages svn–2017, 2017.

45. Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. Purdue university, west lafayette, in 47907, usa {jhjin, adundar, euge}@ purdue. edu. *arXiv preprint arXiv:1412.5474*, 2014.

46. Kamer Kayaer and Tulay Yıldırım. Medical diagnosis on pima indian diabetes using general regression neural networks. In *Proceedings of the international conference on artificial neural networks and neural information processing (ICANN/ICONIP)*, pages 181–184, 2003.

47. Mehdi Khashei, Saeede Eftekhari, and Jamshid Parvizian. Diagnosing diabetes type ii using a soft intelligent binary classification model. *Review of Bioinformatics and Biometrics*, 2012.

48. Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, pages 972–981, 2017.

49. Igor Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine*, 23(1):89–109, 2001.

50. Santosh Kumar and A Kumaravel. Diabetes diagnosis using artificial neural network. *International Journal of Engineering Sciences & Research Technology*, pages 1642–1644, 2013.

51. Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

52. Kevin P Murphy. Naive bayes classifiers. *University of British Columbia*, 18, 2006.

53. A Muthuchudar and Lt Dr S Santosh Baboo. A study of the processes involved in ecg signal analysis. *International Journal of Scientific and Research Publications*, 3(3):1–5, 2013.

54. Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

55. Jalal A Nasiri, Mahmoud Naghibzadeh, H Sadoghi Yazdi, and Bahram Naghibzadeh. Ecg arrhythmia classification with support vector machines and genetic algorithm. In *Computer Modeling and Simulation, 2009. EMS'09. Third UKSim European Symposium on*, pages 187–192. IEEE, 2009.

56. Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848, 2002.

57. Ebenezer Obaloluwa Olaniyi and Khashman Adnan. Onset diabetes diagnosis using artificial neural network. *International Journal of scientific and engineering research*, 5(10), 2014.

58. Rashidah Funke Olanrewaju, Nur Syarafina Sahari, Aibinu A Musa, and Nashrul Hakiem. Application of neural networks in early detection and diagnosis of parkinson's disease. In *Cyber and IT Service Management (CITSM), 2014 International Conference on*, pages 78–82. IEEE, 2014.

59. Mahesh Pal. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222, 2005.

60. Sadri Sa'di, Ramin Hashemi, Arman Abdollapour, Kamal Chalabi, and Mohammad Amin Salamat. A novel probabilistic artificial neural.

61. Sadri Sa'di, Amanj Maleki, Ramin Hashemi, Zahra Panbechi, and Kamal Chalabi. Comparison of data mining algorithms in the diagnosis of type ii diabetes. *International Journal on Computational Science & Applications (IJCSA)*, 5(5):1–12, 2015.

62. Robert J Schalkoff. *Artificial neural networks*, volume 1. McGraw-Hill New York, 1997.

63. Zahed Soltani and Ahmad Jafarian. A new artificial neural networks approach for diagnosing diabetes disease type ii. *International Journal of Advanced Computer Science & Applications*, 1(7):89–94, 2016.

64. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

65. Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

66. Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62, 1997.

67. X Tang and L Shu. Classification of electrocardiogram signals with rs and quantum neural networks. *International Journal of Multimedia and Ubiquitous Engineering*, 9(2):363–372, 2014.

68. Laura Toloşi and Thomas Lengauer. Classification with correlated features: unreliability of feature ranking and solutions. *Bioinformatics*, 27(14):1986–1994, 2011.

69. M Vijayavanan, V Rathikarani, and P Dhanalakshmi. Automatic classification of ecg signal for heart disease diagnosis using morphological features. *Int. J. of Comput. Sci. and Eng. Technology (IJCSET)*, 5(4):449–455, 2014.

70. Shui-Hua Wang, Yin Zhang, Yu-Jie Li, Wen-Juan Jia, Fang-Yuan Liu, Meng-Meng Yang, and Yu-Dong Zhang. Single slice based detection for alzheimer's disease via wavelet entropy and multilayer perceptron trained by biogeography-based optimization. *Multimedia Tools and Applications*, pages 1–25, 2016.

71. Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.

72. Sung-Nien Yu and Kuan-To Chou. Integration of independent component analysis and neural networks for ecg beat classification. *Expert Systems with Applications*, 34(4):2841–2846, 2008.

73. Behnam Zebardast, Rahim Rashidi, Taha Hasanpour, and Farhad Soleimanian Gharehchopogh. Artificial neural network models for diagnosing heart disease: a brief review. *International Journal of Academic Research*, 6(3):73–78, 2014.

74. Zahia Zidelmal, Ahmed Amirou, Djaffar Ould-Abdeslam, and Jean Merckle. Ecg beat classification using a cost sensitive classifier. *Computer methods and programs in biomedicine*, 111(3):570–577, 2013.

Appendices

A Hardware Specification

Operating System	Ubuntu 17.10
Processor	Intel i7-7700K
Memory	64GB DDR4
Graphics	1x NVIDIA GeForce 1080TI

Part IV

Publications

A Multi-Layer Feed Forward Neural Network Approach for Diagnosing Diabetes

1st Micheal Dutt
Centre for AI Research (CAIR)
University of Agder
Grimstad, Norway
miched16@student.uia.no

2nd Vimala Nunavath
Centre for AI Research (CAIR)
University of Agder
Grimstad, Norway
vimala.nunavath@uia.no

3rd Morten Goodwin
Centre for AI Research (CAIR)
University of Agder
Grimstad, Norway
morten.goodwin@uia.no

Abstract—**Diabetes is one of the worlds major health problems according to the World Health Organization. Recent surveys indicate that there is an increase in number of diabetic patients resulting in increase in serious complications such as heart attacks and deaths. Early diagnosis of diabetes, particularly of type 2 diabetes, is critical since it is important for patients to get insulin treatments. However, diagnoses could be difficult especially in areas with few medical doctors. It is therefore a need for practical methods for the public for early detection and prevention with minimal intervention from medial professionals.**

A promising method for automated diagnosis is the use of artificial intelligence and in particular artificial neural networks. This paper presents an application of Multi-Layer Feed Forward Neural Networks (MLFNN) in diagnosing diabetes on publicly available Pima Indian Diabetes data set. A series of experiments are conducted on this data set with variation in learning algorithms, activation units, techniques to handle missing data and their impact on diagnosis accuracy is discussed. Finally, the results are compared with other state of arts methods reported in literature review. The achieved accuracy is 82.5% best of all related studies.

I. Introduction

Diabetes is one of the most common metabolic disorders in the world and prevalence of diabetes in adults has been been rapidly increasing [7]. Diabetes can be classified into two categories: type 1 and type 2. Type 1 diabetes is caused due to β-cell destruction, usually leading to absolute insulin deficiency. Type 2 diabetes is due to progressive insulin secretory defect on the background of insulin deficiency [1]. Generally when the amount of glucose in the blood is high, the performance in different body organ is defected. If the disease in such condition is not timely diagnosed, then it may lead to heart attacks, strokes, kidney failures and blindness [6]. In this paper, we focus on diagnosing diabetes type 2. In many cases, the diagnosis is generally based on patient test results and physician's experience. Thus, diagnosis is a complex task requiring high skills and experience. Early diagnosis and medical care of patients can greatly reduce the problems of patients [19].

A lot of machine learning and AI based solutions have been proposed for different disease so far [**3, 18, 4**]. In fact, an increasing number of medical devices available with embedded AI algorithm that are available on market manifest this [8]. Artificial Neural Networks (ANNs) are characterized as computational models which have inherent massively parallel-distributed architectures. The design of ANNs was motivated from the structure of real brain, but processing elements (neurons) and the architecture used in ANN have exceeded far from its biological inspiration. When we apply ANN to identify a disease, the main goal is to achieve high accuracy rate [19]. With the advancement in AI, a lot of activation units and learning algorithms are proposed or are implemented to increase the accuracy of tasks, that they need to solve. Before starting the task of diagnosis, the ANN model must be trained using patients data sets. After training and testing the patients data sets using models of ANN, the way of achieving the higher accuracy and minimum error rate is provided [17]. In this paper, our aim is to evaluate four different activation units and two learning algorithms and try to achieve maximum accuracy during training and testing phase of diagnosing diabetes type 2 using MLFNN models. The data set named Pima Indian Diabetes with 768 samples is used. For distinguishing the performance, we used testing and training accuracy as a performance measure. Furthermore, we implement the model of MLFNN for diagnosing diabetes in Keras.

The remainder of paper is organized in following way. First, we discuss some related works in the field of ANN which have been preciously done in diagnosing diabetes type 2 in section 2. In section 3, we discuss about the features, attributes and the other aspects of publicly available Pima Indian data set.In section 4 we discuss the experimental setup. It is further subdivided into the architecture we use, different activation functions and learning algorithms. In section 5, we discuss the results. Since, we have performed three different experiments, so all of the results are provided and discussed in the subsections. In Section 6, we conclude this paper and suggest some key points which needs to be considered while developing any machine learning model.

II. Literature Review

Significant researches have been done till now in the context of diagnosis of diabetes using artificial neural network. In this section, we study some of these studies, and then compare them with testing and training accuracy in the process of diagnosis of diabetes type 2.

In [9] use General Regression Neural Networks (GRNN) and Pima Indian Diabetes data set for identifying diabetes. In this paper, GRNN model is assumed to be four layers: input layer with 8 features form data set, two hidden layers with 32 and 16 neurons, respectively. Finally, there is single neuron in output layer, which determines whether patient is diabetic or not. Out of total 768 samples, 75% and 25% of samples are used for training and testing process. The accuracy achieved with training and testing phase are 82.99% and 80.21%, respectively.

In [14] use Multilayer Artificial Neural Network with back propagation for diagnosing diabetes. In backpropagation, neural network compares computed output value with actual value and calculates the error. The weights are adjusted in each round in such a way that the calculated error to be always less than the previous round. In this paper, network consist of input layer with 8 neurons, hidden layer with 6 neurons and 2 neurons in output layer. The data set contains a total 768 samples out of which 500 samples are used during training and remaining 268 during testing phase. The achieved diagnosis accuracy after 2000 rounds of data set training becomes 82%.

In [17] use Probability Neural Network (PNN) for diagnosing diabetes. In this paper, PNN model consist of input layer with 8 neurons representing each feature, single hidden layer, and output layer with two neurons to diagnose whether a patient is diabetic or not. The data set which consist of 768 samples, 90% of samples are used in training and remaining 10% during testing phase. The achieved training and testing accuracy is 81.49% and 89.56% respectively after 200 rounds.

In [16] use different data mining techniques such as Naïve Bayes, J48 and Radial Based Artificial Neural Network for diagnosing diabetes type 2. They used Pima Indian data set with 768 data samples out of which 268 samples were used during testing phase. Naïve Bayes proved to be more efficient with 76.95% accuracy followed by J48 with 76.5% and RBF with 74.34% accuracy's, respectively.

In [11] use a data set with 250 data samples consisting of 27 features. These features also include blood pressure, creatinine, urine PH, fasting glucose. The average age of patients is between 25 to 78 years. They use Multilayer feed-forward artificial neural networks with backpropagation for diagnosis. Three training functions are applied in backpropagation algorithm namely BFGS Quasi-Newton, Bayesian Regulation and Levenberg-Marquardt. Finally, backpropagation with Bayesian Regulation function achieved highest of 88.8% of diagnosing accuracy performing better than BFGS Quasi-Newton and Levenberg-Marquardt functions.

III. Dataset Description

In this study, the Pima Indian Diabetes Data set (PIDD) is taken from University of California, Irvine (UCI) repository of machine learning database [15]. This data set is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. All woman in this database are of at least 21 years old of Pima Indian heritage and was tested for diabetes according to World Health Organization criteria. This data set is used in various researches to build classification system. This is the main reason for choosing this data set as we can compare our study with various other researches for Pima Indian Diabetes diagnosing problem. This data set is composed of a total of 768 instances. Each instance in the data set is characterized by 8 features. All the features have numerical values as seen in table I.

Table I
Attributes/Features of PIMA Indian Dataset

Attribute/Features	Types/Values
Number of times pregnant	Numerical Values
Plasma Glucose Concentration	Numerical Values
Diastolic Blood Pressure	Numerical Values (in mmHg)
Triceps skin fold thickness	Numerical Values (in mm)
2-Hour Serum insulin	Numerical Values (in μU/ml)
Body mass index	Numerical Values (in kg/m^2)
Diabetes pedigree function	Numerical Values
Age	Numerical Values (in years)

The last feature has binary value and is used for classification as it is divided into two classes: Class Zero (Non-diabetic) and Class One (Diabetic). The first 8 features are used as input and the last feature is only output. This data set also contains a significant portion of missing data as shown in table II.

Table II
Missing Data in PIMA Indian Data set

No.	Attribute/Features	Types/Values
1	Number of times pregnant	-
2	Plasma Glucose Concentration	5
3	Diastolic Blood Pressure	35
4	Triceps skin fold thickness	227
5	2-Hour Serum insulin	374
6	Body mass index	11
7	Diabetes pedigree function	1
8	Age	63

The total number of Diabetic cases are 268 which corresponds to 34.90% of total cases. The number of Non-diabetic cases are 500 (65.10%).

IV. Approach

In this paper, we have used special type of ANN called multi-layer feed forward neural network. The main goal of a multi-layer feed forward network is to approximate some function f^*. As for an example, in a classifier problem, $y = f^*(x)$ maps an input x to a label y.

An artificial neural network is typically specified by following things [16]:

- *An architecture:* the width and depth of network.
- *A learning algorithm:* for updating the weight to model a task correctly.
- *An activation unit:* to transform a neuron's weighted input to its output activation.

The same architecture is used in all experiments as discussed below, but with variations in activation function, learning algorithm and techniques to handle missing data. All the activation functions and learning algorithms used in these experiments have been formally introduced below. We have used cross entropy loss function in all the experiments. It is most commonly used in binary classification problem. Cross entropy is computed by:

$$H_{y'}(y) = -\sum_i y'_i \log(y_i)$$

Cross entropy measures the divergence between the two probabilities distribution. If the cross entropy is large it means the difference between the two distribution is large. If the cross entropy is small, it means the two distributions are like each other. For the output layer sigmoid activation function is used in all the experiments but for the hidden layers activation units varies w.r.t the experiments.

A. Network Architecture

A key design consideration of a feed forward neural network is to determine the overall structure of the network. It includes choosing the depth of the network and the width of each layer. The depth of network means the number the hidden layer and the width corresponds to the number of neurons in the layer. A feed forward network with only single layer can represent any function but the layer may be infeasible large and may fail to learn or generalize correctly. In contrast, the deep models can significantly reduce the number of units per layer and the generalization error. Another key consideration in designing the network is the connections between the layers. If we decrease the number of connections, we can reduce the number of parameters and the amount of computation that is required to evaluate the network. These all architectural considerations are highly problem dependent. The ideal network architecture for a task must be found by doing repeated experimentation and monitoring the evaluation metrics.

We have designed the network with input layer (8 Neurons), three hidden layers with 50 neurons each and an output layer with single neuron.

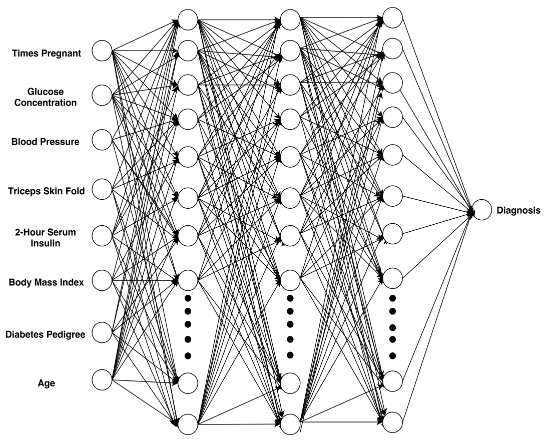

Figure 1. Network Architecture

B. Learning Algorithms

In this section, we introduce two different types of learning algorithms: *stochastic gradient descent (SGD) and Adam.* In the following subsections, we introduce both learning algorithms.

1) Stochastic Gradient Descent (SGD): Stochastic Gradient Descent are probably the most used learning algorithm for machine learning and particularly for deep learning [5]. The standard gradient descent algorithms update the parameters approximated by evaluating the cost and gradient over the full training set. The SGD updates and computes the gradient of parameters using single or few training examples. The parameter update is given by:

$$[P = P - LR * gradients] \tag{1}$$

P stands for Parameters and LR stands for Learning Rate.

The benefit for updating the parameters based on few training examples is, it reduces the variance in the parameter update and lead to a stable convergence. The crucial parameter of SGD is learning rate which must adjusted with lot of trial and error. In this study, we have used default learning rate which is 0.01.

2) Adam: Adam is an adaptive learning rate optimization algorithm which derives its name from phrase "adaptive moments" [5]. Adam algorithm is based in estimation of 1st and 2nd order moments. The algorithm estimates "first moment" as the mean and the "second moment" as variance. So the update rule Adam is:

$$[P = P - LR * Means/Sqrt(Variance)] \tag{2}$$

Here, P stands for Parameters and LR stands for Learning Rate.

So if the variance of gradient is high, it becomes unclear how parameter should be changed so algorithm chooses the small step size in update rule. If the variance is low the algorithm takes a larger step. Adam also includes bias corrections to estimate both the mean and variance to account for their initialization at the origin.

C. Activation Units

In this section, we introduce the four kinds of activation unit: rectified linear unit (ReLU), leaky rectified linear (Leaky ReLU), exponential linear unit (ELU) and scaled exponential linear unit (SELU). In the following subsections, we introduced each activation unit formally.

1) ReLU: The Rectified Linear Unit is first used in Restricted Botlzmann Machines [13]. It is the simplest nonlinear activation function and is defined as:

$$f(x) = \begin{cases} 0 & \text{for} \quad \text{x} < 0 \\ \text{x} & \text{for} \quad \text{x} \geqslant 0 \end{cases}$$

2) Leaky ReLU: Leaky ReLU activation is first introduced in acoustic model [12]. These are one attempt to fix the dying ReLU problem. Mathematically, it is defined as:

$$f(x) = \begin{cases} \alpha\text{x} & \text{for} \quad \text{x} < 0 \\ \text{x} & \text{for} \quad \text{x} \geqslant 0 \end{cases}$$

where $\alpha = 0.01$.

3) ELU: The exponential linear unit (ELU) [2] with $0 < \alpha$ is

$$f(x) = \begin{cases} \alpha(\text{exp(x)-1}) & \text{for} \quad \text{x} < 0 \\ \text{x} & \text{for} \quad \text{x} \geqslant 0 \end{cases}$$

The ELU hyperparameter α controls the value to which an ELU saturates for negative net inputs.

4) SELU: The SELU activation function [10] is given by:

$$f(x) = \lambda \begin{cases} \alpha(\text{exp(x)-1}) & \text{for} \quad \text{x} < 0 \\ \text{x} & \text{for} \quad \text{x} \geqslant 0 \end{cases}$$

SELU allows to construct a mapping with properties that lead to Self-Normalizing Neural Networks (SNN). SNN keeps normalization of activation's when propagating through layers. We consider activation's of neural network to be normalized, if both their mean and their variance across samples are within predefined intervals.

V. Results and Discussion

This section presents a comparison of our network by applying four different activation units, followed by an experiment with comparison of two learning algorithms on Pima Indian data set. Finally, we present experiment from approaches to handle missing data.The over all aim is to find the combination to achieve the highest classification accuracy.

A. Activation Units

In our first experiment, we compare classification performance of four different activation units with Adam as a learning algorithm. Table III show the performance results of four activation units we have considered for the experiment.

Table III
Results With Variation In Activation Functions

	ELU	SELU	Leaky ReLU	ReLU
Training Accuracy%	83.91	87.11	92.63	98.98
Testing Accuracy%	82.50	82.05	81.15	78.73
Training MSE	0.111	0.094	0.057	0.011
Testing MSE	0.131	0.132	0.143	0.178

The results show that the ELU gives the highest accuracy of 82.5% followed by SELU (82.05%), Leaky ReLU (81.15%) and ReLU (78.73%). ELU and SELU have improved learning characteristics as compared with other two activations' for Pima Indian dataset. In contrast with ReLU, ELU does not simply discard the neurons with negative values, which allows them to push mean unit activation closer to zero like batch normalization. ELU activation is more robust to noise whereas Leaky ReLU, which also consider negative values do not ensure noise robust deactivation state. SELU allows to construct self-normalizing neural networks. It pushes activation's to zero mean and unit variance, leading to same effect as batch normalization, avoiding exploding and vanishing gradient.

On training set, the mean squared error of ReLU (0.011362) is lowest followed by Leaky ReLU (0.057298). The error of other two activations ELU and SELU are 0.111245 and 0.094413 respectively. This indicates that both ReLU and Leaky ReLU may suffer from overfitting issues, since the Pima Indian data set is small, and the network is complex []. It indicates for Pima Indian dataset. The proposed neural network produces improved effectiveness with ELU and SELU since it is combating overfitting.

B. Learning Algorithms

In our second experiment, we study the impact of two different learning algorithm on classification performance on our data set.

Table IV
Results With Variation In Learning Algorithms

	Adam	SGD
Training Accuracy%	83.91	78.93
Testing Accuracy%	82.5	78.43
Training MSE	0.111	0.140
Testing MSE	0.131	0.146

Table IV shows the classification performance of Stochastic gradient descent and ADAM algorithm using ELU as an activation function on Pima Indian data set.

Not surprisingly, we find the performance of Adam (82.5%) better then SGD (78.4%). Since for the SGD, adjusting learning rate is one of the crucial parameter and we are using the default learning rate and it is taking predefined steps in order to reduce the error. Whereas, Adam is cleverly adjusting the learning rate w.r.t variance in gradient values it encounters.

C. Handling Missing Data

In our last experiment, we use three different approaches to deal with missing values in the data set and compare the classification performances. The first approach and easiest way to deal with missing values is to remove the instances with missing values. This technique may lead to the loss of highly valuable information but is very useful when the database is highly skewed. There are total of 768 instances but after removing instances with missing values we are left with only 392 instances. The second approach is to replace all missing values with zeros. The third technique is to impute all missing values with mean. In this method the missing value of an attribute is replaced by average of all the available values of the same attribute in the data.

Table V
Results With Variation In Techniques To Handle Missing Data

	Remove samples	Replace (mean)	Replace (zero)
Training Accuracy%	83.91	80.23	81.00
Testing Accuracy%	82.50	79.05	78.93
Training MSE	0.111	0.131	0.130
Testing MSE	0.131	0.145	0.149

Table V shows the classification performance of all the three approaches. The testing accuracy after removing the sample (82.5%) is superior then replacing the missing values with mean (79.05%) followed by replacing with zero (78.93%). The reason behind this behaviour is that sometimes replacing values make it hard for the neural network to properly adjust the impact of certain features which normally plays crucial role in solving the problem on the output.

VI. Conclusion

This paper introduces a neural network approach for classification and in turn diagnoses of diabetes type 2 using the Pima Indian data set. Further, a comparative study is carried out using different activation units and learning algorithms, which plays crucial role in solving the problem in neural network. Since activation units like ELU and SELU do not simply discard negative outputs by neurons, they proved to work better for our data as compared to ReLU. This paper also looks through how learning algorithms like Adam and SGD impact the neural network for a particular data set. Adam with its adaptive nature simply outperforms SGD algorithm for the Pima Indian data set. Moreover, in this paper we also discussed how different techniques to counter the missing data are playing role in diagnosing the diabetes type 2. The accuracy with removing the instances with missing values simply outperforms the techniques where we replaced these instances with mean or zero. So, this paper simply concludes that every architectural design plays crucial role in improving the accuracy of the model and needs to choose with careful experimentation's, analysis and according to the data set. Our neural network is able to correctly detect diabetes in 82.5% in the validation set after removing samples with missing values when using ELU activation unit and the Adam as learning algorithm.

References

[1] American Diabetes Association et al. "Standards of medical care in diabetes—2015 abridged for primary care providers". In: *Clinical diabetes: a publication of the American Diabetes Association* 33.2 (2015), p. 97.

[2] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. "Fast and accurate deep network learning by exponential linear units (elus)". In: *arXiv preprint arXiv:1511.07289* (2015).

[3] Joseph A Cruz and David S Wishart. "Applications of machine learning in cancer prediction and prognosis". In: *Cancer informatics* 2 (2006), p. 117693510600200030.

[4] Rahul C Deo. "Machine learning in medicine". In: *Circulation* 132.20 (2015), pp. 1920–1930.

[5] Ian Goodfellow et al. *Deep learning.* Vol. 1. MIT press Cambridge, 2016.

[6] UK Prospective Diabetes Study (UKPDS) Group et al. "Intensive blood-glucose control with sulphonylureas or insulin compared with conventional treatment and risk of complications in patients with type 2 diabetes (UKPDS 33)". In: *The lancet* 352.9131 (1998), pp. 837–853.

[7] Leonor Guariguata et al. "Global estimates of diabetes prevalence for 2013 and projections for 2035". In: *Diabetes research and clinical practice* 103.2 (2014), pp. 137–149.

[8] Fei Jiang et al. "Artificial intelligence in healthcare: past, present and future". In: *Stroke and Vascular Neurology* (2017). ISSN: 2059-8688. DOI: 10.1136/svn-2017-000101.

[9] Kamer Kayaer and Tulay Yıldırım. "Medical diagnosis on Pima Indian diabetes using general regression neural networks". In: *Proceedings of the international conference on artificial neural networks and neural information processing (ICANN/ICONIP).* 2003, pp. 181–184.

[10] Günter Klambauer et al. “Self-normalizing neural networks”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 972–981.

[11] Santosh Kumar and A Kumaravel. “Diabetes Diagnosis using Artificial Neural Network”. In: *International Journal of Engineering Sciences & Research Technology* (2013), pp. 1642–1644.

[12] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. “Rectifier nonlinearities improve neural network acoustic models”. In: *Proc. icml*. Vol. 30. 1. 2013, p. 3.

[13] Vinod Nair and Geoffrey E Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[14] Ebenezer Obaloluwa Olaniyi and Khashman Adnan. “Onset diabetes diagnosis using artificial neural network”. In: *International Journal of scientific and engineering research* 5.10 (2014).

[15] *PIMA dataset*. 2017. URL: https://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes.

[16] Sadri Sa’di et al. “Comparison of data mining algorithms in the diagnosis of type II diabetes”. In: *International Journal on Computational Science & Applications (IJCSA)* 5.5 (2015), pp. 1–12.

[17] Zahed Soltani and Ahmad Jafarian. “A New Artificial Neural Networks Approach for Diagnosing Diabetes Disease Type II”. In: *International Journal of Advanced Computer Science & Applications* 1.7 (2016), pp. 89–94.

[18] Nooritawati Md Tahir and Hany Hazfiza Manap. “Parkinson Disease Gait Classification based on Machine Learning Approach”. In: *Journal of Applied Sciences* 12.2 (2012), pp. 180–185.

[19] Behnam Zebardast et al. “Artificial neural network models for diagnosing heart disease: a brief review”. In: *International Journal of Academic Research* 6.3 (2014), pp. 73–78.