UiA University of Agder

Master

Classification of GNSS Jammers using Machine Learning

Multivariate Time Series and Image Classification Based Approaches

JAKOB MICHAEL VOIGT

SUPERVISORS
Lei Jiao, Associate Professor UiA
Andreas Taraldsen, Egde

University of Agder, 2021
Faculty of Engineering and Science
Department of Engineering and Sciences

Obligatorisk gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. Manglende erklæring fritar ikke studentene fra sitt ansvar.

1.	Jeg erklærer herved at vår besvarelse er eget arbeid, og at jeg ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.	Ja
2.	**Jeg erklærer videre at denne besvarelsen:** • Ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands. • Ikke refererer til andres arbeid uten at det er oppgitt. • Ikke refererer til eget tidligere arbeid uten at det er oppgitt. • Har alle referansene oppgitt i litteraturlisten. • Ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	Ja
3.	Jeg er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§ 31.	Ja
4.	Jeg er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert.	Ja
5.	Jeg er kjent med at Universitetet i Agder vil behandle alle saker hvor det forligger mistanke om fusk etter høgskolens retningslinjer for behandling av saker om fusk.	Ja
6.	Jeg har satt meg inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider.	Ja
7.	Vi har i flertall blitt enige om at innsatsen innad i gruppen er merkbart forskjellig og ønsker dermed å vurderes individuelt. Ordinært vurderes alle deltakere i prosjektet samlet.	NA

Publiseringsavtale

Fullmakt til elektronisk publisering av oppgaven Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven. §2).
Oppgaver som er unntatt offentlighet eller taushetsbelagt/konfidensiell vil ikke bli publisert.

Vi gir herved Universitetet i Agder en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering:	Ja
Er oppgaven båndlagt (konfidensiell)?	Nei
Er oppgaven unntatt offentlighet?	Nei

Acknowledgements

To all who have helped and me in this process, I am ever grateful!

Thanks to UiA and Lei Jiao for all guidance throughout this thesis and my ICT master's degree.

Thanks to Edge for lending Andreas Taraldsens time, for all his clever insights and practical recommendations.

Thanks to NKOM, in particular Espen Bekkelien and Nicolai Gerrard, for pleasant and effective communication throughout the project.

For all the love and support from my wife, family, and friends, especially to Amund Faller Råheim for his invaluable discussions and editorial help. Thank you all very much!

Nasjonal kommunikasjons-myndighet

This project is a contnuation of ICT Seminar 3, IKT442-G

Abstract

GNSS is one of the most widely used positioning techniques in modern technology. However, the signals from the GNSS satellites are weak on earth due to the propagation attenuation, making GNSS-based systems vulnerable to interference or jamming of signals. This thesis proposes a machine learning approach to detect the presence of jammers in the GNSS spectrum bands in recorded data. We have employed the state-of-the-art and baseline machine learning techniques for Image- and multivariate time series classification and evaluated their ability to classify the presence of illegal jammer activity. In addition, we propose a novel complexity reduced version of a recently proposed multivariate time series transformer model. Experiment results show that the tested machine learning techniques, after proper configurations, achieve a classification accuracy of up to 99.5%. Moreover, the simplified transformer-based approach achieves the same level of performance while reducing the number of parameters by nearly half compared to comparable artificial neural network models. The high accuracy confirms the applicability of the machine learning approach in jammer classification.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The use of Navigation Satellite Systems (GNSS) in applications is growing [56]. With the increased use of GNSS positioning for public services, such as road toll for heavy transport, GNSS signal jammers in vehicles are expected to increase proportionally. The use of radio frequency signal jammers is illegal, and monitoring and tracking such activity in Norway is under the jurisdiction of Nkom.

Currently, Nkom uses a process of manually annotating potential jammer events from recorded radio frequency data. They have labeled historical data and are interested in automating some of the annotation process. For this purpose, we will investigate the use of state-of-the-art machine learning methods for the classification of time series and image data.

1.1 Introduction to the Domain of GNSS Interference

In the field of signal processing, a signal is a carrier of information through a medium, from a transmitter to a receiver [53]. The connecting medium is commonly either a wired connection (such as an Ethernet cable) or a wireless connection. Communication over wireless connections utilizes electromagnetic (EM) radiation to encode information. The frequencies of these EM waves are within the radio spectrum (30Hz to 300GHz) [22]. Ranges of frequencies may be referred to as frequency bands and are continuous ranges defined by a lower and an upper frequency bound.

Global Navigation Satellite System (GNSS) is any system that uses position and timing data from satellites. The satellites send EM signals to GNSS receivers on earth, providing Position, Navigation, and Timing (PNT) services to these receivers [1]. GNSS receivers are, for example, present in most modern computers, phones, and cars. There are several GNSS systems, including GPS, Galileo, GLONASS, and BeiDou [56]. Each system uses its own allocated frequency bands within the *L Band* from 1 GHz to 2 GHz [31].

Radio frequency interference (RFI) is when two signals of equal frequency coexist at the same space and at the same time. If this happens in the proximity of a receiver, it will create disruptions and cause reception problems. RFI can occur naturally or be created by human activity [56]. *GNSS jammers* are an example of the latter. They are devices that cause disruptions to PNT services in their vicinity by broadcasting strong EM waves of the same frequencies used by the GNSS applications. Due to the jammers' relative proximity to GNSS receivers, they overpower the EM waves from GNSS satellites [9].

1.2 Motivation

The use of GNSS jammers is increasing [45] globally, and there has been an exponential growth in reported GPS outages in recent years [23].

At the same time, the use of GNSS signals has increased in a variety of modern applications [56]. An example is medical aid helicopters which depended on reliable GPS access. A jammer-caused GNSS disruption could have severe consequences for such critical services.

GNSS jammers are also being used in Norway. A national decrease in jammers was observed in 2019 [56], likely due to media attention on the topic Appendix A. However, we expect the use of jammers to increase again with the seemingly imminent advent of satellite-based road toll for heavy transportation [61]. Many European countries have already implemented GPS-based tolling systems, and The Ministry of Transport and Communications, together with the Ministry of Finance and the Ministry of Climate and Environment, has issued a 161-page report concluding such a system would have substantial benefits [60]. There is also some positive political sentiment for implementing GPS-based systems for civilian traffic toll [47]. Naturally, this would further incentivize the use of GNSS jammers, as drivers could use them to circumvent the tolling mechanism. Therefore, GPS-based toll would further increase the demand for monitoring and regulating the radio frequency spectrum.

The Norwegian Communications Authority (Nkom) states on their website that, "Nkom is an executive, supervisory and administrative authority for postal and electronic communication services in Norway" [48]. They are responsible for surveying interference in the GNSS bands and identifying illegal RFI events such as jammers. Currently, experts from Nkom have to manually process and label RFI events captured by monitoring stations across Norway, which is tedious work.

In alignment with the Norwegian governments' strategy to search for opportunities to use machine learning in the public sector [2], this thesis aims to investigate the potential of machine learning methods for jammer detection, and to provide recommendations on the implementation of a system based on these methods.

Automatic detection of illegal jammer activity would increase the chances of apprehending the perpetrators due to faster response times. Another benefit would be freeing up valuable human resources currently tied down by this manual processing.

For further reading we have included a number of news articles in Appendix A, Newsarticles on Jamming.

1.3 Problem Statement

The over-arching goal of this thesis is to detect the presence of GNSS jammers in recorded RFI events using machine learning. Nkom will provide the raw data files and corresponding labels. We will analyze the data and establish appropriate pre-processing techniques to prepare the events for the relevant machine learning classifiers.

We aim to treat the jammer detection problem respectively as a multivariate time series classification task, and an image classification task. By evaluating the state-of-the-art and commonly used approaches in the machine learning literature, we will advise Nkom in their implementation of AI-based solutions to handle RFI events. Our emphasis will be on finding a sufficiently effective solution while minimizing the requirements for computational

resources.

1.3.1 Thesis Goals

We summarize our aspirations by the following goals:

1. Evaluate the state-of-the-art in multivariate time series classification, image classification, and machine learning applied to signal interference management.
2. Achieve a 99% detection accuracy in jammer detection.
3. Achieve a 100% recall on jammer detection while maintaining a 95% accuracy.
4. Improve upon existing algorithms by reducing complexity.

1.3.2 Hypothesis

With this thesis, we will investigate the following hypothesis.

1. There exists patterns in jammer-caused RFI events that are distinguishable from other RFI interference.
2. Machine learning algorithms can learn these patterns to detect the presence of jammers in recorded RFI events.

1.4 Contributions

Based on the literature review of this thesis, we observe a discrepancy between the state-of-the-art in multivariate time series classification and machine learning methods applied to jammer detection. This thesis connects these domains by testing several of the best-performing models from the field of multivariate time series classification. We also tested numerous image classification models of varying complexity, whereas previous research in this field has commonly only used simple CNN and SVM-based models [45][72][25]. Finally, we implemented a complexity-reduced version of a recently developed Transfomer designed for time series classification.

1.5 Paper Outline

In Chapter 2, Theoretical background and State-of-the-art, we will introduce the theoretical framework for machine learning, and introduce the problem domain in more detail. We also present related work from jammer classification, and the state-of-the-art machine learning methods that we use. Chapter 3, Machine Learning Approach for Jammer Detection, describes in detail the data analysis, preprocessing techniques, machine learning methods, and training methodology we have used. The results are presented in Chapter 4, Results and evaluation, along with a deeper dive into four of the best performing models. Finally, Chapter 5, Conclusion, sums up the approach and results we have outlined in this thesis. We also give recommendations for how the models presented in this paper can be put into practical use, and some future work to increase the robustness of the models.

Chapter 2

Theoretical background and State-of-the-art

This chapter aims to provide the reader with background knowledge relevant to this thesis and elaborate on related works.Section 2.1, Theoretical Background, contains definitions and explanations from the domains of machine learning and GNSS services. Section 2.2, The state of the art, presents the respective literature from multivariate time series classification and image classification as well as articles applying machine learning to signal interference.

2.1 Theoretical Background

For the readers unfamiliar with the domain of GNSS or the current practices of jammer detection in Nkom, we have included Section 2.1.1, GNSS Background Knowledge. Furthermore, a target audience for this thesis is from the domain of GNSS management, and they may not be familiar with the concepts of machine learning. We therefore provide some introductory explanations in Section 2.1.2, Machine Learning Background Knowledge.

2.1.1 GNSS Background Knowledge

Concerning the management of RFI, we will provide a brief overview of the file format Nkom uses for recorded events, a paragraph on GNSS jammers, and a description of how expert personnel currently monitor and annotate interference in the frequency bands.

Standard Data Exchange Format for Frequency Band Registrations, CEF Files

The raw data we have used in this project are recordings of RFI events stored in the file format defined by *RECOMMENDATION ITU-R SM.1809, Standard data exchange format for frequency band registrations* [65]. We refer to this format as the *common exchange format (CEF)*. The format standardizes how frequency band registrations are stored so agencies could more easily share data between monitoring campaigns.

[65]

The CEF file format starts with header fields containing metadata of the events and con-

figurations of the monitoring equipment used to record the event. The format allows many possible header fields and a list of which is shown in 2.1. We have marked the fields most relevant to this thesis in bold type. The files store data points after the header fields and a separating blank line. The data points represent monitored signal levels across a bandwidth, and we refer to a row of these points as a "trace". The rows begin with a timestamp indicating when the trace scan was started, followed by comma-separated data point values. An example file and further explanations is given in Section 3.1.1, The Raw Data Format.

GNSS Jammers

Figure 2.1 depicts GNSS jammers similar to those typically confiscated from apprehended perpetrators. The jammers are small devices powered by a battery or the car's electrical system. They can, among other uses, be used to hinder the tracking of a vehicle and circumvent GPS toll systems. [56] found that jammers used in Norway are most frequently found in trucks with company logos. Most likely, this is due to people using company cars for private use or illicit work.

Figure 2.1: Picture of typical small GNSS jammers [6].

Monitoring stations

Monitoring stations are sites with RF monitoring equipment continuously observing certain frequency bands. The respective monitoring agency uses the stations to detect and record RFI events.

The equipment at the monitoring stations scans through the configured bandwidth one segment at a time, recording values for several data points simultaneously. Once the scan has gathered data segments for the entire bandwidth, one trace is complete, and the equipment begins again at the start frequency (of the BW). The size of the segments, filters, sampling techniques, and scan speeds vary between instrument manufacturers.

Recording of an RFI event is triggered when monitored EM radiation levels exceed a threshold limit for longer than a set number of traces (commonly 5).

Current Approach to Jammer Detection without ML

After a monitoring station has recorded an RFI event, the event must be inspected and labeled according to what caused the interference. Currently, experts at Nkom are respon-

sible for this processing, and we will briefly elaborate on how they undertake this manual inspection.

Figure 2.2 illustrates some of the views the experts can use to analyze the data and are all produced from the data points in the CEF file. The top image shows the average data point value per trace over the time duration of an event. The image below shows max- mean- and min-hold graphs. In this view, the x axis represents data point columns (which in turn represents frequencies) and calculates the respective operation (min, mean or max) based on each column of values. As there is essentially no way to check if a label is correct other than tracking down the source, i.e., apprehending a person using a jammer, it is difficult to be 100% certain. Furthermore, spectrograms of certain types of jammer-caused and non-jammer caused events can be quite similar. As a result, some noise in the labels is expected.

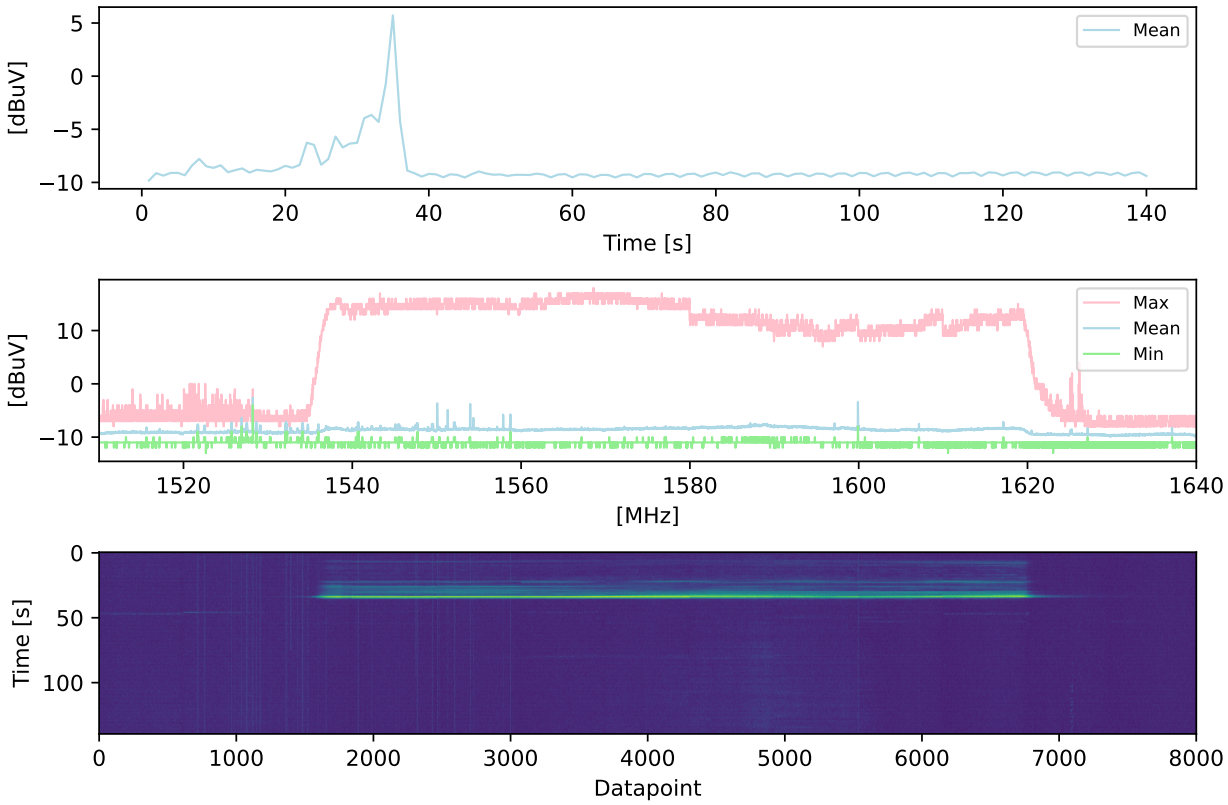


Figure 2.2: Manual inspection views for analyzing an RFI-event.

2.1.2 Machine Learning Background Knowledge

This subsection aims to briefly introduce some general aspects of machine learning methodology, with a particular focus on the training and design of Artificial Neural Networks (ANN). Some important ANN operations, such as Convolution and Attention, are described in greater detail.

Machine Learning Versus Optimization

Machine learning differs from optimization more generally in some key aspects. For one, the model's performance is measured with a *loss function*. This loss function produces a training signal for the model, where the negative gradient of the loss is often used to iteratively update the model towards a lower loss.

Because the loss function must be differentiable to produce gradients, some common performance metrics do not qualify as loss functions. In the case of classification, the common metrics of accuracy and F1-score can be replaced with the differentiable *cross-entropy* loss function.

Unlike optimization, machine learning optimizes the loss function on a *training dataset*, with the aim of generalizing to a distinct *test dataset*. Achieving a low loss from the loss function shows that the model is able to *fit* the data, but its ultimate performance is measured on the test dataset.

In addition to the training and test datasets, we may employ a validation dataset as a proxy for the test set during training. This dataset can for example be used to stop training upon convergence or to change parameters of the network that are not affected by the gradient of the loss function. These parameters are usually parameters of the model or the training procedure and are referred to as *hyperparameters*.

Learning rate

The learning rate is the most important hyperparameter in training ANNs [64]. It determines the step size of the optimizer in the direction of the gradient. Setting the learning rate too high may cause the training to diverge, and setting it too low may cause the model not to converge or to reach a sub-optimal state.

A *learning rate schedule* successively decreases the learning rate over training iterations, allowing the model to converge to some local minima.

Optimizers

In training ANNs, we rely on *backpropagation* to find an update rule to the network weights. Backpropagation computes the gradient of all network weights with regard to the negative gradient of the loss function. On updating the weights in this manner, the loss goes down.

An *optimizer* decides the update rule for the network weights. Trivially updating along the gradient is called *gradient descent*, while *Stochastic Gradient Descent (SGD)* uses mini-batches sampled from the training dataset to compute the loss and gradient. The popular *Adam optimizer* keeps a memory of previous gradients and their second moments to speed up the learning rate along each individual parameter[33].

Metric

The proper evaluation metric for model performance is not the loss function. A performance metric is primarily evaluated on the test dataset, and not during training.

In this thesis, we use metrics for binary (two-class) classification, where the classes are *Jam* and *NoJam*. In binary classficiation tasks, we group the classification results in four categories: *True Positive* samples are correctly classified as Jam (the "positive" class); *True Negative* sample are correctly classified as NoJam (the "negative" class); *False Positive* are wrongly classified as Jam (i.e. the label was NoJam); and *False Negative* are wrongly classified as NoJam (i.e. the label was Jam).

We rely on the following evaluation metrics, bound between 0 and 100, where a higher score is always better:

- Accuracy: Percentage of correct predictions for all classes.

$$\text{Accuracy} = 100 \times \frac{\text{True Positive} + \text{True Negative}}{\text{Total number of predictions}} \tag{2.1}$$

- Precision: Percentage accuracy when the model predicts Jam.

$$\text{Precision} = 100 \times \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.2}$$

- Recall: Percentage of correctly classified Jam samples.

$$\text{Recall} = 100 \times \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}. \tag{2.3}$$

- F1-Score: Weighted average between Precision and Recall.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} * \text{Recall}}{\text{Precision}+\text{Recall}}. \tag{2.4}$$

Data Augmentation

Large ANN models typically require large amounts of data to train successfully. Collecting more data is often resource demanding, and various techniques exist to produce training data from the existing training dataset.

Data augmentation [63] is a particularly popular approach to producing data in the field of computer vision. Examples of data augmentation techniques for image data includes rotation, cropping, and mirroring of images.

To the best of our knowledge, data augmentation has never been used in the field of jammer detection. Many researchers use simulated data and thus produce ample training data that way.

Artificial Neural Networks

An Artificial Neural Network (ANN) is a machine learning model with sequences of linear transformations followed by some non-linear activation. The linear transformations are typically called *layers* of the network.

It has been shown in the universal approximation theorem [39], that ANNs with an arbitrarily large layer, or alternatively with arbitrarily many layers, can model any function between two Euclidean spaces.

Convolutional Neural Networks

The advent of large Convolutional Neural Networks (CNNs) [36] has sparked a revolution in the field of computer vision. CNNs are a category of ANN which uses an operation of

sliding "kernels" across the input data or some layer of the network. Using kernels allows the models to extract features of the data that are spatially independent. Importantly, using kernels greatly reduces the number of weights as compared with fully connected layers.

Current state-of-the-art ANNs in computer vision use various operations in addition to CNNs. These operations either improve the performance, facilitate training, or reduce the size of the networks. For example, residual connections allow some information to skip past some layers in the network, which allows the gradient to pass quicker to deep layers of the network during the backward pass.

Attention mechanisms

Attention Mechanisms are used in ANNs to facilitate influence between all tokens of an input and output sequence [29], independently of the positioning of each token [68]. It can be thought of as a learned token-to-token relevancy. 2.3 shows the attention scores produced while translating a sequence of English words to French and illustrates how each word token is allowed to attend to the entire input and focus on the most relevant parts. There exists a multitude of attention mechanisms, such as Additive attention [3], Dot-Product Attention [42], and the more recent Sliding Window Attention [8]. Positional embeddings, an embedding of information regarding token positioning in the sequence, are often added to the tokens prior to the attention module because the attention computation in itself is not aware of token positions. [68] [73]

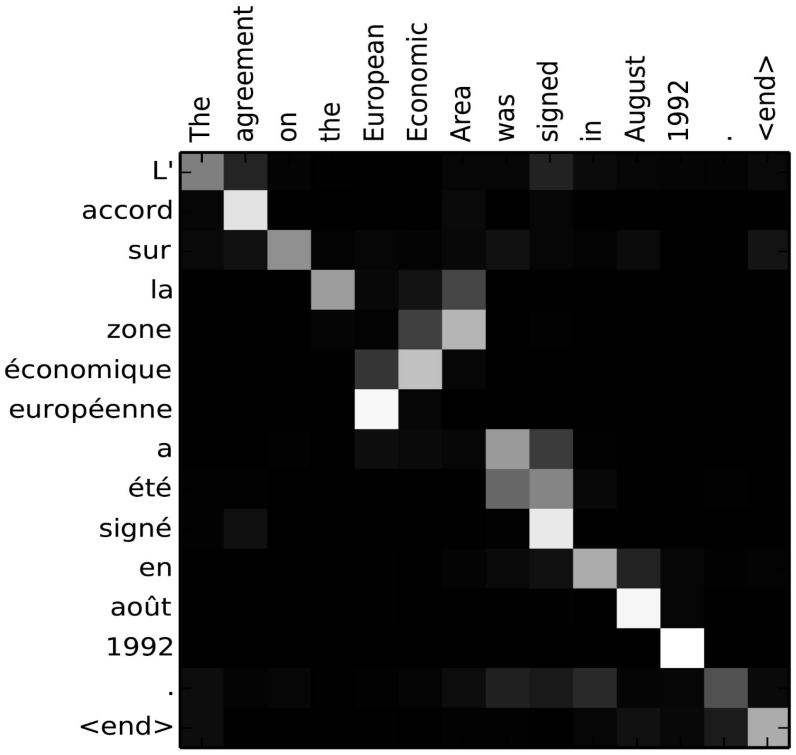


Figure 2.3: Attention scores when translating a English sentence to French [3].

Tranformers and Self-attention

Transformers were introduced in 2017, [68], and used a novel architecture where attention modules were the core units. Previously, attention mechanisms had only been used in con-

junction with other architectures [68] such as RNNs [4] or CNNs [14]. A key concept in the Transformer is the use of self-attention, where the attention mechanism attends to the input sequence, determining a new representation of each token based on the context of the whole sequence [68]. Another key concept is to use multi-headed attention, which replaces each attention operation with multiple parallel attentions over the same input. The attention computation is not aware of the token position [68].

Transformers now dominate the SOTA in NLP [73] [18], and have also been successfully applied to other ML domains [73].

The original Transformer is a sequence-to-sequence Encoder-Decoder architecture. The decoder is used to generate output, and the encoder is used to generate representations. With a self-supervised reconstruction task, the encoders can be used without decoders. The representation generated by (pre)trained encoders can be used as the basis of many tasks, such as classification [73] [18].

2.2 The state of the art

We will now elaborate upon state-of-the-art in fields of interest. Section 2.2.1, Multivariate Time Series Classification Algorithms, describes recent developments and the contest between ANN-based and other models. Section 2.2.2, Image Classification Algorithms, provides an overview over recent CNNs. Lastly, Section 2.2.3, Machine Learning Applied to Signal Interference, gives some examples over articles applying machine learning to signal interference.

2.2.1 Multivariate Time Series Classification Algorithms

A recent review of the state-of-the-art in multivariate time series classification from October 2020 [19] found that in contrast to other machine learning tasks, the dominance of deep learning is not established in this particular task. It found that the ROCKET [17] approach was overall most accurate, followed by two other non-deep learning classifiers, namely WEASEL-MUSE [62] and MrSEQL [37]. However, the introduction of the TS-Transformer [73] could have changed this as it outperforms on all benchmarked datasets, except on two by ROCKET and one by XGBoost [15].

Time Series Transformer

Time series transformers (TS-Transformers) [73] have shown promising results on multiple benchmark datasets of multivariate time-series data. The model architecture is inspired by the transformer architecture of [68] and [18]. TS-Transformers are significantly different from previous SOTA models working with multivariate time-series, as they are the first model to work on a wide variety of tasks. Characteristic of ANN-based models, it does not rely on heavy data manipulation or arcane domain knowledge to achieve good results. On introducing the TS-Transformer, [73] states that it is the first SOTA ANN model for multivariate time-series data. More details on this model and elaboration on how we reduced its complexity by a novel configuration are found in Section 3.3.3.

ROCKET

ROCKET[17] (RandOm Convolutional KErnel Transform) is a transformation technique used to transform time series data in preparation for a linear classifier. It transforms data by using a large number of randomly generated and kernels similar to those found in CNNs. Though in contrast to CNNs, ROCKET employs a massive variation in kernel types, especially varied in their dilation.

For each kernel, two values are computed, the global max and proportion of positive values (PPV). The authors state that the PPV values are the single most critical element for the approach's effectiveness because it identifies the relevance of a pattern in the transformed data. Transforms can be run in parallel, allowing scalability to large datasets.

The main strength of ROCKET is its extremely fast training speed. As stated in [17] 'Using this method, it is possible to train and test a classifier on all 85 'bake off' datasets in the UCR archive in <2h, and it is possible to train a classifier on a large dataset of more than one million time series in approximately 1 h.'

The method was first developed for univariate time series, but the authors have since developed a version for multivariate data as well. A recent comparison of multivariate time series classifiers [58] found that it is effective across a range of datasets and even proposed it as a baseline for future papers.

2.2.2 Image Classification Algorithms

Since the introduction of AlexNet in 2012 [36], CNNs have been the go-to machine learning approach to image tasks [21]. Resnets were introduced in 2016 [26], wich facilitate training of deeper networks by introducing residual connectios (also know as skip connections), and the state-of-the-art from that point on have been permeated with similar CNNs [43] [71] [34]. One exmaple is the EfficientNet family of CNNs, which increased efficiency by a new scaling coefficient in width, depth and resolution. Resolution here referes to the final part of the network consisting of densly connected feed forward layers [67].

A recent introduction, and possible contender, is the Vision Transformer [21]. It offers comparable accuracy with other SOTA CNNs, with a significant reduction in computation time and parameters.

2.2.3 Machine Learning Applied to Signal Interference

We reviewed around 35 papers pertaining to the combination of machine learning and signal interference but we found no articles with exactly the same frequency and resolution data. The most similar paper we found is elaborated upon in some detail Section 2.2.3 Jammer Classification in GNSS Bands With Machine Learning Algorithms. Other than this we have included a variety of interesting papers using machine learning in relation to siglal interference.

Jammer Classification in GNSS Bands With Machine Learning Algorithms

The paper most similar to out work in terms of application domain and data is [45], though the temporal duration of their signals are significantly lower, and the sampling rate significantly higher.

The authors argue that the classification of jammers is an important part of effective GNSS interference management solutions and that the literature is lacking in that regard. To remedy this, they created a dataset by sampling GNSS data from frequency band GPS L1 (at an I/Q rate of 20 MS/s [55]), and made mathematical models to simulate jammer signals. The simulated signal where added to the sampled GPS data to create the dataset samples. Five classes pertaining to common types of jammers [10] as well as a class for interference free samples was included. A sixth main type of jammer, namely Wide Band jammers, was not included as they are particularly hard to distinguish.

The dataset and code [46] was made publically available and contains 61,800 gray-scale 512 $\times$ 512 pixel images. The images are distributed into a 6000 image training set, 1800 image validation set, and a 54,000 image test set. All sets contained an even distribution among classes.

They trained a relatively simple CNN architecture, and trained for 25 epochs using the Adam optimizer. The layers of the ANN were: an input layer; a convolutional layer with 16 filters of 12 x 12 x 1; ReLU activation function; Max pooling 2 x 2, stride 1; a fully connected layer; Softmax activation function; Classification based on softmax output.

For their SVM model, features were extracted using the Bag of Features [49] method, which extracts local features from the images that disregard image location and orientation, and builds a vocabulary by clustering the features. The authors used K-means clustering and a 500 word visual vocabulary. The SVM model utilized a RBF kernel and was optimized using the Sequential Minimal Optimization method [51].

They achieved a mean accuracy of 94.90% with the SVM and 91.36% with the CNN, and almost a 99% accuracy in binary classification of presence or absence of jammers. The CNN achieved a 0.4% higher accuracy in the binary classification, and the authors speculate that the lower mean accuracy of the CNN compared to the SVM was due to the number of parameters of the CNN resulting in a more difficult to train model. Raw GPS data without simulated jammers is available here [55].

An Efficient Way for Satellite Interference Signal Recognition Via Incremental Learning

In [72] an incremental learning SVM, originally developed in 2003 by [20] is applied to recognition of satellite interference signals. The authors extend the binary SVM into a multiclass classifier in three ways and compare it to the LIB-SVM multiclass SVM [13]. Their methods achieve comparable results with reduced memory and processing requirements.

Recognition Method of Dense False Targets Jamming Based on Time-frequency Atomic Decomposition

instead of using traditional time-frequency analysis methods such as short-time Fourier transform or Wigner distribution, the authors of [25] use Gabor time-frequency atomic decomposition to extract feature vectors for a SVM classifier. The paper's interest lies in recognizing

the presence of dense false target jamming in the domain of radar detection.

The authors build a discrete dictionary of Gabor atoms (which are modulated Gauss functions), decompose the signal by sparse decomposition, and iteratively match and subtract Gabor atoms from the signal until the signal power is under a certain threshold or another stopping criterion is met. The extracted Gabor atomic time-frequency parameters are used as the feature vector for the SVM classifier. The specifications on the SVM classifier, such as which kernel function was used, were not disclosed.

Compound Jamming Signal Recognition Based on Neural Networks

This article [59] extracts 10 features from the time and frequency domain as well as from fractal dimensions and uses a simple ANN of 10 input neurons, 15 hidden neurons, and 12 output neurons, all with sigmoid activation functions, to classify various forms of compound jamming. Compound jamming is the presence of more than two jammer signals simultaneously. The papers interests lies in military radar applications and electronic warfare.

Support Vector Machines for Classification of Automotive Radar Interference

The authors of [74] apply machine learning in the context of radar-to-radar interference in automotive radars in advanced driver assistance systems. The studied frequencies are around 77 GHz, which are commonly used in automotive radars such as the mmWave automotive radar [74]. The 77 GHz signals are received and run through on board processing by use of a mixer, reference chirp, amplification and a low-pass filter. Lastly, the signals are digitally converted to by an analog-to-digital converter (ADC).

The time-frequency data is then transformed by Stretch-processing and Pulse-Doppler Processing. A section of the Range-Doppler response becomes the high-dimensional frequency domain waveform data which is classified into one of seven categories by a large-scale linear SVM based on [28]. The authors choose SVMs because they require relatively little computation, while still providing a high accuracy, and is therefore a candidate solution for real time interference mitigation in autonomous vehicles. They show a 90.6% cross-validation generalization accuracy, with confusion matrix values from 76.0% to 98.9% and argue that misclassification happens in less common interference types, and is of less importance to the task.

Radio Ground-to-air Interference Signals Recognition Based on Support Vector Machine

In [35] the authors use an SVM, optimized by the gravitational search algorithm (GSA) [54], to classify normal and abnormal signals in radio to ground communication in the interest of detecting interference. GSA is a type of heuristic optimization method inspired by Newtown's laws. The search algorithm agents act like objects attracted to each other by gravitational forces, and the fitness function assigns heavier masses to better solutions so that they change more slowly and attract others more heavily. The algorithm finds the solution when agents converge to the heaviest mass, which represents the best solution.

The data was sampled from the aviation frequency bands, with a sampling rate of acoustic frequency 44.1Kb [35]. A set of six features were used; Short-term average energy [41];

Short-term magnitude mean [30]; Short-term magnitude variance [44]; Short term zero cross-rate [32]; Short-term normalized kurtosis sum [5]; and Amplitude spectrum kurtosis index [5]. The data samples were all 15s in duration, and the six feature were extracted from 300 ms sections of the time-frequency data

CEF file header fields			
Fieldname	Data format	Description	Example
File Type	Text	Type and version of the datafile	Common exchange format V2.0
Location Name	Text	Name of location where the measurement was made	NERA
Latitude	Text	Coordinates in format DD.MM.SSx with $x \in \{N, S\}$	52.10.04N
Longitude	Text	Coordinates in format DD.MM.SSx with $x \in \{E, W\}$	005.10.09W
Frequency Start	Numeric (real)	Start frequency in kHz	1000.000
Frequency Stop	Numeric (real)	End frequency in kHz	2000.000
Antenna Type	Text, Numeric (real), Numeric (real)	Info, gain in dBi, Kfactor in dB/m (the last two can be omitted)	LPD, 7, 10
Filter Bandwidth	Numeric (read)	In kHz	0.2
Level Units	Text	One of dBuV, dBuV/m or dBm	dBuV
Date	Text	Start date of measurement	2006-06-25
Data Points	Numeric (integer)	Number of data points in each trace (row of data)	80000
Scan Time	Numeric (real)	Scan duration in seconds for the equipment to scan through the frequency range	24.1
Detector	Text		RMS
Note	Text	General comments	
Antenna Azimuth	Text	DDD.DD (0 = North)	181.12
Antenna Elevation	Text	DD.DD (0 = No elevation)	45.32
Attenuation	Numeric (integer)	Equipment attenuation setting in dB	3
Filter Type	Text	Filtertype bandwidth and shapefactor	Gaussian 3 dB shapefactor 3.2
Displayed Note	Text	Short note	
Multiscan	Text	Y or N, defaults to N	Y
Measurement Accuracy	Numeric	Total accuracy of the system	
Video Filter Type	Text	Video filtertype bandwidth and shapefactor	

Table 2.1: Common Exchange Format header fields.

Chapter 3

Machine Learning Approach for Jammer Detection

This chapter will elaborate the details of the approach we used to achieve results described in Chapter 4. First, Section 3.1 Data Analysis describes how we analyzed the raw data and provides the basis of how we choose to preprocess the data samples as explained in Section 3.2 Data Preprocessing. Lastly, Section 3.3 AI Based Algorithms for Jammer Detection explains the technical details of the machine learning classifiers used in this thesis.

Over the duration of this thesis, we made two main iterations of the training and validation datasets. We made the first iteration from data available in February, 2021 and the second iteration from data available in April in the same year. Though both iterations were used to train and validate classifiers, we will explain the process as it pertains to the second iteration. In addition, we made some adjustments to the approach based on preliminary results obtained with the first iteration.

3.1 Data Analysis

In this section, we will analyse the raw data in order to obtain a good comprehension of the data itself. In more details, Section 3.1.1 The Raw Data Format provides a brief explanation of the raw data files used to generate the data samples. Next, Section 3.1.2 Spectrogram Visualization illustrates how we visually inspected the data points. Finally, in Section 3.1.4 Sample Distributions we elaborate on the analysis we used to decide how the samples should be standardized.

3.1.1 The Raw Data Format

The raw data used in this thesis represents RFI-events stored in CEF files. Section 2.1.1 gives a lengthier description of the format. In this subsection, we will describe the features most relevant to this chapter. The header field *LocationName* tells us which monitoring station captured the RFI event. More specifically, *FreqStart* and *FreqStop* define the frequency band captured in the measurement. We refer to this range as the bandwidth (BW) of the event. *DataPoints* denotes how many discrete points are recorded to represent the continuous frequency band defined by the BW. We refer to a row of data points as a *trace*. The Resolution Bandwidth (RBW) is the segment size of the total BW of each data point.

LevelUnits indicates the unit used with the data points to record amplitudes. Listing 3.1 illustrates a CEF file example, with added comments inside brackets.

Listing 3.1: CEF file example with comments inside brackets.

```
FileType Standard Data exchange Format 2.0  [Name of file type]
LocationName xxx            [Location of measurement]
Latitude 11.22.33N
Longitude 44.55.66E
FreqStart 1560000           [Start frequency in kHz]
FreqStop 1610000            [Stop frequency in kHz]
AntennaType someAntenna     [Type of antenna connected to instrument]
FilterBandwidth 5000        [Distance between measurements in kHz]
LevelUnits dBuV             [Measurement unit]
Date 2020-29-9              [Date of when data was collected]
DataPoints 10               [Data points in each trace]
ScanTime 0.018              [Time to measure one detector window]
Detector FFM                [Detector type for measurements]
Note Example                [Custom notes can be added]

[Time]   [data point 1   ....   data point n]
17:08:10,-11,-11,-7,-9,-13,-11,-9,-11,-8,-8 [Trace 1]
17:08:11,-11,-10,-7,-9,-13,-14,-9,-14,-8,-9 [Trace n-3]
17:08:12,-11,-12,-7,-4,-14,-12,-8,-11,-8,-6 [Trace n-2]
17:08:13,-15,-12,-7,-9,-13,-11,-6,-11,-8,-4 [Trace n-1]
17:08:14,-11,-12,-7,-2,-13,-11,-7,-11,-8,-8 [Trace n]
```

3.1.2 Spectrogram Visualization

We visually inspected the RFI events by generating spectrogram images from the CEF files. The spectrograms are heatmaps of the amplitude values across frequencies and time. Data point numbers (frequencies) are along the x-axis and traces (time) on the y axis.

Figure 3.1 shows three typical spectrograms generated from JAM labeled CEF files, and the spectrograms in Figure 3.2 are from CEF files labeled with NoJAM. The top spectrogram in Figure 3.1 only shows the first 160 traces of the sample and illustrates that a clear jammer pattern can be observed already within trace 50.

From the spectrograms, we can also observe the 30-second buffer preceding the triggering amplitudes in the first few dozen traces. This buffer might contain a certain ramp-up of the amplitudes but did not reach the threshold to start the recording of an event.

3.1.3 Data Cleaning

We found that a portion of the CEF files contained traces without data points and removed these empty traces.

Next, we applied the 1s MaxHold transformation described in Section 3.2.1 which reduced all files to 1 trace per second. Prior to this reduction, they ranged from 1 to 10 traces per second depending on the speed and settings of monitoring equipment.

Lastly, for some we set the maximum number of traces per CEF file to 250, keeping the earlist ones. We allowed these early modifications based on the assumption that the CEF files

would still contain ample information to classify the RFI event and provided the practicality of significantly reducing the total storage requirements of the data.

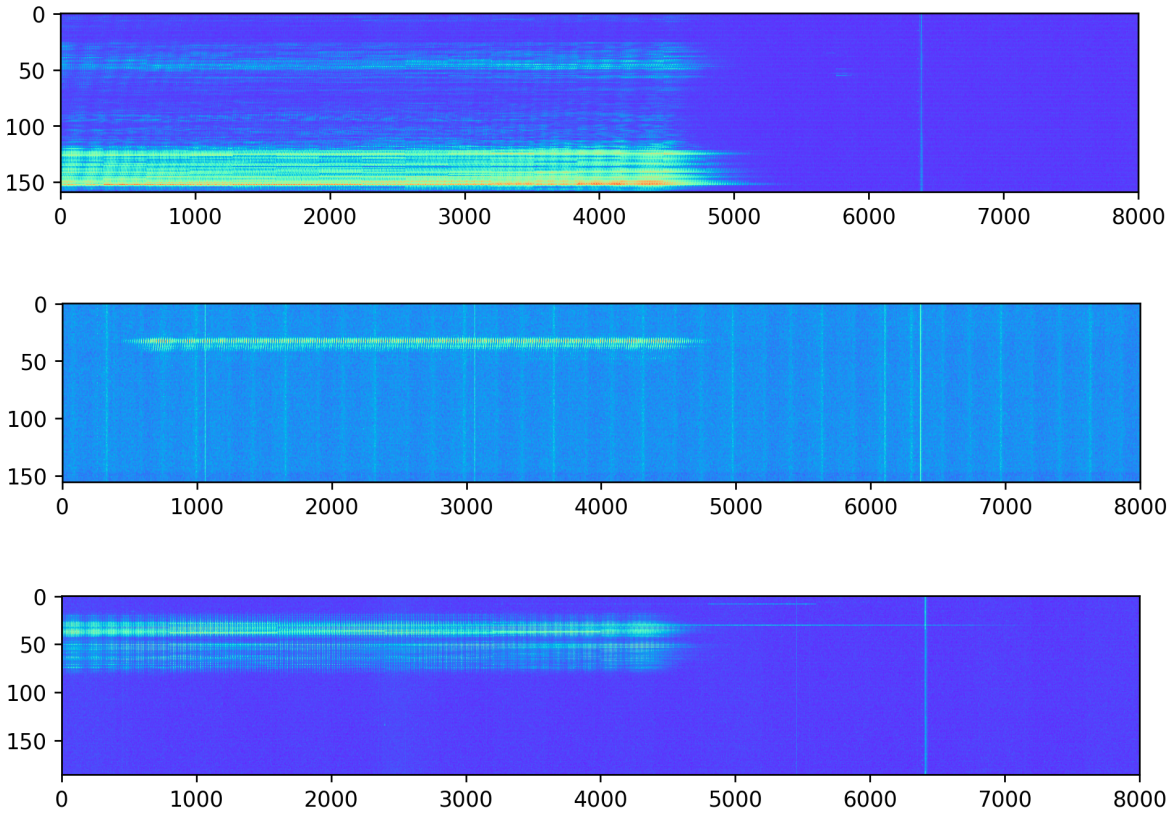


Figure 3.1: Spectogram images of typical JAM-events.

All measurements use dBuV as the measurement unit, eliminating the requirement to translate the data points to a common amplitude scale.

3.1.4 Sample Distributions

Label distribution

CEF files believed to correspond to RFI events caused by jammers are labeled *JAM*. Otherwise, they are labeled *NoJAM*. The distribution of the two labels is unbalanced with a ratio of around 1:7, with JAM in the minority as shown in Figure 3.3. The prevalence of NoJAM data is due to the monitoring stations being extra sensitive to ensure all potential JAM events are detected. This configuration results in many different interference events triggering the detection mechanisms, and theese events are labeled NoJAM. For example old cars or faulty radio transmitters can trigger the mechanism. NoJAM events are generally not disruptive to GNSS services, unlike jammers.

Event Duration

Figure 3.4 shows a histogram of the number of samples with a given duration. Duration is given by the number of traces in each sample after the data cleansing mentioned above.

The high concentration of samples, around 150 traces, is due to configuration settings most

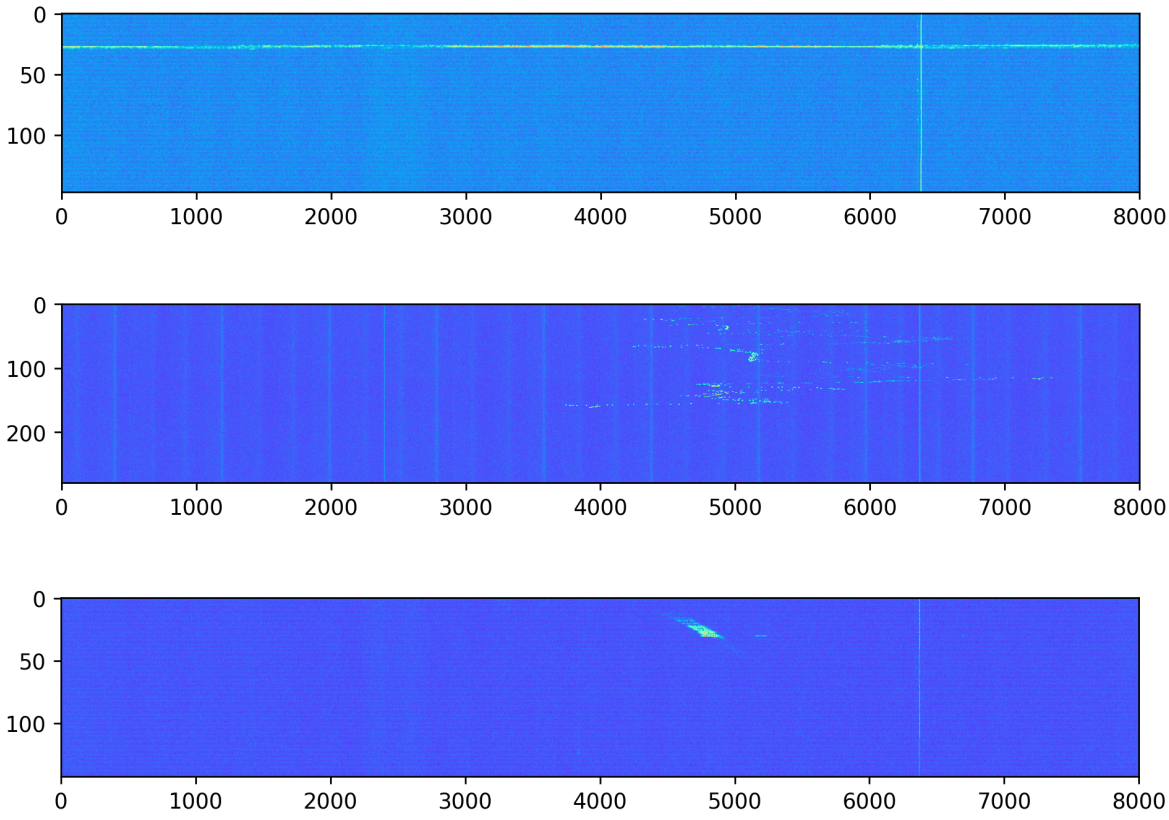


Figure 3.2: Spectogram images of typical NoJAM-events.

common to the monitoring stations capturing the events used. This setting keeps a buffer of 30 seconds so that when amplitudes exceeding the set threshold limit are found, this buffer is prepended to the CEF file as well as the 120 seconds after the amplitudes show below the threshold. The samples in Figure 3.4 to the left of the peak are samples from which we removed corrupted traces, and were left with fewer than 150 traces.

Bandwidth and Resolution Bandwidth Distribution

Figure 3.5 shows an overview of the BW ranges captured in the CEF files. The ranges are configured to monitor interference that affects GNSS systems, particularly interference in the GPS L1 band. Therefore, a strong concentration of CEF files with BW 1560-1610 MHz and only small deviations from this range is seen. This is beneficial because this means the raw data holds comparable frequency information. Furthermore, that allows us to transform the data from this narrow set of frequencies into common features for the classifiers.

The number of data points in each sample varies according to the number of traces and the number of data points in each trace. In our dataset, the number of data points in each trace ranges from 8001 to 24,000 per trace, with RBW from 2.5 kHz to 6.25 kHz.

Source Distribution

The RFI events we have used in this thesis originate from six different RFI interference monitoring stations at different locations. These are referred to as PAT, ACM, BGO, GEN, KHA, and ROS. Figure 3.6 shows the distribution of samples according to their source. The events that originate from ACM are from controlled field tests of actual jammers.

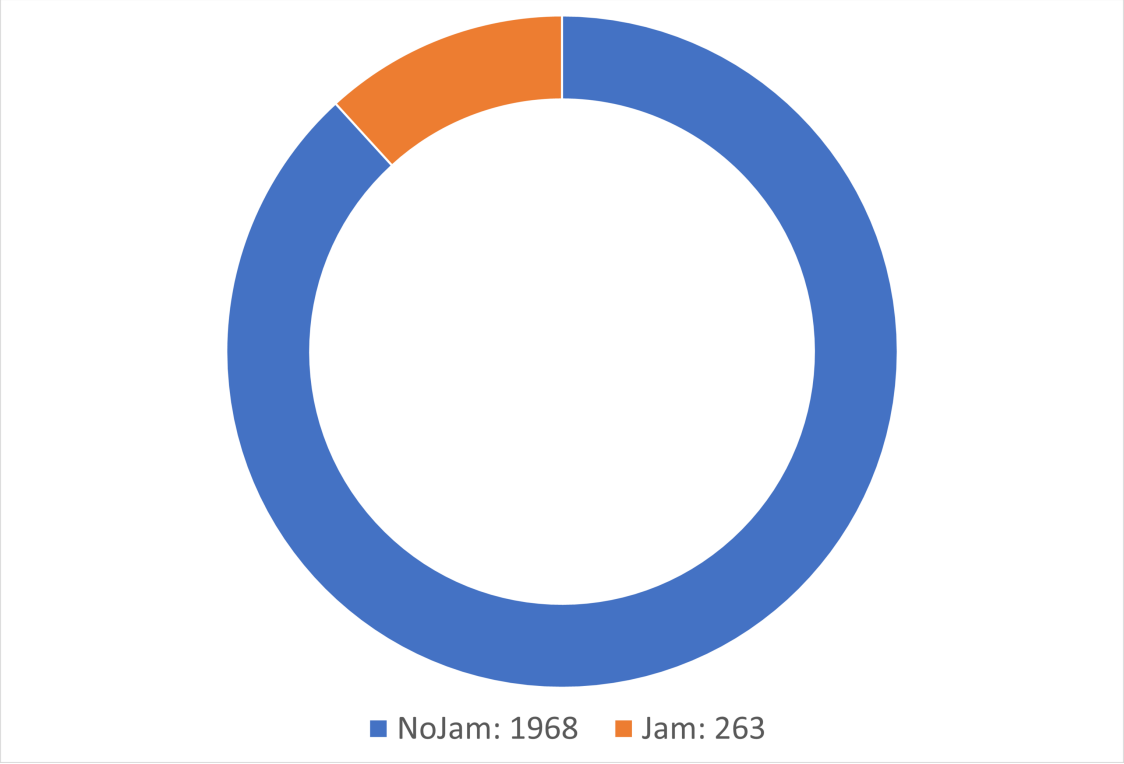


Figure 3.3: Distribution of JAM and NoJAM labels in the dataset.

Informed by the analysis stated in this section, we implemented the preprocessing described in the following section. In early discussions, we speculated in padding schemes that could compensate for differences in bandwidth by adding columns according to missing frequency ranges. However, we learned that the raw data holds relatively similar frequency information, allowing us to attempt a simpler preprocessing for the standardization of data points. Furthermore, we discovered key information regarding event duration and what the data rows were most likely to hold critical information, which dictated how we should standardize the number of traces.

3.2 Data Preprocessing

This section will describe how we prepared the collection of CEF files for our machine-learning based classifiers. Section 3.2.1 CEF File Transformations describes the transformations we applied to turn each CEF file into data samples. Next, we see how early model results uncovered mislabeled data in Section 3.2.2 Correcting Labels. Section 3.2.3 Dataset creation details the distribution of these datasamples into training, validation and testings sets, and finally Section 3.2.4 Datasets lists some statistics of the resulting dataset.

3.2.1 CEF File Transformations

In order to transform the CEF files into data samples that the classifiers can digest, we needed to find a common standardization procedure for all data.

The transformations we describe here are based on the analysis in the previous section. The most important procedure was the standardization of data points and traces. Several schemes of utilizing the header fields of the CEF files were considered, such as incorporating

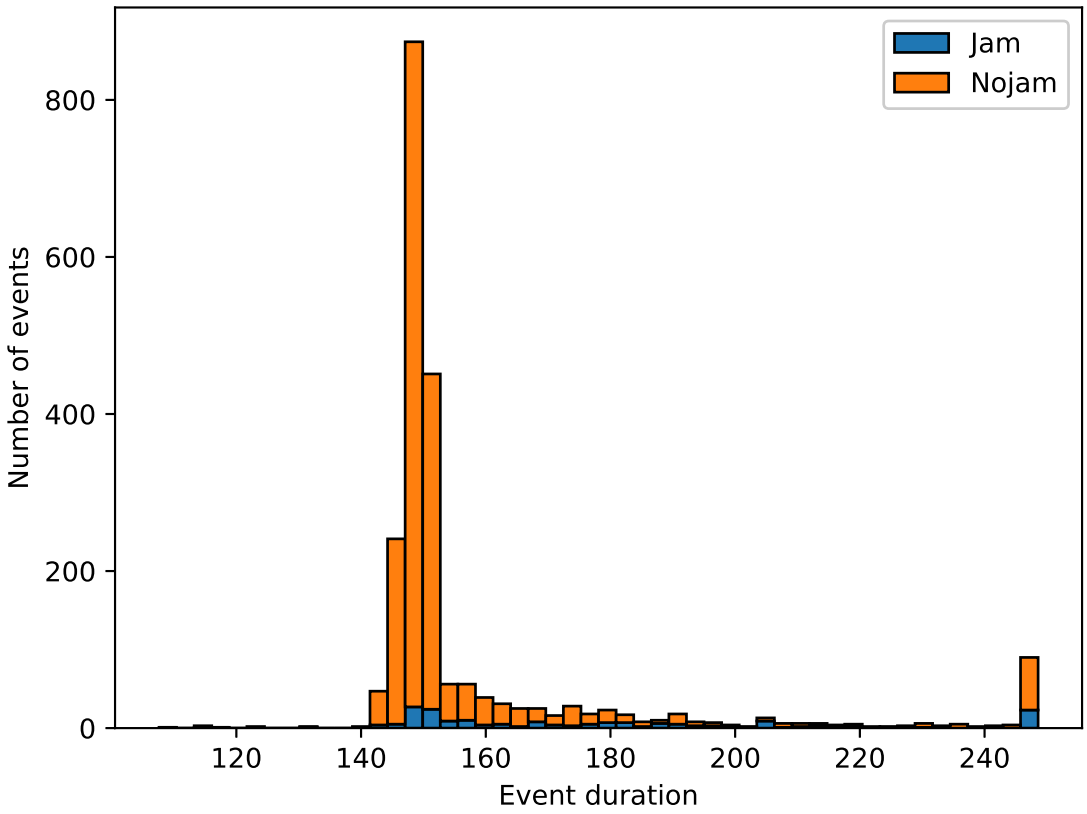


Figure 3.4: Historgram of the number of signals with a given duration.

location information to mitigate a potential problem caused by variation in background noise based on source. However, we decided to first focus on the data points, and disregard the header fields as input to the classifiers, and observed that this was sufficient to obtain the results reported in this thesis.

The data is not normalized in any way during preprocessing. This is firstly because all data points are already recorded on on the same scale. Secondly, minimizing the computational requirements during preprocessing is beneficial in the scenario of deploying the model to the monitoring stations with their limited resources.

1s MaxHold

We applied a 1-second MaxHold transformation to CEF files containing more than one trace per second. The MaxHold operation selects the highest valued amplitude from each frequency across all traces with the same time stamp. This way, traces with the same second-precise time stamp are reduced to one trace. Listing 3.2 shows an example of how this transformation converts two traces with the same timestamp into one new trace.

Cropping and Padding of Event Duration

From our knowledge of the monitoring configurations, we concluded that cropping the files to 140 traces would be a practical and viable way to standardize the vast majority of the

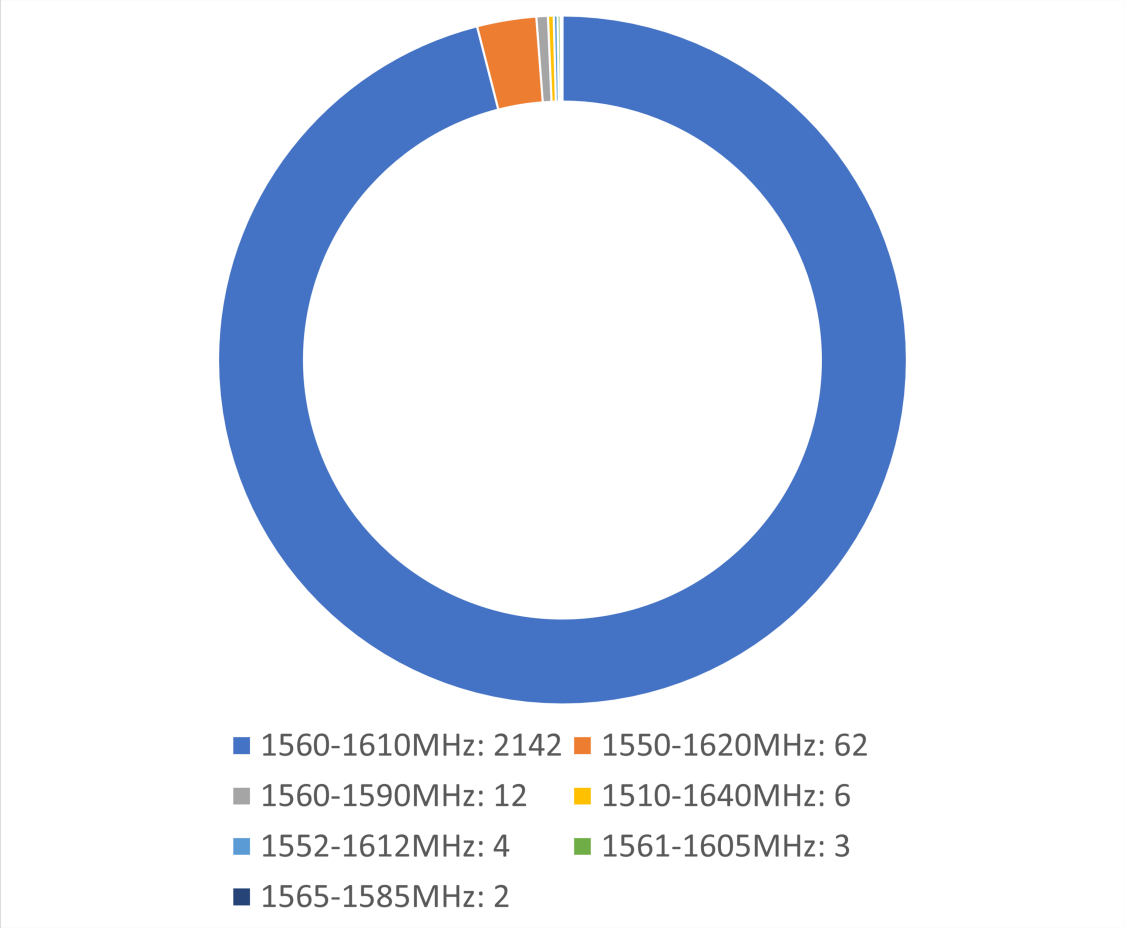


Figure 3.5: Event Bandwidth Distribution.

samples. This approach is also motivated by inspecting the RFI event spectograms, and the number of traces per sample in the raw dataset.

Listing 3.2: Two traces for time 17:08:10, become one trace by 1s MaxHold.

```
17:08:10,-11,-11,-7,-9,-13,-11,-9,-11,-8,-8      [Trace A, timestamp 17:08:10]
17:08:10,-11,-10,-7,-9,-13,-14,-9,-14,-8,-9      [Trace B, timestamp 17:08:10]

17:08:10,-11,-10,-7,-9,-13,-11,-9,-11,-8,-8      [New trace, after
                                                  applying 1s MaxHold]
```

Only ten CEF files contained fewer than 140 traces. These samples were padded by a method we call "reflection padding", where we append the last traces of the file in reverse order, excluding the last trace. Using this padding regime, the same background noise that is found in the last parts of the signal is reused to simulate a realistic ending to the sample. We allowed a maximum of 30% of the traces to be padded, and removed one file based on this criteria. Listing 3.3 shows an example of how this padding scheme would pad a file with 4 missing lines.

Listing 3.3: Illustration of reflection padding of four traces.

```
17:08:10,-11,-11,-7,-9,-13,-11,-9,-11,-8,-8 [Trace 132]
17:08:11,-11,-10,-7,-9,-13,-14,-9,-14,-8,-9 [Trace 133]
17:08:12,-11,-12,-7,-4,-14,-12,-8,-11,-8,-6 [Trace 134]
17:08:13,-15,-12,-7,-9,-13,-11,-6,-11,-8,-4 [Trace 135]
17:08:14,-11,-12,-7,-2,-13,-11,-7,-11,-8,-8 [Trace 136]
[        File ends, padding starts         ]
```

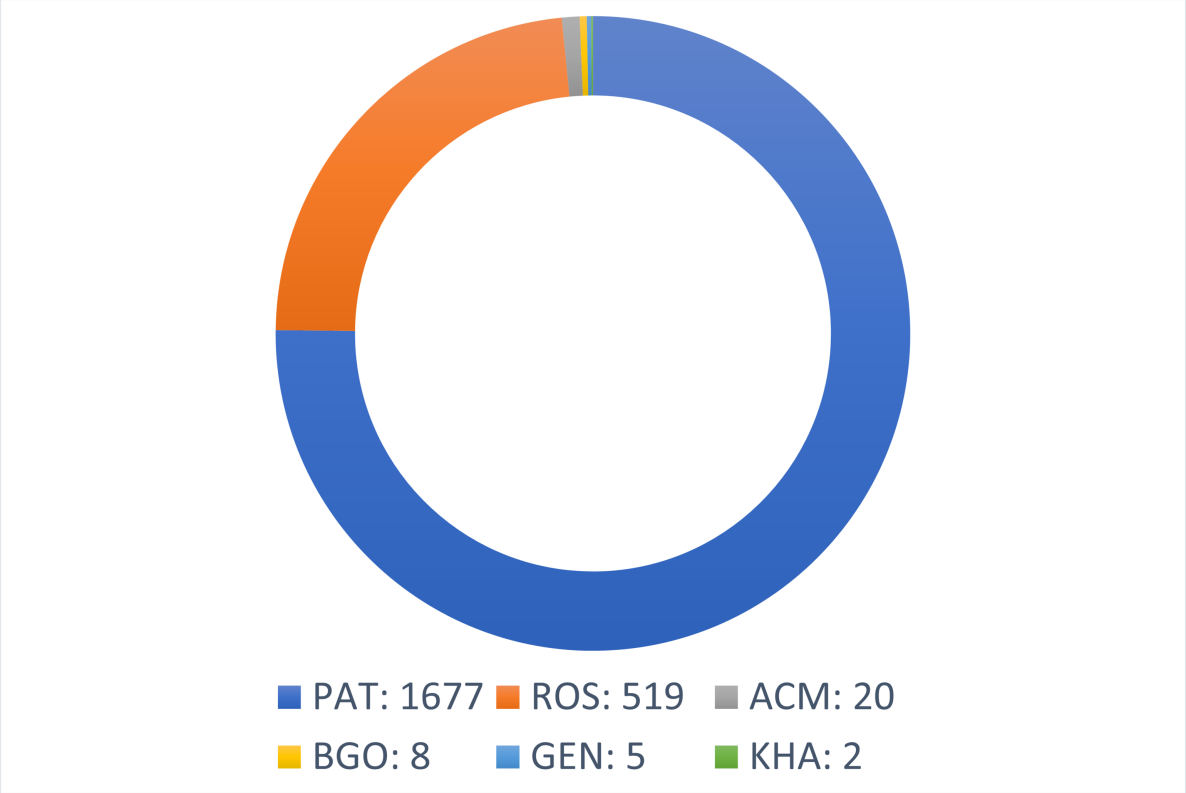


Figure 3.6: Distribution of samples according to source location.

```
17:08:13,-15,-12,-7,-9,-13,-11,-6,-11,-8,-4 [Trace 137, copy of trace 135]
17:08:12,-11,-12,-7,-4,-14,-12,-8,-11,-8,-6 [Trace 138, copy of trace 134]
17:08:11,-11,-10,-7,-9,-13,-14,-9,-14,-8,-9 [Trace 139, copy of trace 133]
17:08:10,-11,-11,-7,-9,-13,-11,-9,-11,-8,-8 [Trace 140, copy of trace 132]
```

Standarize Data Points

To accommodate for a wide variety of potential classficiation models, we aimed to standardize the number of data points in each sample. Using the size of the smallest sample in the dataset, we downsampled all traces in all data samples to 8001 data points.

If we observe that 8001 data points is sufficient for a strong classification performance, the monitoring devices could be configured to only capture a range of 8001 data points. Then, the measurements can be fed directly to the classifier without any preprocessing. The monitoring devices can also be configured to capture only one trace per second, making the 1s MaxHold transformation obsolete.

3.2.2 Correcting Labels

In the preliminary results, we obtained an accuracy of around 98%. We then inspected the samples that were wrongly classified during validation and discovered that some of the samples were mislabeled.

We created a set of samples that are candidates for being mislabeled, and thus should be reinspected. First, we created five versions of the dataset with an 80/20 split, using non-overlapping samples in each of the five validation splits (also called cross-validation). We

then trained our AlexNet, Squuezenet1_1, and Resnet18 models on each of the five versions for 20 epochs, achieving 96% to 99% accuracy. Any sample that was wrongly classified by one of the models was added to candidates for reinspection. Following this approach, 72 samples were marked for reinspection, and ten samples were relabeled by a human expert after an inspection.

3.2.3 Dataset creation

Datasample Views

The samples were presented to the classifiers either as PNG images with a resolution of 500×140 pixels, or as matrices of shape 8001×140. The image width of 500 was selected to approximate the size used in [45]. The height of 140 is equal to the number of traces. Note that we trained models on both three-channel color images and one-channel grayscale images for the image view and found that the color version worked best with our models.

Training, Validation splits

Our final training and validation sets were based on the samples made available in April. We grouped the samples by source monitoring station and label. We group by source to present the model with more varied data. From each group distinguished by source and label, we put 80% in the training set and 20% in the validation set. This is shown in Figure 3.7.

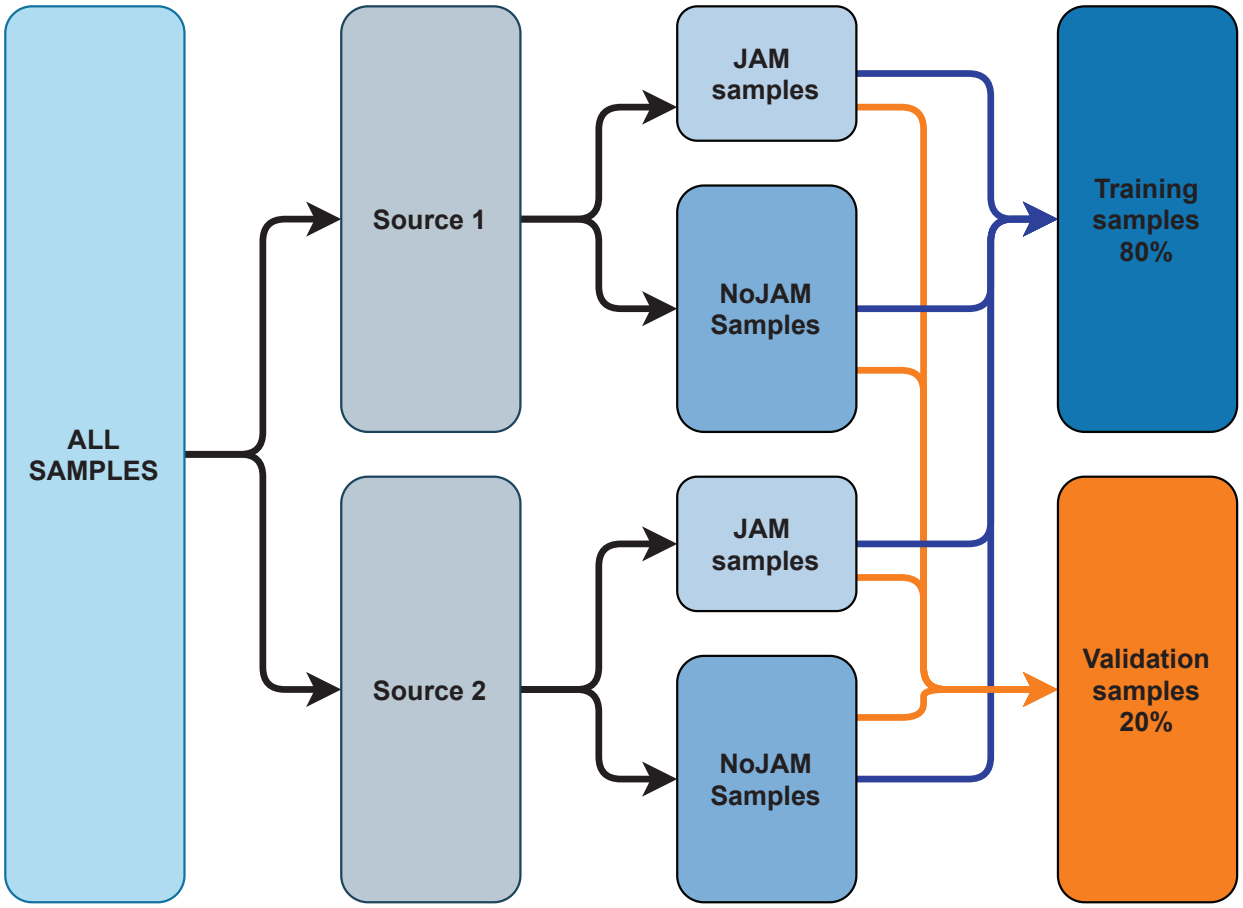


Figure 3.7: Divition of samples based on source and label.

Augmentation and Oversampling on the Training set

The distribution of classes in the data was very unbalanced, with 11% of samples labeled JAM and 89% of samples labeled NoJAM. For that reason, we experimented with data augmentation and oversampling to observe the effects of balancing the dataset.

We investigated augmentation techniques that would produce realistic samples. By this criteria, augmentation by horizontally flipping (or mirroring) the samples was the strongest candidate. As shown in Figure 3.8, this still produces a realistic jammer spectrogram, with the only difference being that the interference is observed at different frequencies.

Other relevant augmentation techniques include vertical flipping, zooming, and shifting, but these would have be more administered carefully if we wanted to maintain the typical characteristics of the samples. As the motivation for augmentation was to balance the ratio between JAM and NoJAM samples, we sought augmentations to apply only to JAM signals. Because the augmentations are only applied to one class, we cannot create artefacts in the data that the classifier can use as shortcuts. For example, by vertically flipping the JAM samples, the classifiers could easily learn to classify samples with the 30 trace buffer at the end as JAM samples. To reduce the risk of introducing such artefacts, we concluded that horizontal flipping was sufficient augmentation for our needs.

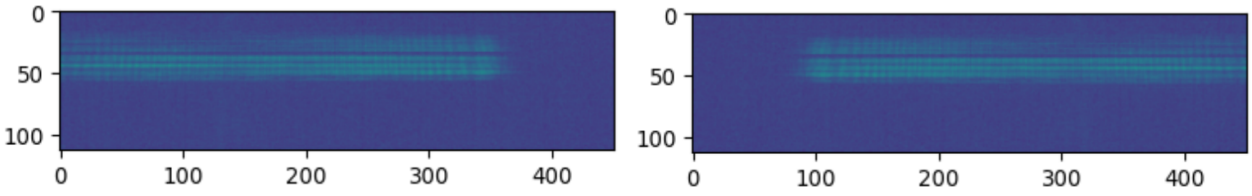


Figure 3.8: JAM samples augmented by horizontal flip. Left: Original, Right: Augmented version.

With oversampling and vertical augmentation as our tools, we created four versions of the training set. The "Basic" dataset has no augmentation or oversampling, the "Flipped" uses horizontal flipping as augmentation on the JAM data. Additionally, "Basic-1to1" is an oversampled version of the unaugmented dataset, and "Flipped-1to1" is an oversampled version of the augmented dataset. The oversampling of JAM samples give a JAM-to-NoJAM ratio of 1:1. Table 3.1 gives an overview of the settings for the four models.

With the PNG image view of these four datasets, we train a Resnet34 based model on each of the datasets. Surprisingly, we saw no benefit from either augmentation or oversampling. However, the "Basic" dataset marginally outperformed the others in terms of accuracy, while also converging faster. Following these results, we used the "Basic" version of the training set in this thesis, with no augmentation or oversampling. We also compared using three-channel color images to one-channel greyscale images for the image view, and found that color images performed best with our models.

	Vertical flipping	Oversampling
Basic	No	No
Flipped	Yes	No
Basic-1to1	No	Yes
Flipped-1to1	Yes	Yes

Table 3.1: Summary of dataset configurations by augmentation and oversampling.

3.2.4 Datasets

The final number of samples in the respective training, validation, and test sets is shown in Table 3.2. The test dataset consists of all data made available to the project after April 2021, when the final training and validation sets were created. The number of JAM to NoJAM samples is even less balanced in the testset as compared to the training and validation sets.

	Training	Validation	Testing
JAM	209	54	43
NoJAM	1571	396	909
Total	1779	450	952

Table 3.2: Training, validation, and testing split summary.

3.3 AI Based Algorithms for Jammer Detection

Here, we will elaborate on the classifiers and configurations we employed in this thesis. Section 3.3.1 Algorithm Overview summarizes the algorithms we implemented. Section 3.3.2 ANN Commonalities and Training Parameters describes the training procedure for all ANN based classifiers. Finally, in Section 3.3.3 TS-Transformer32, we describe the state-of-the-art multivariate time series Transformer classifier (TS-Transformer) found in [73], and how we modified this architecture to reduce the number of parameters by half while maintaining comparable results.

3.3.1 Algorithm Overview

We found in our literature review (see Section 2.2.3) that most machine learning approaches to the jammer detection problem had used relatively simple models, almost exclusively focusing on SVMs and simple CNNs. This motivated us to try classifiers of varying complexity, and we have included CNN models ranging from 1.3 million to 60 million weights in our trials. We also observed no use of the state-of-the-art multivariate time series classifiers, and we test several of these algorithms for jammer detection. After preliminary results, we observed that we could train very accurate models in less than 1 hour (over 99% accuracy with F1 Scores over 0.98) and therefore set 1 hour as the upper limit for allowed training time. This excluded the use of MUSE and MrSEQL models, as they did not complete training within this limitation. We also excluded other multivariate time series classifiers reported to take longer to train than MUSE and MrSEQL, such as HIVE-COTE [38].

We include two approaches that are not based on ANNs. The first, an SVM based classifier, is the best performing model used in [45], which is the paper in the literature with the most similar data samples to ours, to the best of our knowledge. The second is ROCKET [17], a fast-to-train time series classification model that has obtained good results on a wide range of benchmark datasets. ROCKET is suggested as a new benchmark model in [58] and outperforms the commonly used dynamic time warping benchmark in all results we have come across.

Most notable among the ANN models we trained are the three versions of the TS-Transformer. One of these models, with the smallest architecture, uses a novel architecture configuration. This is also the model we recommend in this thesis.

In Table 3.3 and Table 3.4 we give an overview of all the models we have used in this thesis. In Table 3.3, we include which Python machine learning libraries we used to implement the models. The respective libraries are: Fastai [69], Tsai [50], EfficientNet-PyTorch [40], and Sktime [70]. Exceptionally, the SVM based model was implemented in MatLab, based on code provided by the authors of [45]. We also list the number of trainable weights for the ANN based models. The column "Year" denotes when the model type was first introduced. We note that, though SVMs were first introduced in 1992 [11], the approach we use incorporates more recent technology such as SURF image feature extraction, which was introduced in 2006 [7]. Finally, the "Sample view" column indicates which of the two data views we used for the model, as further described in Section 3.2.3.

Model	Implemented with	*Weights	Year	Sample view
AlexNet	Fastai	1.7	2012	Image
EfficientNet	Fastai & EfficientNet-PyTorch	1.3	2019	Image
InceptionTime	Fastai & Tsai	1.9	2020	2D matrix
Resnet18	Fastai	11.7	2016	Image
Resnet34	Fastai	21.8	2016	Image
Resnet50	Fastai	25.6	2016	Image
Resnet152	Fastai	60.2	2016	Image
ROCKET	Sktime	-	2020	2D matrix
SVM	Matlab	-	1992	Image
SqueezeNet v1.1	Fastai	1.2	2016	Image
TS-Transformer32	Fastai & Tsai	0.3	2020	2D matrix
TS-Transformer64	Fastai & Tsai	0.7	2020	2D matrix
TS-Transformer128	Fastai & Tsai	1.4	2020	2D matrix
xResnet18	Fastai	11.7	2019	Image

Table 3.3: A summary of models discussed in this thesis.
* The number of trainable weights in the ANN based models in millions.

3.3.2 ANN Commonalities and Training Parameters

All ANN based models have been trained with similar training parameters, which will be outlined in this section.

We used two settings of training parameters, denoted primary and secondary settings. The differences between the settings are shown in Table 3.5.

An early stopping mechanism was implemented, ending training if no decrease in validation loss was observed over a number of epochs as indicated by the *patience* parameter. Otherwise, the model trains for the maximum number of epochs, as indicated in Table 3.5.

We did not operate with any parameter freezing during training. This means all weights of the network was allowed to change during training. This is opposed to some fine-tuning schemas of pre-trained networks, where some layers of the network are left untrained.

We first trained all models with the primary settings. If a model made more than five wrong classifications on the evaluation set, they were reinitialized and trained with the secondary settings. These settings were empirically set from results in the preliminary trials, where some models showed improvements when given a lower learning rate and longer training time. Further hyper-parameter tuning was not necessary, as all models reached a performance of four or fewer wrong classifications from the 450 validation samples. As explained in Section 3.2.3, this performance is as good as we can expect, given the quality of the labeling.

Model	Model Notes
AlexNet	A classic CNN commonly used as a benchmark in image classification.
EfficientNet	Part of a recent family of CNNs, with novel variations in width, depth, and resolution scaling optimized with neural architecture search.
InceptionTime	Time series classifer built with an ensamble of CNNs based on Inception-v4 [66].
Resnet18	CNN with residual connections, which facilitates training deeper networks. 18 layers.
Resnet34	34 layers.
Resnet50	50 layers.
Resnet152	152 layers.
*ROCKET	Extracts features based on randomized kernels, global max pooling, and proportion of positive values. We used 10,000 kernels and a *RidgeLine Classifier*.
SVM	Uses SURF feature extraction [7], a RBF kernel and Sequential Minimal Optimization (SMO) [52].
SqueezeNet v1.1	Uses "fire modules" and other techniques to minimize network size, all while maintaining comparable accuracy to previous models.
*TS-Transformer32	Transformer based network for multivariate time series classification, see section Section 3.3.3 for details. Model dimension 32.
*TS-Transformer64	Model dimension 64.
*TS-Transformer128	Model dimension 128.
xResnet	Resnet based model with a collection of tweaks as presented in [27]

Table 3.4: Descriptions of the models trained in this thesis.
* Models described in more detail in Section 2.2.

The learning rates were configured based on the learning rate finder scheme proposed in [64]. We launched a mock training session with exponentially growing learning rates from 10^{-7} to 10 over a hundred iterations with a minibatch size of 10, and tracked the loss in relation to the learning rate, with early stopping upon divergence. The learning rate schedule can be seen in Figure 3.9.

Figure 3.10 shows the graph produced when setting the learning rate for the AlexNet based model used in this thesis. The two default learning rate recommendations proposed in [64] are marked by red dots, where the slope is steepest (left) and one-tenth of the minimum (right) [64]. Based on preliminary trials and literature suggestions lowering the learning rates for unbalanced datasets [12], learning rates were set as shown in Table 3.5.

We used a cross-entropy loss function for all models, as implemented in PyTorch with default parameters [16]. Cross-entropy loss is commonly used in classification tasks, where the last layer of the neural network outputs numerical scores for each class. When combined with the softmax function, as is the case in Equation (3.1), this produces a probability distribution

	Primary setting	Secondary setting
Learning rate	$\frac{Min}{30}$	$\frac{Min}{300}$
Epochs	100	200
Patience	20	40

Table 3.5: Primary and secondary training settings used for ANN models.

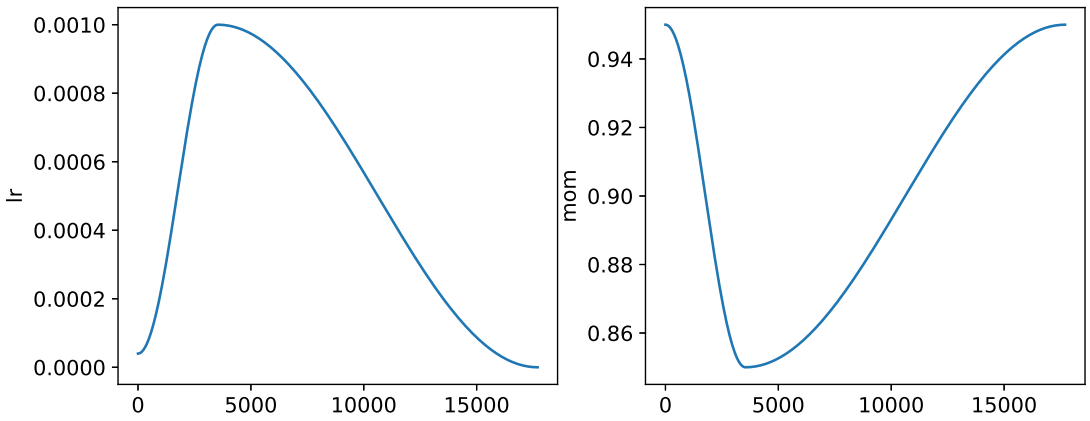


Figure 3.9: Learning rate and momentum schedule for ANN training. Maximum learning rate, and minimum momentum after 20% of epochs noted in Table 3.5.

over the classes.

$$\text{loss}(\mathbf{x}, class) = \log_2 \left(\frac{e^{\mathbf{x}[class]}}{\sum_{j=1}^{\text{n}} e^{\mathbf{x}[j]}} \right), \; n = \text{the number of classes.} \tag{3.1}$$

The softmax-cross-entropy function takes two parameters. Firstly, a vector $\mathbf{x}$ of the network's predictions for a sample, with numerical scores for each class (i.e., the non-normalized prediction scores for JAM and NoJAM). Secondly, *class* is the ground truth labels of the sample. Logits are used as $\mathbf{x}$ because the Pytorch CrossEntropyloss implementations normalize the input itself by use of the log-sum trick [24].

We used the Adam optimizer [33], which is a popular and effective optimizer [57]. It has been found to be effective in computer vision tasks [33], and is suitable for all the models we trained.

The parameters used for the Adam optimizer are summarized in Table 3.6. β_1 and β_2 are exponential decay rates for the first and second-moment estimates, on which the adaptive learning rates for each individual weight are based. ε is a small number used to prevent zero division. The optional weight decay parameter is used to apply weight decay to the weights directly. This assists in counteracting exploding gradients and incentivizes a model with smaller weight values.

3.3.3 TS-Transformer32

The TS-Transformer [73] was initially introduced with very good results on 11 benchmark datasets, arguably establishing itself as the state-of-the-art in multivariate time series clas-

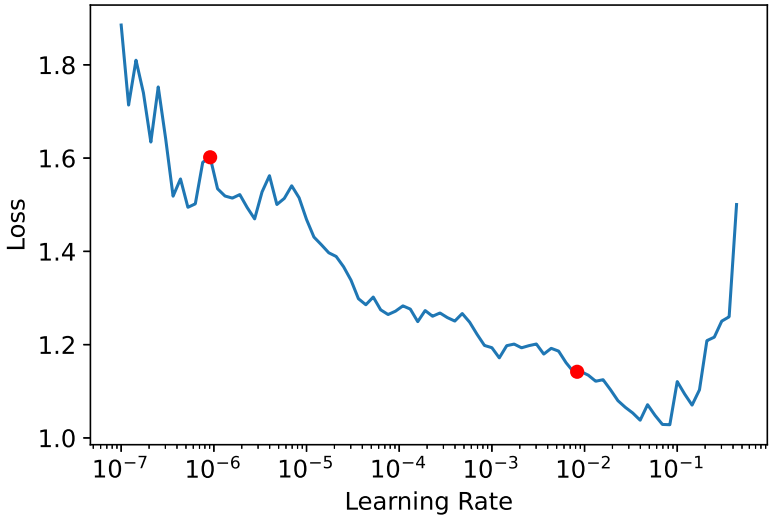


Figure 3.10: Learning rate finder.

β_1	0.9
β_2	0.99
ε	10^{-5}
Weight Decay	0.1

Table 3.6: Parameters of the Adam optimizer.

sification methods. We have focused on understanding this architeture in particular, and investigated how to reduce the model size with negligable loss of performance. In this subsection we will explain the model in more detail, and contrast our implementation to the original model found in [73].

The TS-transformer

Essentially, the TS-transformer is an adaption of the BERT architecture [18] for multivariate time series data. The emphasis is on a model architecture that is generally applicable to a wide range of tasks and datasets. This is in contrast with previous related work, where either models or data preparation are highly specialized to a specific dataset or task. The TS-transformer uses a stack of transformer encoders with the same architecture as the encoders in [68] and [18]. Unlike these papers, which use natural language text as input to the network, the TS-transformer uses multivariate time series data as input.

Where, in the NLP domain, an input text document needs to be converted to tokens, which are again turned into learned initial vectors, the TS-transformer simply projects the raw time series data to input vectors. In our dataset, the traces of a sample represent one second of data. We trivially treat the traces as "tokens" on passing them to the TS-transformer.

In [73], they investigate the use of a self-supervised pre-training task for the model, without the need for extra data. This task is inspired by the Masked Language Model task in [18], with auto-encoding of masked input data. In the case of [73], the masking is applied to consequtive sequences of data points across tokens. In our case, this is equivalent to masking data points of the same frequency, across consecutive traces.

In our trials, the performance without using the pre-training task was sufficiently good. For

this reason, we did not use the pre-training task. As reported in [73], using the pre-training does not always improve the performance. Motivated by our results without pre-training and the studies in [73], we suggest pre-training only when labeled data is scarce compared to unlabeled data.

Tuning the Architecture

As one of the stated goals of our research was to find a minimally sufficient model to perform jammer detection, we looked at how we could reduce the model parameters of the TS-transformer.

The model has mutiple architectural hyperparameters, some of which influence the size of the resulting neural network:

- The model dimension d decides the length of the internal representation of each trace in all layers of the network.
- The number of encoders in the network N_{enc} decides how many encoders are placed in a sequential stack. Each new encoder allows for a new layer complexity building on the output of the previous encoder.
- The width of the feed-forward neural network after each attention operation.
- Maximum sequence length l decides the longest sequence that the encoders can handle. This does not significantly influence model size.
- The number of attention heads N_{heads} decides the number of parallel attention computations in each encoder. This does not increase the size of the model, or its computational complexity.

Input data to the model is projected from the initial dimensionionality to the model dimension d in the input layer. Note that the time-dimension represented by the 140 traces, stays intact throughout the network. The 8001 data points in each trace is linearly projected to the model dimension d. Next, a learned position embedding is added to the data. Equation (3.2) shows this operation.

$$u_t = Wx_t + e_{pos} \tag{3.2}$$

The projection matrix W has $d \times m$ trainable parameters, where $m = 8001$ is the number of data points in each trace. A trace x_t is a feature vector of length m at time t. The learned positional embedding e_{pos} lets the model know the relative distance between traces. Positional information would otherwise be lost in the attention computation, as it treats all tokens the same.

The model dimension d and the number of encoders N_{enc} influences the model size more significantly than the other architectural parameters. Decreasing any of these parameters will also limit the complexity of the model, and thus limiting their ability to approximate functions. If we compare the Transformer network to a classic feed-forward neural network, the model dimension d decides the width of the hidden layers. The number of encoders N_{enc} decides the depth (i.e., number of hidden layers).

We choose to focus on the model dimension d, as reducing the depth of a neural network causes a rapid loss in complexity ("deeper is better than wider"). Changing the model

dimension d causes a linearly proportial change in the number of weights of the network. Consequently, successively decreased the size of the architecture by varying the model dimension. The model results can be found in Chapter 4. As a requirement for further shrinking the network, we set the maximum number of allowed misclassifications to 5.

We trained three models with settings $d \in \{128, 64, 32\}$, aptly named TS-Transformer128, TS-Transformer64, and TS-Transformer32. The other architectural parameters were kept fixed between the models, following the settings in Table 3.7.

Parameter	**Value**
Encoders N_{enc}	3
Feed-forward width	256
Attention heads	16
Sequence length	140

Table 3.7: Common architecture settings between the TS-Transformer architectures.

Refer to Table 3.8 for a detailed list of the architecture of all the networks, where the layer shape and number of parameters is from the TS-Transformer32 network. Note that there is no softmax activation function after the output layer, as the loss function already includes a stable softmax. See Section 3.3.2 for details on this.

Layer Type	**Layer Shape**	***Parameters**
Input Layer:		
Linear	8001×32	256064
Dropout		
First encoder, Attention:		
Linear	32×32	1024
Linear	32×32	1024
Linear	32×32	1024
Linear	32×32	1024
Dropout		
BatchNorm1d	140	280
First encoder, Fully connected layers:		
Linear	32×256	8448
GELU		
Linear	256×32	8224
Dropout		
BatchNorm1d	140	280
Second Encoder, Attention:		
Linear	32×32	1024
Linear	32×32	1024
Linear	32×32	1024
Linear	32×32	1024
Dropout		
BatchNorm1d	140	280
Second Encoder, Fully connected layers:		
Linear	32×256	8448
GELU		
Linear	256×32	8224
Dropout		
BatchNorm1d	140	280
Third Encoder, Attention:		
Linear	32×32	1024
Linear	32×32	1024
Linear	32×32	1024
Linear	32×32	1024
Dropout		
BatchNorm1d	140	280
Third Encoder, Fully connected layers:		
Linear	32×256	8448
GELU		
Linear	256×32	8224
Dropout		
BatchNorm1d	140	280
Output Layer:		
Flatten		
Linear	64×2	8962
		Sum = 329,010

Table 3.8: A list of the layers of the TS-transformer network, with the output shape and number of paramters in each layer.
* Includes bias parameters, unlike the Layer Shape.

Chapter 4

Results and evaluation

In this chapter, we present the results, obtained from the approach we have laid out in the previous chapter. Section 4.1, Evaluation metrics, explains the metrics we used to evaluate our classifiers. Section 4.2, Validation Scores, will provide an overview of the performances of all the models on the validation dataset. Based on this performance, we narrow down our lists to four candidate models, and elaborate more on them in Section 4.3 Four models chosen for testing. We also present the evaluation results of these models on the test dataset.

4.1 Evaluation metrics

As common metrics to evaluate all models, we used accuracy, precision, recall, and F1-score. Note that the calculations for the latter three are based only on predictions on samples labeled as Jam. For a detailed explanation of the these metrics, see Section 2.1.2. The recall metric is of particular interest, as it represents the ratio of the actual Jam samples that the classifiers recognize. In other words, with $100 - \text{recall}$, we get the percentage of undetected illegal jammers. The precision metric indicates how reliable a jammer prediction is, based on the ratio between true positive and false positives. The F1-score is a weighted average between precision and recall, and accuracy is the overall classification accuracy of both Jam and NoJam signals.

4.2 Validation Scores

The validation dataset was only used to determine early stopping during training, and we consider the results on the validation dataset to be representable of the models' generalization performance. As noted in Section 3.3.2, extensive hyperparameter search was not necessary to obtain good results.

A summary of the evaluation performances on the validation dataset for all models is shown in Section 4.2. For the ANN-based models, the reported results are taken from the epoch noted in the "Epoch" column. These are the epochs of the model with the highest F1-score. In parenthesis, we include the average score of each metric over the ten epochs following the peak F1-score.

From the validation scores, we see that all ANN-based models achieve over 99% accuracy, with F1-scores from 96.2 to 99.1. The non-ANN SVM and ROCKET models, achieve sig-

Model	Accuracy	Precision	Recall	F1-score	Epoch
SVM	92.0	92.0	85.2	88.5	-
ROCKET	96.9	97.0	99.5	98.6	-
Alexnet*	99.3 (98.5)	98.1 (95.3)	96.3 (92.4)	97.2 (93.8)	132
EfficientNet	99.3 (98.6)	98.1 (96.2)	96.3 (92.2)	97.2 (94.0)	43
InceptionTime	99.8 (98.2)	100.0 (95.1)	98.1 (90.2)	99.1 (91.9)	71
Resnet18	99.8 (99.0)	100.0 (99.0)	98.1 (92.2)	99.1 (95.4)	58
Resnet34	99.6 (99.2)	100.0 (99.2)	96.3 (94.3)	98.1 (96.7)	80
Resnet50	99.8 (99.4)	100.0 (99.8)	98.1 (94.8)	99.1 (97.2)	59
Resnet152	99.1 (97.7)	98.1 (95.8)	94.4 (84.6)	96.2 (89.8)	54
Squeezenet1.1	99.6 (99.1)	98.1 (97.6)	98.1 (94.6)	98.1 (96.0)	107
xResnet18	99.1 (98.7)	98.1 (97.3)	94.4 (91.9)	96.2 (94.5)	45
TS-Transformer32*	99.1 (97.8)	98.1 (90.2)	94.4 (92.4)	96.2 (91.1)	112
TS-Transformer64*	99.3 (98.0)	98.1 (92.5)	96.3 (92.2)	97.2 (92.1)	111
TS-Transformer128	99.1 (95.4)	98.1 (81.5)	94.4 (92.8)	96.2 (85.4)	62

Table 4.1: Peak model results on the validation dataset. The metrics are multiplied by 100, and 100 is the maximum score.
* Models trained with secondary training settings (see Table 3.5).

Table 4.2: Summary of results.

nificantly lower accuracy, though the ROCKET-based classifier maintains a relatively high F1.

Some models, such as Resnet18 and InceptionTime, correctly classified all NoJam validation samples and only mislabeled one Jam sample. An interesting observation is that xResnet18, which is supposed to be an improvement of Resnet18, was achieved a lower performance. However, differences between classifier performances are very small for all ANN models.

When considering the complexity of the models, as measured by the number of model parameters, the TS-Transformer32 performs the best. It maintains comparable performance to the other models, by using half as many parameters as the second smallest model (the TS-Transformer64), and a tenth of the parameters of the third smallest classifier (SqueezeNet1.1).

None of the models achieve 100% recall. In an attempt to increase recall, we experimented with a weighted loss function, increasing the training signal for misclassified Jam samples. We tried ratios up to 1:10 for Jam-to-NoJam penality, but all configurations resulted in a worse results.

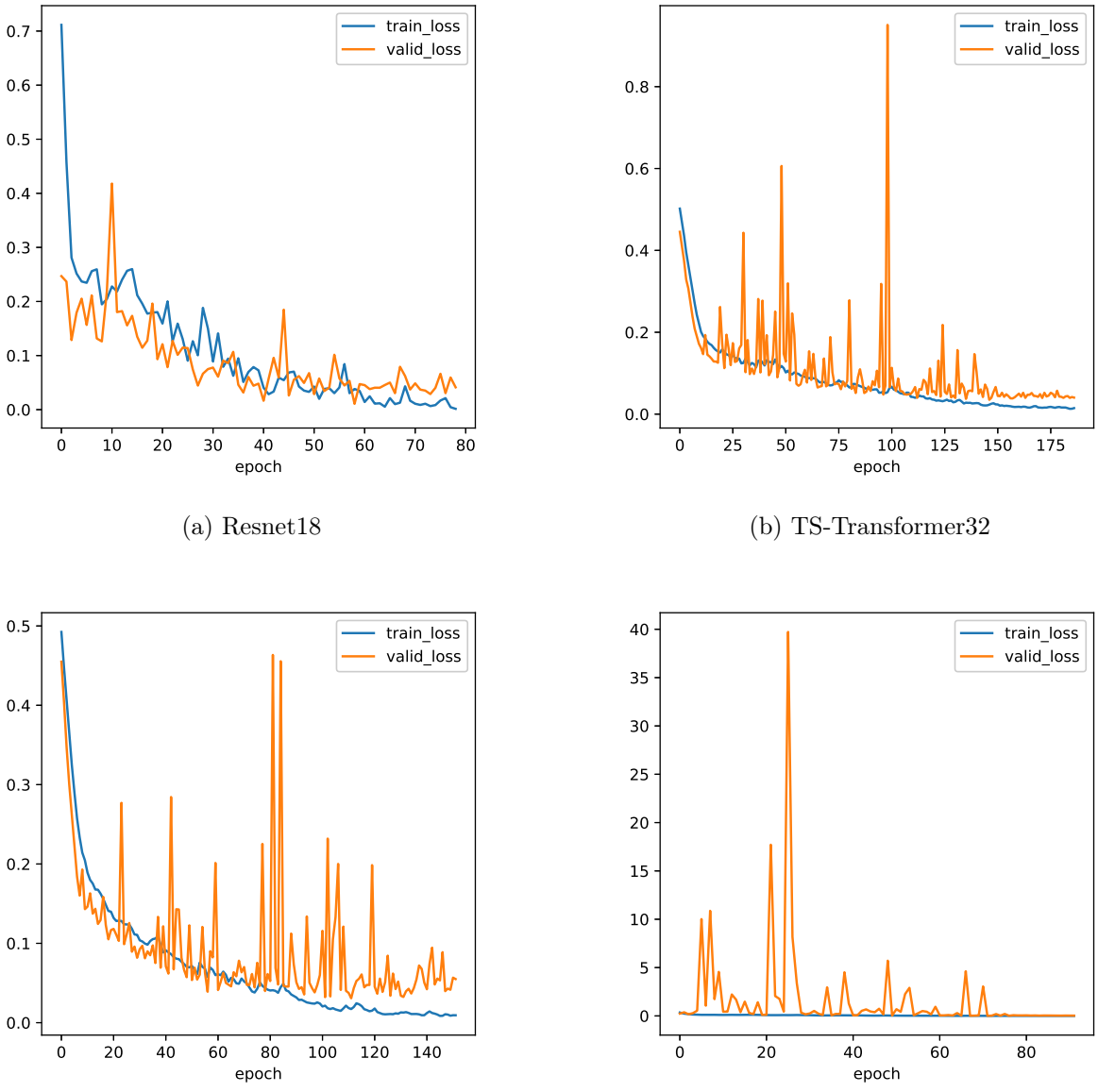
We also observed that our early stopping mechanism performed well. All models converged to peak F1-score in the epochs prior to the early stopping was triggered. We also found that early stopping with validation loss performs better than early stopping based on the F1-score. High fluctuations in the F1-score in each training epochs would lead to early stopping with sub-optimal F1-scores, as the "patience" for improvements would run out.

4.3 Four models chosen for testing

From the large number of classifiers we evaluated in the previous section, we chose to focus further on four classifiers in particular.

Resnet18 was the CNN image classifier with the lowest number of parameters that achieved only one misclassification. Inception time also achieved only one misclassification, still with a relatively low number of parameters. The InceptionTime classifier is featured in many multivariate time series classification papers, and is thus relevant for comparison. Lastly, we pick two versions of the time series transformer. We chose the TS-Transformer32 model, because it has the lowest number of parameters of any model. We also choose the TS-Transformer64 to compare two models from the same family.

4.3.1 Training and validation loss



(a) Resnet18

(b) TS-Transformer32

(c) TS-Transformer64

(d) InceptionTime

Figure 4.1: Training and validation loss of selected models over training epochs.

Figure 4.1 shows the trend in training and validation loss over epochs for some select models. The Resnet architectures (Figure 4.1a) all showed a relatively smooth decrease in loss, with few spikes compared to some other models. The transformer-based models (Figure 4.1b and Figure 4.1c) show some small spikes in training and validation loss during training, similarly to other trained models. InceptionTime (Figure 4.1d) produced large spikes in training and validation loss, similar to those observed with the Squeezenet model.

4.3.2 Classications from Validation and Inspection of Missed Predictions

In this section, we will elaborate some specific classification results for the four selected models on the validation dataset. Table 4.3 shows the confusion matrices for the four models.

We use the model state at the epoch with peak F1-score, as noted in Section 4.2. We observed that all the models misclassify at least one Jam sample, while only the TS-Transformer32 misclassified a NoJam sample. On inspecting the spectrograms of the mislabeled samples, we learned that the mislabeled Jam samples pertain to particularly weak jammer signals, and the mislabeled NoJam sample pertains to a particularly strong NoJam-event. Figure 4.2 shows some example spectograms of mislabeled samples. The middle sample shows strong occurence of wideband noise (or possibly a mislabeled jammer) and the right spectogram is from a weak jamming signal.

Resnet18

	JAM_p	$NoJAM_p$
JAM_a	53	1
$NoJAM_a$	0	396

TS-Transformer32

	JAM_p	$NoJAM_p$
JAM_a	51	3
$NoJAM_a$	1	395

TS-Transformer64

	JAM_p	$NoJAM_p$
JAM_a	51	3
$NoJAM_a$	0	369

InceptionTime

	JAM_p	$NoJAM_p$
JAM_a	53	1
$NoJAM_a$	0	396

Table 4.3: Confusion matrices for chosen models on the validation set. $Label_a$ denotes the actual label, while $Label_p$ denotes the predicted label.

To increase our recall score we considered implementing a threshold adjustment to the pseudo probabilities outputted by our final softmax function, i.e., labeling all samplins with a softmax score for JAM above 0.3 as a jammer event (instead of a fair spilt at 0.5). When we observed the probabilities pertaining to the misclassified samples we learned that this would not be a good soltuion, as the as the classifiers were "too confident" in the prediction when misclassifying samples.

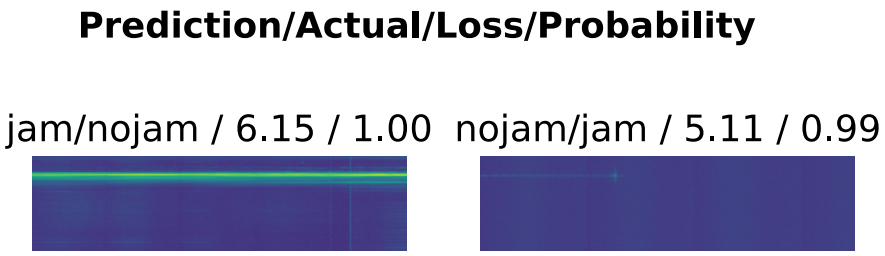


Figure 4.2: Inspection of mislabeled samples in the validation set

4.3.3 Evaluation on the Test Dataset

Metric scores				
	Accuracy	Precision	Recall	F1
Resnet18	98.8	84.7	90.6	87.5
TS-Transformers32	98.7	96.9	74.4	84.2
TS-Transformers64	98.9	92.3	83.7	87.7
InceptionTime	80.1	16.5	86.0	27.7

Table 4.4: Performance metrics of the four models on test data.

Towards the end of the project we received a new batch of 952 datasamples that we used for testing of our four chosen models. The confusion matrices in Table 4.5 show that the number of misclassifications was higher compared to the validation dataset. Restnet18, TS-Transformer32 and TSTransfomer64 maintain a good performance with 10 to 12 misclassifications, but the performance of the InceptionTime model fell drastically due to misclassifying NoJam samples. The performance of the various models can be found in Table 4.4.

Resnet18

	JAM_p	$NoJAM_p$
JAM_a	39	4
$NoJAM_a$	7	902

TS-Transformer32

	JAM_p	$NoJAM_p$
JAM_a	32	11
$NoJAM_a$	1	908

TS-Transformer64

	JAM_p	$NoJAM_p$
JAM_a	36	7
$NoJAM_a$	3	906

InceptionTime

	JAM_p	$NoJAM_p$
JAM_a	37	6
$NoJAM_a$	183	762

Table 4.5: Confusion matrices for chosen models on the test set. $Label_a$ denotes the actual label, while $Label_p$ denotes the predicted label.

Excluding the InceptionTime model, our classifiers maintain relatively high accuracy, but suffer from a lower recall. Among our top three classifiers, the F1 score remains similar, but we observe significant differences in recall and precision scores. This is due to the fact that the models perform differently on the two classes. The Resnet18 model obtains a 90.6 recall score as it only misclassified four jam samples. On the other hand, the TS-transformer32 has the same accuracy, but only obtains a 96.9 Precision on the Jam class due to more false positive classifications.

To better understand the increase in misclassifications, we inspected the samples misclassified by the top three models. Spectograms of some of these samples can be found in Figure 4.3. About half of these samples show a predominant noise level, identified by the green tone to the plotted spectograms. This is caused by background noise in the vicinity of the measuring station, and these noise values are under-represented in the training and validation sets. Many of them hail from a new data source which does not appear in the training and validation datasets (the station is titled "ODD"). We also observe that low power jammers (possibly from the stations with monitoring antenna further from the RFI sources, such as at the ODD station) were hard to classify. The classifiers also tended to mix very high-powered wideband jammers with wideband noise.

After discussions with Nkom, we further speculate that fast-approaching vehicles can cause

a low amount of traces with visible jam signal. Seeing as we operate with only one trace per second, this might provide too little data for the classifiers that are trained on longer jamming signals. Furthermore, we expect there is more noise in the labels of the test dataset than in the training and validation dataset, given the analysis and relabeling described in Section 3.2.2.

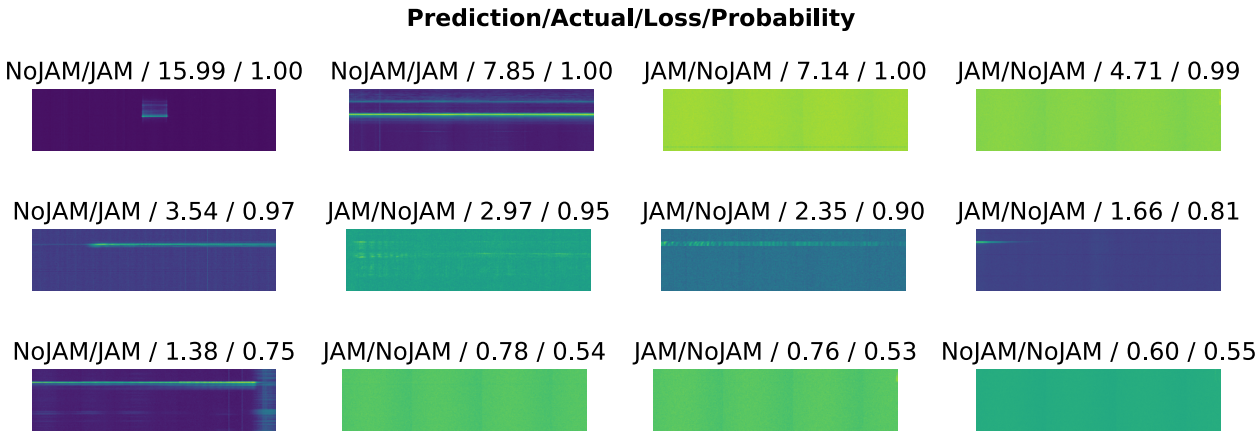


Figure 4.3: Inspection of mislabeled and hard to classify samples from the testset.

Chapter 5

Conclusion

Based on the goals of this thesis and our obtained results, we will now provide a final discussion on machine learning applied to jammer detection. First, we state that the jammer detection problem is solvable to a high accuracy standard by machine learning algorithms. However, noise in the labeling and differences between monitoring stations make it difficult to achieve perfect scores. This is to be expected in real-world datasets. Most jammer-caused RFI events contain distinguishable patterns, and the classifiers we have evaluated detect these patterns.

There is a notable discrepancy between state-of-the-art machine learning classifiers applicable to this task, and the methods applied to jammer detection in the literature. We have found clear benefits of using state-of-the-art multivariate time series and image classification models compared to the simple approaches commonly observed in the literature. We obtained significantly improved performance on our dataset over commonly applied methods, such as the top-performing model of a related paper [45].

Furthermore, we achieve comparably strong performance to the other evaluated methods with a reduced computational cost in our TS-Transformer32 model. This shows the potential of ANN-based approaches even for resource-limited applications. Based on the CNN-based approaches we evaluated, an increase in model size could have some benefits to the performance. However, none of our findings indicate that models above 10 to 15 million parameters would achieve any increase in performance.

We also find that a 99% accuracy is achievable. We observed up to 99.5% accuracy on our validation dataset, which was only used to stop training upon convergence. When we evaluated our test dataset of 952 previously unseen samples, including samples from entirely new sources with new factors affecting the samples, we still achieved a 98.9% accuracy.

Furthermore, we found that a 100% jammer recall score proved to be challenging to obtain. From the oputput probabilities of the models, we observed that lowering the threshold for prediction would not be sufficient. We also saw that weighted backpropagation in favor of the underrepresented jammer samples harmed the overall classification performance. We will present a potential solution to mitigate this in Section 5.1 Advice to Nkom.

We have found that lower temporal resolution in the data can be adequate when detecting the presence of jammers, as we obtained our results on samples with only one trace per second. However, we add that one trace per second may be too low for reliably classifying samples of particular short duration (i. e fast driving cars). Commonly in the literature, much higher temporal data resolution is used.

5.1 Future work

Finally, we will concretely state our advice to Nkom regarding implementing machine learning approaches to detect jammer interference in the GNSS bands. We also include suggestions and potential ideas for future works.

Advice to Nkom

For a minimally resource expensive solution, our recommendation is an approach based on our TS-Transformer32. However, if resources are sufficient, more computationally intensive classifiers such as Resnet18 may yield a slight increase in performance. The main obstacle to high jammer recall and precision scores (meaning percentage of detected jammers and reliably predicted jammer presence) seem to be certain subgroups of hard-to-classify jammer and non-jammer events. To elaborate, jammers events with particularly weak signal strength or non-jammer-interference of wideband nature with high signal levels are examples of this. We suggest that a third label "Unsure", be introduced so that the classifiers could potentially separate these samples under a different label. An automatic detection system could inform human personnel on predicting either "jammer" or "Unsure". This approach could still filter out a significant portion of non-jammer interference and provide jammer recall scores much closer to 100. However, this suggestion would require human personnel to re-label the samples and introduce the "Unsure" class.

Another improvement, independent of the first suggestion, is fine-tuning individual models for each station. This approach could allow the classifiers to learn the unique characteristics of the samples generated from a particular station and adjust for variations such as background noise, which is often location-specific.

Finally, the configuration of monitoring stations in terms of the bandwidth, number of data points, and traces per second should be standardized. A uniform configuration could eliminate the need for any preprocessing, and classifiers could run directly on the generated data point values. We suggest standardizing data points to around 8000 frequencies per trace, and setting the number of sampled traces per second to a common configuration that all stations can reliably maintain.

Improvement for Algorithms

Beyond the suggestions stated above, we also wish to include some of the ideas we did not put into practice.

Preprocessing techniques designed to remove the influence of background noise could be attempted. A simple scheme could be implementing a global minimum value m such that each data point d is replaced by $max(m, d)$.

If labeled data from a particular monitoring station is sparse, we suggest experimenting with the pre-training task proposed in [73], in combination with fine-tuning on the available labeled data.

Appendix A

Newsarticles on Jamming

Nå flyr de i «blinde»

Norsk Luftambulanse er første operatør i Europa som har utviklet et system for instrumentinnflyvning til andre steder enn flyplassene. Foto: Fredrik Naumann/Felix Features

Frøydis Braathen

Tidligere måtte legehelikoptrene stå på bakken når sikten var dårlig. Takket være GPS-teknologi kan de nå gjøre betydelig flere oppdrag enn før.

Lave skyer, dårlig sikt og fare for ising har ofte ført til at mange legehelikopter-oppdrag må kanselleres. Men et egenutviklet nett av forhåndsbestemte ruter og innflyvninger gjør at helikoptrene på Østlandet nå kan fly ved hjelp av GPS.

På én uke i november ble 19 av 21 oppdrag fra Lørenskog-basen løst ved hjelp av instrumentflyvning. For få år siden ville disse oppdragene sannsynligvis blitt kansellert, og alvorlig skadede pasienter hadde måttet vente lenger på akutthjelp, ifølge basesjefsflyger Bent Næss ved Norsk Luftambulanse på Lørenskog.

– Uten GPS-verktøyet ville sannsynligvis 30–40 av oppdragene i oktober og november ha blitt kansellert. På mange av de andre basene på Østlandet er situasjonen den samme, sier Næss.

Norsk Luftambulanse er første operatør i Europa som utviklet et system for instrumentinnflyvning til andre steder enn flyplassene.

Siden 2006 er et nettverk av kartruter i lav høyde gradvis bygget opp. Forhåndsbestemte innflytningsruter fantes fra før til flyplasser, men ikke til sykehus og andre landingssteder som ofte brukes av ambulansehelikoptrene. Luftambulansen har bygget ut et helt nytt rutenett, i samarbeid med den statlige Luftambulansetjenesten HF og de regionale helseforetakene.

I tillegg gir nesten 100 værkamerastasjoner svært nøyaktig informasjon om været langs flyruten.

Skal bygges ut flere steder

Norsk Luftambulanse har nå innflyvninger til over 60 sykehus, baser og andre landingssteder i Norge. Nettverket er aller best bygget ut på Østlandet, men arbeidet med å utvide nettverket i resten av Norge, pågår for fullt. Nord-Norge er et stort satsingsområde.

Organiseringen av Helse-Norge i dag baserer seg på at pasienter skal komme seg raskt inn til de store sykehusene. Uten GPS-nettverket, ville ikke like mange akuttpasienter fått så rask hjelp som i dag, ifølge flyoperativ fagsjef Erik Normann i Stiftelsen Norsk Luftambulanse.

– I Europa er vi nesten alene om denne teknologien. Sveits har begynt å bygge ut samme type nettverk, men Norsk Luftambulanse er fortsatt i front, sier Normann.

Hindres av gammelt regelverk

Selv om teknologien gjør det mulig å fly i svært dårlig vær, må fortsatt en del legehelikopter-oppdrag avvises på grunn av et strengt regelverk. Selv om 21 oppdrag lot seg løse i uke 47, måtte likevel åtte oppdrag avvises på grunn av værforholdene.

– Vi ønsker å få lov til å ta av og lande i enda dårligere vær, men hindres av et gammelt regelverk, som ikke tar høyde for de nye verktøyene. Vi er i dialog med det europeiske luftfartssikkerhetsbyrået EASA for å prøve å få endret reglene, og dermed kunne hjelpe enda flere pasienter på raskeste måte, sier Normann.

FAKTA

Luftambulansen på Lørenskog

Har døgnkontinuerlig beredskap av to Airbus legehelikoptre med mannskap.

Basen dekker store deler av Østlandet og har i overkant av 2000 iverksatte oppdrag pr. år.

På landsbasis blir hvert 10. legehelikopteroppdrag avvist eller avbrutt på grunn av dårlig vær.

Det betyr at 700 syke eller skadede pasienter hvert år ikke får potensielt livreddende legehjelp på raskeste måte.

BERGENS TIDENDE MANDAG 17. DESEMBER 2018 NYHETER // 7

KRITISK: Marius Malmo er basesjefspilot for Luftambulansen i Bergen. Han opplevde selv å miste GPS-signalet på tur for å plukke opp en pasient i oktober. – Vi ramler ikke ut av himmelen selv om vi mister signalet, men hvis det er overskyet eller dårlig sikt, må vi enten legge om ruten, eller avbryte oppdraget. Det er klart det kan få konsekvenser for pasientene, sier han.

SVART SKJERM // Piloten var overlatt til det han så ut vinduet for å finne veien til den kritisk syke pasienten.

Han måtte finne veien selv, mistet GPS-signal

LARS KVAMME
larskvamme@gbt.no
foto: FRED IVAR UTSI KLEMETSEN
foto: ØRJAN DEBIZ

3. oktober i år rykket luftambulansen ut på et akutt oppdrag. Etter kun tre minutter var redningsmannskapet i luften på vei til en kritisk syk pasient.

Helikopteret fløy sørover, over E39 på Vallaheiane. Piloten fulgte ruten som var plottet inn på et digitalt kart. De avanserte GPS-systemene viste dem hvor de var hele veien.

Akuttsentralen fulgte helikopteret ferd på sine skjermer. Slik kunne sentralen gi viktige beskjeder om når legen ankom skadestedet.

Plutselig forsvant helikopteret fra kartet. GPS-signalet var borte.

Nå var det opp piloten å finne veien selv. Han måtte navigere hovedsakelig basert på det han så 1500 fot under seg.

Risikerer å måtte avbryte

Marius Malmo er basesjefspilot i Bergen. Han var ikke pilot på

JAMMER: Dette er en GPS-jammer som likner den som slo ut GPS-systemene til Luftambulansen.

den første flygingen, men flydde samme rute senere samme dag og opplevde det samme.

I det samme området mistet de GPS-signalene i rundt to minutter.

– Vi ramler ikke ut av himmelen selv om vi mister signalet, men hvis det er overskyet eller dårlig sikt, må vi enten legge om ruten, eller avbryte oppdraget. Det er klart det kan få konsekvenser for pasientene, sier Malmo.

Hvis en pasient allerede er om bord i helikopteret, kan tap av signal bety at de må dra til et annet behandlingssted enn opprinnelig planlagt.

Tredje oktober var det klarvær. Til tross for at de erfarne pilotene måtte avbryte den planlagte innflygingen basert på GPS, klarte de fint å navigere seg frem til pasienten uten at det fikk helsemessige konsekvenser.

– Det er første gang vi opplever dette i Bergen på samme stedet flere ganger, så vi meldte fra til politiet, sier Malmo.

Fant kilden på Vallaheiane

Politiet meldte fra videre til Norges sjøvdspåsser når det kommer til å avsløre signalforstyrrelser av denne typen: Nasjonal kommunikasjonsmyndighet (Nkom).

I Bergen har Nkom to spesialutrustede biler med peileutstyr for å finne frem til syndere. Kort tid etter forespørselen til Nkom, hadde de sirklet inn området på Vallaheiane hvor forstyrrelsen kom fra.

Siste delen måtte senioringeniør Øystein Bakke fra Nkom gjøre til fots. Da brukte

NKOM-BILEN: Sjefingeniør i Nkom, Åsgeir Henriksen, er glad for at det settes fokus på GPS jamming. Han er stadig utu med bilen for å søke etter ulik signalstøy. Det er ikke alltid støyen skyldes bevisst jamming – det kan også være forårsaket av ødelagt elektronikk.

han et håndholdt instrument og peileantenne.

Etter en tids manuell peiling, ble kilden til forstyrrelsen identifisert. En parkert bil i området, med en GPS-jammer koblet i sigarettennerens 12 voltsuttak.

– Vi har ikke mange slike hendelser nasjonalt hvert år, men det er et økende problem. Dessuten er potensialet for å forstyrre veldig stort siden GPS-signaler er svake, så dette er noe vi tar på største alvor, sier Per Eirik Heimdal, avdelingsdirektør i teknologiavdelingen i Nkom.

I hast har Nkom også vært i kontakt med Luftambulansen i forbindelse med en tilsvarende forstyrrelse i Ardal.

Heimdal forklarer at utstyret som brukes til å forstyrre signaler er relativt lett tilgjengelig på nett, til tross for at det er strengt forbudt å selge, eie og bruke i Norge.

Årsakene til at det brukes kan være mange, forklarer Heimdal:

- I Europa har noen land innført veipris ing som baserer seg på en GPS-brikke i bilen som registrerer hvor mye du bruker avgiftbelagte veistrekninger, og noen bruker en GPS-jammer for å unndra seg disse gebyrene.
- Man kan ødelegge for flåtestyring som noen transportfirma har, noe som gjør at kjøretøyene blir unyttige for arbeidsgiver.
- Til slutt kan det brukes for å skjule tyvegods. Mange nye biler og andre kostbare produkter kan ha en GPS-mottaker innebygget for å rapportere posisjon. Forstyrres GPS-signalet, kan ikke bilen/produktet finnes igjen siden posisjonen ikke kan rapporteres.

Vil øke oppmerksomheten

Generelt har GPS-jammere begrenset rekkevidde, men kan likevel skape store problemer.

– Luftambulansen er én ting, men også i nærheten av flyplasser, kan dette skape store problemer. Heldigvis har flyrutfikk mange systemer å spille på, sier Heimdal, og legger til at de vil ha økt oppmerksomhet mot slikt utstyr fremover.

– På generelt grunnlag må vi nok øke oppmerksomheten gjennom kontroll av fedevnsbruken, og dessuten gjennom forebyggende tiltak og informasjon rettet mot salg og innførsel av utstyr for jamming, sier han.

Høye gebyrer

Nkom ser svært alvorlig på bruk av jammere.

– Dette er utstyr som er laget for å skape interferens og forstyrrelser av GPS-systemer eller andre samfunnsviktige kommunikasjonstjenester. I slike saker mener Nkom at det bør legges en klart merkbar reaksjon for å hindre at bruk av jammere øker, sier Heimdal.

Eieren av GPS-jammeren på Vallaheiane måtte umiddelbart skru den av, og det ble varslet et overtredelsesgebyr. Størrelsen på gebyret i saker som denne kan komme opp i titalls tusen kroner, ifølge avdelingsdirektøren.

FAKTA

GPS-jammer

- Radioutstyr som sender ut radiobølger som overdøver andre signaler på samme frekvens.
- GPS-signalene som kommer fra satellitter i verdensrommet er svake. Hvis man sammenlikner dem med visking, kan man si at en GPS-jammer bruker.
- Det er ulovlig å eie, bestille og bruke GPS-jammere i Norge.
- De minste får plass i 12V-uttaket på bilen, men kan likevel slå ut GPS-signalene i et stort område.

DØGNET

Redigert av: Linda S. Vatnelid

HAVARERT: Arbeidet ved KNM «Helge Ingstad» ble i går innstilt inntil vinden løyer. FOTO: ALICE ARNTSEN AUS

Arbeidet på «Helge Ingstad» innstilt

BERGEN KL. 11.52: I går meldte Forsvaret at arbeidet ved den havarerte fregatten «Helge Ingstad» var innstilt på grunn av kraftig vind. Først når været tillater det, starter arbeidet igjen. Havaristen har ligget i ro de siste dagene og dykkeroperasjoner har pågått, ifølge Forsvaret. Målet er at fregatten skal bøves. Dykkerfartøy og kranleKteren «Rambiz» har seilt til Hanøytangen for å laste om bord utstyr til neste operasjon, ifølge Forsvaret.

Slukket stort bål

ÅSANE, KL. 13.51: Brannmannskaper rykket i går ut for å slukke et stort bål med mye svart røyk i Dalavegen i Åsane. De kom kjapt til stedet og fikk bålet raskt slukket, meldte 110-sentralen på Twitter.

Vei ryddet etter steinras

BØMLO KL. 15.09: Fylkesvei 541 ved Lykling på Bømlo ble åpnet igjen på strekningen mellom Nyheim og Ekornsætre etter å ha vært stengt etter et steinras. – Store steiner blokkerte hele veien, opplyste trafikkoperatør Hanne Norheim hos Vegtrafikksentralen.

To tatt for ran på Nygårdshøyden

BERGEN SENTRUM KL. 06.12: Klokken 04.14 fikk politiet melding fra en mann om at han var blitt ranet av to personer ved Johanneskirken på Nygårdshøyden i Bergen sentrum. Han fortalte at han var blitt truet med kniv og fratatt et pengebeløp. Politiet kom til stedet og pågrep to personer. Patruljen fant pengebeløpet og en kniv gjemt. De to ble pågrepet og satt i arrest, meldte Vest politidistrikt på Twitter.

HAUKELI: Det var uvær i fjellet søndag. FOTO: STATENS VEGVESEN

Slik var sikten for bilførere på Haukeli

BERGEN KL. 06.34: Det var både stengte veier og kolonnekjøring på flere fellesvogganger søndag morgen på grunn av uvær. Noen fjellovergange kunne åpnes igjen utover dagen, men på en rekke strekninger var det fortsatt kolonnekjøring i går ettermiddag. Av erfaring er det ofte flere ulykker når snøen begynner å komme, og det er denne typen vær. Bilførere ble i går oppmodet om å kjøre etter forholdene.

Bil havnet på taket

BERGEN KL. 04.49: Like over klokken 03.00 natt til søndag fikk politiet melding om et trafikkuhell i Steinsvikvegen. En bil var havnet på taket. Sjåføren stakk fra stedet, ble innhentet og er mistenkt for ruskjøring, meldte politiet på Twitter. Veien ble stengt ved krysset Søvikmarka fordi en lyktestolpe ble knekt og det var strømledninger i veien. Ved 07.30-tiden var begge felt åpne igjen.

Bibliography

[1] Online; accessed 11. Jun. 2021. Nov. 2018. URL: https://www.regjeringen.no/contentassets/abd1dec7647a4c22aaef7d93046e3f2b/pa-rett-sted-til-rett-tid.pdf.

[2] [Online; accessed 11. Jun. 2021]. Jan. 2020. URL: https://www.regjeringen.no/contentassets/1febbbb2c4fd4b7d92c67ddd353b6ae8/no/pdfs/ki-strategi.pdf.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." In: *arXiv preprint arXiv:1409.0473* (2014).

[4] Dzmitry Bahdanau et al. "End-to-end attention-based large vocabulary speech recognition." In: *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2016, pp. 4945–4949.

[5] Sungwan Bang and Myoungshic Jhun. "Weighted support vector machine using k-means clustering." In: *Communications in Statistics-Simulation and Computation* 43.10 (2014), pp. 2307–2324.

[6] Roland Bauernfeind et al. *Analysis, detection and mitigation of incar GNSS jammer interference in intelligent transport systems*. Deutsche Gesellschaft für Luft-und Raumfahrt-Lilienthal-Oberth eV, 2013.

[7] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features." In: *European conference on computer vision*. Springer. 2006, pp. 404–417.

[8] Iz Beltagy, Matthew E Peters, and Arman Cohan. "Longformer: The long-document transformer." In: *arXiv preprint arXiv:2004.05150* (2020).

[9] Daniele Borio, Cillian O'Driscoll, and Joaquim Fortuny. "GNSS jammers: Effects and countermeasures." In: *2012 6th ESA Workshop on Satellite Navigation Technologies (Navitec 2012) & European Workshop on GNSS Signals and Signal Processing*. IEEE. 2012, pp. 1–7.

[10] Daniele Borio et al. "Impact and detection of GNSS jammers on consumer grade satellite navigation receivers." In: *Proceedings of the IEEE* 104.6 (2016), pp. 1233–1245.

[11] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. "A training algorithm for optimal margin classifiers." In: *Proceedings of the fifth annual workshop on Computational learning theory*. 1992, pp. 144–152.

[12] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. "A systematic study of the class imbalance problem in convolutional neural networks." In: *Neural Networks* 106 (2018), pp. 249–259.

[13] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: A library for support vector machines." In: *ACM Transactions on Intelligent Systems and Technology* 2 (3 2011). Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm, 27:1–27:27.

[14] Kan Chen et al. "Abc-cnn: An attention based convolutional neural network for visual question answering." In: *arXiv preprint arXiv:1511.05960* (2015).

[15] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system." In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[16] *CrossEntropyLoss — PyTorch 1.8.1 documentation*. [Online; accessed 11. Jun. 2021]. June 2021. URL: https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html.

[17] Angus Dempster, François Petitjean, and Geoffrey I Webb. "ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels." In: *Data Mining and Knowledge Discovery* 34.5 (2020), pp. 1454–1495.

[18] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." In: *arXiv preprint arXiv:1810.04805* (2018).

[19] Bhaskar Dhariyal et al. "An Examination of the State-of-the-Art for Multivariate Time Series Classification." In: *LITSA, ICDM 2020* (2020).

[20] Christopher P Diehl and Gert Cauwenberghs. "SVM incremental learning, adaptation and optimization." In: *Proceedings of the International Joint Conference on Neural Networks, 2003.* Vol. 4. IEEE. 2003, pp. 2685–2690.

[21] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale." In: *arXiv preprint arXiv:2010.11929* (2020).

[22] *Electromagnetic radiation - Effect of gravitation*. [Online; accessed 11. Jun. 2021]. June 2021. URL: https://www.britannica.com/science/electromagnetic-radiation/Effect-of-gravitation#ref59180.

[23] *EUROCONTROL voluntary ATM incident reporting (EVAIR) safety bulletin 19*. [Online; accessed 11. Jun. 2021]. June 2021. URL: https://www.eurocontrol.int/publication/eurocontrol-voluntary-atm-incident-reporting-evair-safety-bulletin-19.

[24] Gregory Gundersen. *The Log-Sum-Exp Trick*. [Online; accessed 11. Jun. 2021]. Apr. 2021. URL: https://gregorygundersen.com/blog/2020/02/09/log-sum-exp.

[25] Zhimei Hao, Wen Yu, and Wei Chen. "Recognition method of dense false targets jamming based on time-frequency atomic decomposition." In: *The Journal of Engineering* 2019.20 (2019), pp. 6354–6358.

[26] Kaiming He et al. "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[27] Tong He et al. "Bag of tricks for image classification with convolutional neural networks." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 558–567.

[28] Cho-Jui Hsieh et al. "A dual coordinate descent method for large-scale linear SVM." In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 408–415.

[29] Dichao Hu. "An introductory survey on attention mechanisms in NLP problems." In: *Proceedings of SAI Intelligent Systems Conference*. Springer. 2019, pp. 432–448.

[30] Jun Huang and Yunxin Zhao. "An energy-constrained signal subspace method for speech enhancement and recognition in white and colored noises." In: *Speech Communication* 26.3 (1998), pp. 165–181.

[31] *IEEE 521-2019 - IEEE Standard Letter Designations for Radar-Frequency Bands*. Online; accessed 11. Jun. 2021. June 2021. URL: https://standards.ieee.org/standard/521-2019.html.

[32] Taha Khan, Jerker Westin, and Mark Dougherty. "Classification of speech intelligibility in Parkinson's disease." In: *Biocybernetics and Biomedical Engineering* 34.1 (2014), pp. 35–45.

[33] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." In: *arXiv preprint arXiv:1412.6980* (2014).

[34] Alexander Kolesnikov et al. "Big transfer (bit): General visual representation learning." In: *arXiv preprint arXiv:1912.11370* 6.2 (2019), p. 8.

[35] Mingming Kong et al. “Radio ground-to-air interference signals recognition based on support vector machine.” In: *2015 IEEE International Conference on Digital Signal Processing (DSP)*. IEEE. 2015, pp. 987–990.

[36] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks.” In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[37] Thach Le Nguyen et al. “Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations.” In: *Data mining and knowledge discovery* 33.4 (2019), pp. 1183–1222.

[38] Jason Lines, Sarah Taylor, and Anthony Bagnall. “Time series classification with HIVE-COTE: The hierarchical vote collective of transformation-based ensembles.” In: *ACM Transactions on Knowledge Discovery from Data* 12.5 (2018).

[39] Zhou Lu et al. “The Expressive Power of Neural Networks: A View from the Width.” In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/32cbf687880eb1674a07bf717761dd3a-Paper.pdf.

[40] lukemelas. *EfficientNet-PyTorch*. [Online; accessed 11. Jun. 2021]. June 2021. URL: https://github.com/lukemelas/EfficientNet-PyTorch.

[41] Daniel Pak-Kong Lun et al. “Wavelet based speech presence probability estimator for speech enhancement.” In: *Digital Signal Processing* 22.6 (2012), pp. 1161–1173.

[42] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. “Effective approaches to attention-based neural machine translation.” In: *arXiv preprint arXiv:1508.04025* (2015).

[43] Dhruv Mahajan et al. “Exploring the limits of weakly supervised pretraining.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 181–196.

[44] Man-Wai Mak and Hon-Bill Yu. “A study of voice activity detection techniques for NIST speaker recognition evaluations.” In: *Computer Speech & Language* 28.1 (2014), pp. 295–313.

[45] Ruben Morales Ferre, Alberto de la Fuente, and Elena Simona Lohan. “Jammer classification in GNSS bands via machine learning algorithms.” In: *Sensors* 19.22 (2019), p. 4841.

[46] Ruben Morales Ferre, Elena Simona Lohan, and Alberto De la Fuente. *Image datasets for jammer classification in GNSS*. Zenodo, Aug. 2019. DOI: 10.5281/zenodo.3370934. URL: https://doi.org/10.5281/zenodo.3370934.

[47] Oscar Henrik Biti Næss. “Hareide om ny rapport: – Gode grunner til å vurdere satellittbasert veiprising.” In: *NRK* (Mar. 2020). URL: https://www.nrk.no/norge/hareide-om-ny-rapport_-_-gode-grunner-til-a-vurdere-satellittbasert-veiprising-1.14935182.

[48] Nkom. “About Nkom.” In: *Nkom* (Feb. 2021). URL: https://www.nkom.no/english/about-nkom.

[49] Stephen O’Hara and Bruce A Draper. “Introduction to the bag of features paradigm for image classification and retrieval.” In: *arXiv preprint arXiv:1101.3354* (2011).

[50] Ignacio Oguiza. *tsai - A state-of-the-art deep learning library for time series and sequential data*. Github. 2020. URL: https://github.com/timeseriesAI/tsai.

[51] John Platt. “Sequential minimal optimization: A fast algorithm for training support vector machines.” In: (1998).

[52] John Platt. “Sequential minimal optimization: A fast algorithm for training support vector machines.” In: (1998).

[53] Roland Priemer. *Introductory signal processing*. Vol. 6. World Scientific, 1991, pp. 1–8.

[54] Esmat Rashedi, Hossein Nezamabadi-Pour, and Saeid Saryazdi. “GSA: a gravitational search algorithm.” In: *Information sciences* 179.13 (2009), pp. 2232–2248.

[55] Philipp Richter, Ruben Morales Ferre, and Elena-Simona Lohan. *GATEMAN project – Wide-bandwidth, high-precision GNSS and jammer raw data*. Zenodo, Apr. 2019. DOI: 10.5281/zenodo.2654322. URL: https://doi.org/10.5281/zenodo.2654322.

[56] Anders Rødningsby et al. "RFI Monitoring of GNSS Signals on Norwegian Highways." In: *Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020)*. 2020, pp. 3536–3549.

[57] Sebastian Ruder. "An overview of gradient descent optimization algorithms." In: *arXiv preprint arXiv:1609.04747* (2016).

[58] Alejandro Ruiz et al. "The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances." In: *Data Mining and Knowledge Discovery* 35 (Mar. 2021), pp. 1–49. DOI: 10.1007/s10618-020-00727-3.

[59] Fu Ruo-Ran. "Compound jamming signal recognition based on neural networks." In: *2016 Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC)*. IEEE. 2016, pp. 737–740.

[60] Samferdselsdepartementet. "Satellittbasert veiprising for tungtransport." In: *Regjeringen* (Mar. 2020). URL: https://www.regjeringen.no/no/dokumenter/satellittbasert-veiprising-for-tungtransport/id2692943.

[61] Samferdselsdepartementet. "Veiprising av tunge kjøretøy kan være samfunnsøkonomisk lønnsomt." In: *Regjeringen* (Mar. 2020). URL: https://www.regjeringen.no/no/aktuelt/veiprising-av-tunge-kjoretoy-kan-vare-samfunnsokonomisk-lonnsomt/id2692934.

[62] Patrick Schäfer and Ulf Leser. "Multivariate time series classification with WEASEL+ MUSE." In: *arXiv preprint arXiv:1711.11343* (2017).

[63] Connor Shorten and Taghi M Khoshgoftaar. "A survey on image data augmentation for deep learning." In: *Journal of Big Data* 6.1 (2019), pp. 1–48.

[64] Leslie N Smith. "Cyclical learning rates for training neural networks." In: *2017 IEEE winter conference on applications of computer vision (WACV)*. IEEE. 2017, pp. 464–472.

[65] *Standard data exchange format for frequency band registrations and measurements at monitoring stations*. https://www.itu.int/rec/R-REC-SM.1809/en. Accessed: 2020-12-15.

[66] Christian Szegedy et al. "Inception-v4, inception-resnet and the impact of residual connections on learning." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 1. 2017.

[67] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6105–6114.

[68] Ashish Vaswani et al. "Attention is all you need." In: *arXiv preprint arXiv:1706.03762* (2017).

[69] *Welcome to fastai | fastai*. [Online; accessed 11. Jun. 2021]. June 2021. URL: https://docs.fast.ai.

[70] *Welcome to sktime — sktime documentation*. [Online; accessed 11. Jun. 2021]. June 2021. URL: https://www.sktime.org/en/latest.

[71] Qizhe Xie et al. "Self-training with noisy student improves imagenet classification." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10687–10698.

[72] Ying Yang and Lidong Zhu. "An Efficient Way for Satellite Interference Signal Recognition Via Incremental Learning." In: *2019 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE. 2019, pp. 1–5.

[73] George Zerveas et al. "A Transformer-based Framework for Multivariate Time Series Representation Learning." In: *arXiv preprint arXiv:2010.02803* (2020).

[74] Renyuan Zhang and Siyang Cao. "Support vector machines for classification of automotive radar interference." In: *2018 IEEE Radar Conference (RadarConf18)*. IEEE. 2018, pp. 0366–0371.