Automatic classification of air tracks using raw video from a conventional surveillance radar

Master's thesis in Computer science: Algorithms, Languages and Logic

ANGELICA STRANDBERG
JOAKIM MILLESON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

Master's thesis 2020

Automatic classification of air tracks using raw video from a conventional surveillance radar

ANGELICA STRANDBERG
JOAKIM MILLESON

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2020

Automatic classification of air tracks using raw video from a conventional surveillance radar
ANGELICA STRANDBERG
JOAKIM MILLESON

Supervisor: Ashkan Panahi, CSE
Advisor: Anders Averö, Saab Surveillance
Examiner: Devdatt Dubhashi, CSE

Master's Thesis 2020
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2020

Automatic classification of air tracks using raw video from a conventional surveillance radar
ANGELICA STRANDBERG
JOAKIM MILLESON
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Detecting the type of an airborne vehicle, such as an air plane or helicopter, spotted by a radar, can be very time consuming and expensive if done by examining sensor outputs. This might cause a problem if a potential threat has been detected since the type, and thereafter identity, of a vehicle must be determined rapidly.

This thesis investigates the possibility of classifying jets and helicopters from raw video, collected by a conventional surveillance radar, using a neural network. The raw video is transformed from the time domain into the frequency domain using Fast Fourier Transformation and then plotted as either spectrograms or range doppler plots. In order to remove noise, the plots are filtered using the unsupervised algorithm k-means clustering. Different neural networks such as convolutional neural network (CNN), recurrent neural network (RNN) and convolutional recurrent neural network (CRNN) are then trained and evaluated on the plots remaining after filtering. The plots are treated as images by the network.

The results show that the CNN model trained on range doppler plots performs the best with a test accuracy of 90% and a validation accuracy of 83%. Our conclusion from this work is that it seems to be possible for a CNN to distinguish jets from helicopters when it is fed with images of Fast Fourier Transformed raw video, but some more research has to be done. Mainly, the model must be trained and tested on more data. The contribution this thesis provides is not only that it seems to be possible to classify jets and helicopters from raw video data in almost real time, depending on the performance of the computer being used, but also that it seems to be possible to receive a great accuracy when using a CNN for solving the task.

Keywords: Computer, science, computer science, engineering, project, thesis, time series classification, CNN, RNN, k-means clustering.

Acknowledgements

We would like to thank our supervisor Ashkan Panahi and our examiner Devdatt Dubhashi for all support and help. We would also like to thank Saab Surveillance and our company advisor Anders Averö, Håkan Warston and Jacob Lundberg at Saab Surveillance for their great feedback and assistance throughout the work.

Angelica Strandberg and Joakim Milleson, Gothenburg, June 2020

Word list

ADS-B signal *Automatic Dependent Surveillance-Broadcast signal*
: A signal for carrying information about an airborne vehicle, such as its vehicle type and position.

FT *Fourier Transform*
: A method for transforming a continuous signal from the time domain into the frequency domain.

FFT *Fast Fourier Transform*
: A computationally efficient method for transforming a discrete signal from the time domain into the frequency domain. It calculates DFT.

DFT *Discrete Fourier Transform*
: It describes what is calculated using FFT.

STFT *Short-time Fourier Transform*
: It applies FT on windows of the data and hence, creates spectrograms.

RNN *Recurrent Neural Network*
: A neural network type which uses gained knowledge from previous iterations.

LSTM *Long Short-Term Memory*
: A RNN type which filters information gained in previous iterations by using cell states and gates.

GRU *Gated Recurrent Units*
: Similar to LSTM, but it does not use cell states and has different gates.

CNN *Convolutional Neural Network*
: A neural network type which uses filters on input data to extract features.

CRNN *Convolutional Recurrent Neural Network*
: A combined convolutional and recurrent neural network. It can also include LSTM or GRU.

Contents

List of Figures

List of Tables

1
Introduction

Radar has been used for decades [3] to discover and track vehicles such as air planes and helicopters. Today, radar is amongst other things used for surveiling territories with aim of discovering, classifying and thereafter identifying vehicles, including airborne ones. Being able to classify tracked airborne vehicles – air tracks – is important in itself, but it is also important to perform it as fast as possible. This is so that decisions on possible actions can be promptly taken from the knowledge about the vehicle's identity and intention, especially if it is a potential threat. Analysing radar signals and output from sensors is not only expensive, but also very time consuming. Thus, finding a way to classify airborne vehicles rapidly and with high precision, in order to later on be able to identify it as friendly is of great interest.

Classifying objects fast has been a popular topic for years [4], [5], [6]. Previously, many experiments and investigations in the area of audio classification have been carried out. Audio classification is similar in essence to the classification problem considered in this project since both use time series data, originally located in the time domain. Many conventional techniques rely on first transforming the data from the time domain to the frequency domain using a Fourier Transform and thereby forming spectrograms which can be treated as images. These techniques have had encouraging results, see Chapter 3.

In addition, using different types of machine learning techniques for classifying audio signals has been previously investigated, as presented in Chapter 3. These attempts show that using only convolutional neural networks for classifying sounds results in worse performance in addition to long training times, compared to using for instance recurrent neural networks. Recurrent neural networks show great potential for classifying time series. However, convolutional neural networks usually perform well on image classification.

1.1 Thesis statement and contribution

This thesis continues the research in the area of time series classification by investigating the possibility of classifying airborne vehicles in real time. This is done by using a neural network and raw video data received from a conventional surveillance radar as input. The input to the classifier consists of plotted 2D data arrays produced from the raw video data, which is further described in Section 2.2.1 and 2.2.2.

Several different methods for signal preprocessing, but mainly techniques for machine learning and artificial intelligence were investigated and compared. This in order to find the combination of signal processing steps and a machine learning method that resulted in the fastest and the most accurate model.

This thesis was done in collaboration with Saab Surveillance.

1.2 Ethical considerations

There are a few ethical issues to consider. Firstly, even though letting a neural network classify detected objects has benefits in terms of speed, only relying on it might be risky. Even if the classifier predicts correctly in most cases, it will never be perfect and therefore it is important to not let it decide completely on its own. The classifier can consist as a decision basis by suggesting classifications, which in turn are used when deciding actions, but a human should always have the final say and not blindly rely on the classifier.

Moreover, as often when using artificial intelligence and training a neural network for automatically classifying objects and computing predictions, the network created can potentially replace one or more humans. Thereby, fewer employees in surveiling and analysing radar data might be needed. On the other hand, all systems need maintenance, support and further development when technologies are changed and new ones are added, and the work in this thesis is no exception. Therefore, the demand for employees would likely remain unchanged or perhaps even increase in this area. Furthermore, letting a neural network classify results in more accurate predictions in a short amount of time.

2

Theory

This chapter describes the theory needed to carry out the project's different implementation and investigation tasks. The chapter is divided in the same way as the implementation is. It starts with a background about the data and how to process it before usage and concludes with a description of the classification techniques investigated in this work and how to handle imbalanced data. Imbalanced data was a huge problem in this thesis. All described methods presented in this chapter have been used in similar contexts in related work. How they have been used previously is presented in Chapter 3.

An important reservation is that this chapter's purpose is mainly to refresh terms and concepts related to this thesis and as a result does not include in-depth details.

2.1 ADS-B signal

The data used for this thesis is recorded from a radar and marked with different information about the discovered object, known as the ADS-B signal. ADS-B stands for *Automatic Dependent Surveillance-Broadcast* and is a technique used in aviation to get information about the object the signal is broadcasted from. Examples of such information is identity, velocity, altitude and position. [7] The ADS-B signal can for instance be used for retrieving the type of an airborne vehicle, in other words if it is a helicopter or jet for example, which is used to label the data.

The ADS-B signal is sent using Mode-S signalling schemes, employing automatically and periodically sent messages. Mode-S is an aviation transponder interrogation mode and it helps ensure that the transponder of the aircraft's radio system is only contacted and interrogated when necessary. [8]

2.2 Signal processing

The following sections describes different tools and techniques used for preprocessing the data and the radar echoes (time series) it contains in order to create range doppler plots and spectrograms to be used as input data to the neural networks.

2.2.1 Range doppler plots

Doppler effect, also known as doppler shift, describes how the frequency of a wave travelling from or towards or reflected by an object changes as the object is moving. When the object is moving, the wave fronts in front of the object will compress and the frequency increases, whilst the wave fronts behind will spread out and the frequency decreases. This is illustrated in Figure 2.1 below. The change in frequency is due to its relation to wavelength; they are inverse proportional. The doppler shift increases with the object's speed. [9]

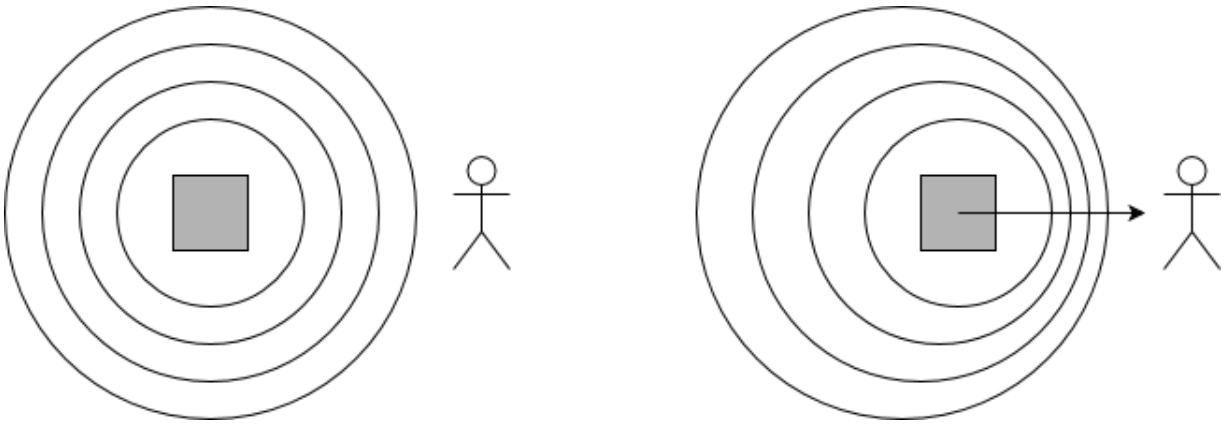

Figure 2.1: The figure to the left shows an object, illustrated by a grey square, standing still. This causes the waves to be evenly spread out around the object. In the figure on the right, on the other hand, the object is moving right and as a result, the wave fronts on the right become compressed, whilst the waves on the left are more spread out. This is known as the doppler effect. This is seen from the non-moving person's perspective in the two figures.

When using radars, doppler shifts are created because of the movement of both the radar and the target object reflecting the wave radiated from the radar. If the object travels towards the radar the wave is compressed, whilst when it is travelling away it is spread out. An important note is that the ground causes no doppler effect when using a ground radar. [9] The doppler effect can be illustrated in a so called range doppler plot which shows how the frequency is related to range and doppler, i.e. distance and velocity. The different colours in such plot show the magnitudes of the frequencies present. Two examples of range doppler plots formed using the data for this thesis can be seen in Figure 2.2 below. Range doppler plots can be treated as images and used as an input to a classifier in order to visualise differences in doppler effect for air planes and helicopters respectively.

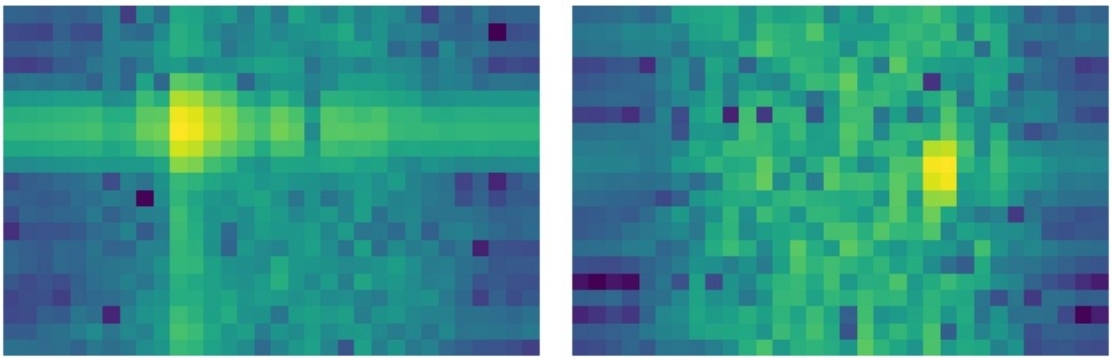

Figure 2.2: The figure shows range doppler plots generated from the dataset. The leftmost is from a helicopter data sample, whilst the rightmost is from a jet data sample. The x-axis represents doppler channels (velocity) and the y-axis range bins. A lighter, or more yellow, colour on the image indicates a higher magnitude.

Range doppler plots are created by plotting the result of applying *Fast Fourier Transform*, denoted by FFT, on time series. FFT is an algorithm for transforming a discrete time signal [10] and calculates the *Discrete Fourier Transform*, or DFT for short. A transformation from the time domain into the frequency domain is thereby carried out and the time axis is replaced by a frequency axis. As a result, the frequencies found in the signal and their respective magnitudes can be extracted. [10]

The time complexity of FFT is $O(n \log n)$, where n is the number of data points [11], and the output a list of complex numbers forming the calculated frequencies in the window. Each output is therefore an amplitude of a frequency bin. The range of the frequencies present is half of the number of the sampling rate. The absolute values of the complex values are thereafter computed, followed by normalization. This gives a 2D matrix, where each row is a window frame number and each column a frequency bin. The values show how strong each frequency is. [10]

2.2.2 Spectrograms

Even though there are several benefits with using FFT, there is one large drawback. When performing the transformation and receiving the frequencies, all time information will be lost. The consequence is that it will not be possible to say in which order the frequencies arrived, which can cause many problems when using them. The solution to this problem is to use spectrograms. [10]

A spectrogram is a graphical illustration of a signal which uses both time and frequency information, one for each of the two axes of the plot [10]. The horizontal axis is the time axis, whilst the vertical is the frequency axis. In the origin the oldest and lowest frequency is found and from there towards the positive side of the two axes the time and frequency increases. [12] The magnitude or amplitude is shown in the plot by using colours. The brighter the colour, the stronger frequency. [10] Spectrograms are created by dividing the signal into windows and then computing DFT for each window by using FFT, followed by combining the computations. This

is known as applying *Short-time Fourier Transform*, or STFT, on the signal. [11] Each window represents a time index and contains frequencies. How many windows to divide the signal into varies from problem to problem. Preferably, the windows should overlap to make sure no frequencies are missed. Usually there is an overlap of 25% to 75%. [10] Finally, the combined results are plotted in order to create a spectrogram [11].

Before using FFT, a weighting window such as hamming is applied. The weighting window's purpose is to handle discontinuity of the signal [10], which in turn emerges from dividing the signal into windows. Examples of spectrograms are shown in Figure 2.3.

Figure 2.3: The figure shows spectrograms generated from the dataset. The leftmost is from a helicopter data sample, whilst the rightmost is from a jet data sample. The x-axis represents time and the y-axis frequency. A lighter, or more yellow, colour on the image indicates a higher magnitude.

A great benefit with spectrograms is that they can be interpreted as images and therefore methods for image classification, for instance neural networks, can be used [11].

2.3 Classification techniques in machine learning

Machine learning is about learning from data and iteratively improving the prediction of the output, without explicitly programming it. One of its uses is in the area classification and it includes several methods that can be used for classifying airborne vehicles. [13]

This section describes the classification techniques investigated in this project, even though there are other methods as well. Firstly, there is a short introduction to neural networks, followed by a more thorough description of the methods to be investigated and applied. Finally, ways to handle imbalanced datasets, such as the one used for this thesis, are presented.

2.3.1 Supervised and unsupervised learning

Supervised learning is about training on a dataset where the data points are already marked with the correct classification or ground truth [14]. Such a dataset is denoted as a labelled dataset [15]. Supervised learning has been shown to be efficient for solving different kinds of tasks, including classification and prediction problems [14].

Unsupervised learning is often used in cluster analysis and for discovering patterns in datasets which are not labelled, denoted as unlabelled datasets [16].

2.3.2 k-means clustering

k-means clustering is an unsupervised machine learning algorithm for forming a, by the variable k, specified number of groups from an unlabelled dataset according to similar features of the data points it contains. The algorithm forms k centroids with random positions to start with and then iteratively updates their position as it is adjusting which data points that appear to be similar and not. This optimization is completed when the centroids are no longer moving or a pre-defined number of iterations have been run. [17] An example of a before and after picture illustrating the effect of k-means clustering is shown in Figure 2.4 below.

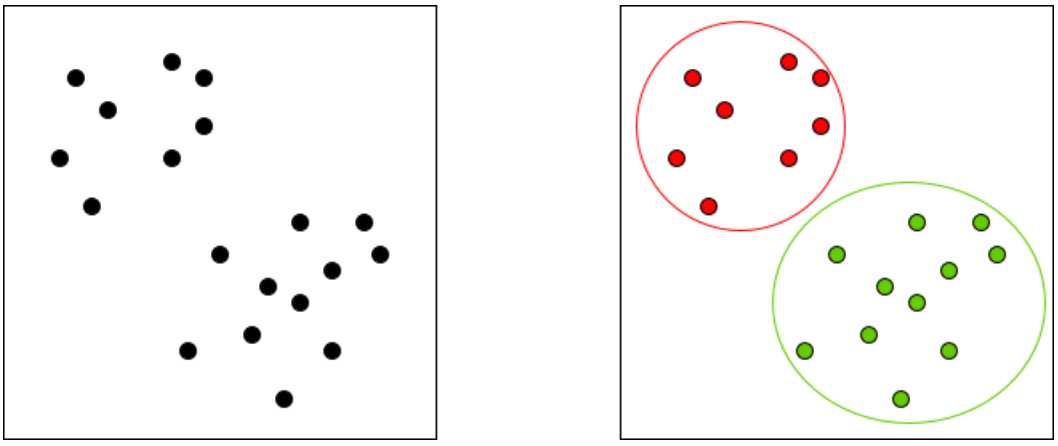

Figure 2.4: An example of k-means clustering with the data points plotted to the left (before clustering) and the data points divided into two clusters ($k = 2$) with one cluster for each colour on the right.

2.3.3 Neural network

A neural network is structured in three different layers. Each layer consists of one or more nodes, which in turn consists of information for extracting features (one feature per node) and the node's value. The value is passed from previous nodes and is thereafter multiplied by its weight and finally summed together with a bias. The first layer is the input layer and where the data is given to the network for eventually calculating and solving a given task. The middle part of the network is called the hidden layers and is where the calculations take place. Finally there is

the output layer which outputs the computed prediction(s). [18] The structure of an example neural network can be seen in Figure 2.5 below.

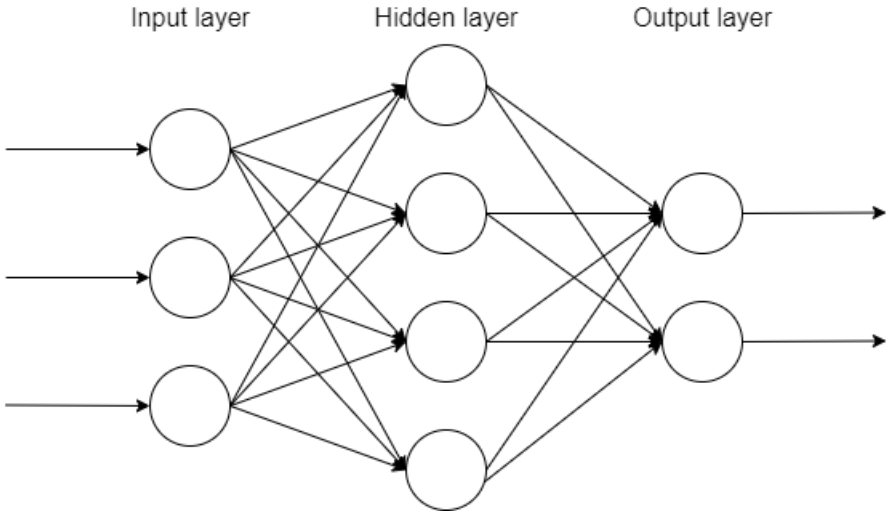

Figure 2.5: An example neural network where each circle represents a node in the layer. Note that the hidden layer can consist of multiple layers.

A neural network can be used for classification problems, amongst other things. The neurons in the output layer then represent the different categories an object can be classified as for the current task. If there are only two classes, the problem can be formulated as a binary classification problem with only one output neuron that gives the probability of the object belonging to the positive class. [19] Otherwise, one output neuron per class is used.

In a neural network, the parameters and weights are updated according to two functions; feed-forward and backpropagation. The feed-forward is the neural network's error or cost function and it can for example be the cross-entropy cost function. In addition, a gradient descent algorithm is used to update the parameters for each iteration run, but to be able to execute such an algorithm a gradient must be calculated for the current parameters. The gradient is calculated using error backpropagation which propagates the error backwards through the network, starting at the output layer.

2.3.3.1 Convolutional neural network

A *convolutional neural network*, or CNN for short, is a neural network that applies a kernel on the input data. The layer doing this is known as a convolutional layer. The kernel is usually of smaller dimension than the data and moves from left to right over the input to extract features. The kernel also reduce the data to a smaller form which is easier to process, without losing its features. The result is known as filters. A pooling operation is applied on subsets of the filters. The pooling operation is used to extract dominant features which are invariant in the dimensions of rotation and position. It also, like the kernel, reduces the dimension in order to reduce the

amount of computational power needed. There are two types of pooling operations; max pooling and average pooling. Max pooling takes the maximum value of a subset whilst average pooling takes the average of a subset. To extract higher level features, multiple convolutional and pooling layers can be applied. Lastly, a CNN contains a fully connected layer for classification. If the data is of higher dimension than one, the output of the pooling layer is flattened to fit the fully connected layer. [20] Figure 2.6 shows a simple example of a CNN.

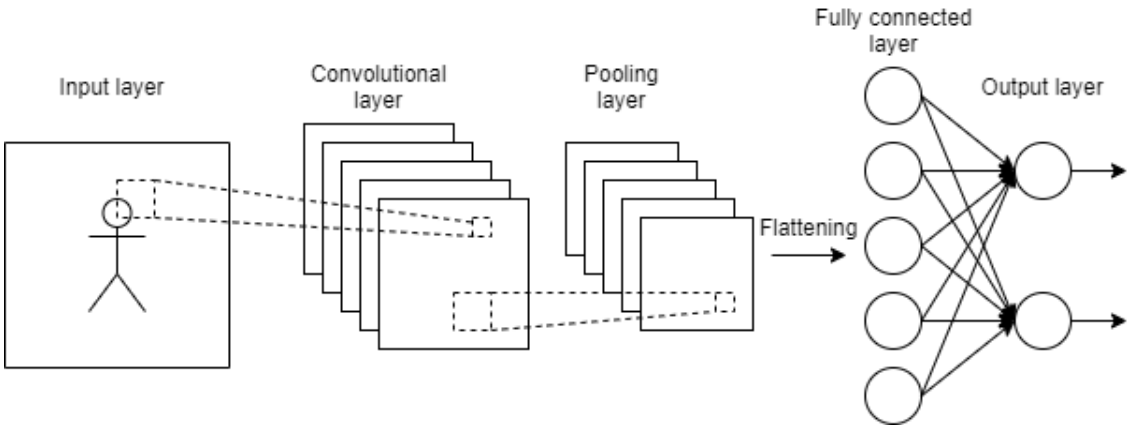

Figure 2.6: An example CNN where the dashed square on the input layer is a kernel that swipes over the input and gives the result in the smaller dashed box in the convolutional layer. The convolutional layer is followed by a pooling layer and the result from the pooling layer is flatten and used as an input to a fully connected layer. The circles represents nodes.

CNN is commonly used for image recognition, such as classification of handwritten letters and digits. It is also used for signal processing and generating synthesised voice. CNN constitutes the basis of Google Assistant's voice synthesizer [21].

2.3.3.2 Recurrent neural network

Recurrent neural network, or RNN for short, is neural networks repeated after each other, where every network gets the input data as well as information from the previous network as input (apart from the first one which only receives the input data). As a result, information gained and knowledge learnt in previous neural networks in the chain are looped back to be used in the subsequent one. In other words, the output of one such neural network uses a combination of the current input and the previous neural network's output. Knowledge therefore travels from one neural network in the chain to the next. [22] RNN also has the benefit of being able to handle multiple inputs and outputs [23]. A typical RNN is illustrated in Figure 2.7 below, with the rectangle representing a chain of neural networks. The loop back carries information learnt.

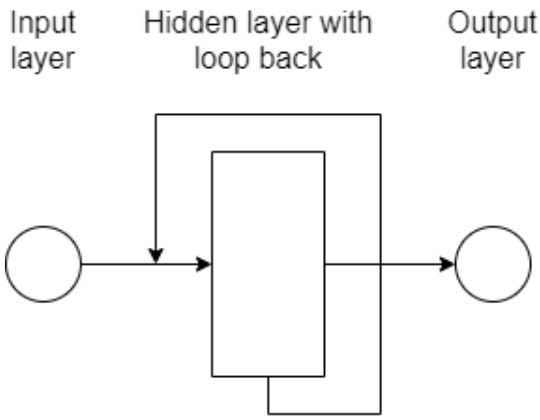

Figure 2.7: An example RNN where each circle represents a node in the layer. The network's characteristic loop back is found in the hidden layer.

RNN has multiple uses, including speech recognition [22] and image captioning [24]. It also has several benefits – amongst others, it can learn which part of an audio stream (frequency data) that is of use. Its two different versions LSTM and GRU, further described in Section 2.3.3.2.1 and 2.3.3.2.2 below, give state-of-the-art results in many applications of RNNs [25]. Moreover, LSTM and GRU are able to transmit knowledge from one layer to another without causing so-called vanishing or exploding gradients. [26]

Since the vanilla architecture of RNN is built so that current knowledge is based on previous information, dependency problems can occur. If the current prediction needs to be built on a prediction made far away in the chain of networks to a high extent, the distance can cause a problem when calculating the gradients. Either the computed gradient is very low or very high. This is known as vanishing and exploding gradients, respectively. Vanishing gradients cause the network to not learn anything, whilst exploding gradients result in a model that crashes. [27]

Moreover, deciding what in the past that is of interest when predicting the current output can be rather tricky. *Attention* is a technique that is combined with RNN to help overcome this issue. It also makes learning easier and improves the performance for many different tasks. [28]

2.3.3.2.1 LSTM

LSTM stands for *Long Short-Term Memory* and is a solution to the short memory problem described in 2.3.3.2. To be able to control which information to save and which to forget, LSTM uses gates in cell states, which in turn work as the network's memory. The gates learn what information that is useful and what is not during training. There are three types of gates used; forget gate, input gate, and output gate. [25]

The forget gate makes the decisions of what is relevant information and not by letting the input pass through a sigmoid function. A sigmoid function transforms

the values so that they will be numbers between zero and one only. The closer to zero a value is, the more likely it is that the information is not useful and the other way around the closer to one a value is. [25]

The input gate, on the other hand, is used for updating the cell state by using the previous hidden state and the current input, transformed by using a sigmoid function. The same two inputs are also sent into a tanh function in order to make it easier to regulate the network. The output from the sigmoid function decides what is useful from the output of the tanh function. The output gate then decides the next hidden state, which is used to make predictions. [25]

2.3.3.2.2 GRU

GRU is an abbreviation of *Gated Recurrent Unit* and is a newer version of a RNN. It is quite similar to LSTM and also has gates, even though the gates themselves are slightly different. Moreover, GRU does not use cell states. The two types of gates GRU has are reset gate and update gate. The update gate works about the same way as LSTM's forget and input gates and hence decides what to keep and what to forget. The reset gate's task, on the other hand, is to determine what information from the past to delete. [25]

The benefit of GRU over LSTM is that GRU has fewer tensor operations, resulting in faster training. Despite this, it is not possible to say that GRU is better than LSTM because it also depends on the problem and task to solve. Therefore, both are usually implemented and compared before deciding which one to use. [25]

2.3.4 Overfitting and underfitting

Overfitting is a common description of a neural network model that is too well adapted to the training data and therefore performs poorly on testing data. Underfitting means that the model is not adapted well enough for the input data [29]. Both overfitting and underfitting are problems that have to be solved.

There are several ways of handling overfitted models. One way of solving it could be using fewer features and thereby making the model less complex or increase the regularisation parameter for instance [29]. Regularisation adds penalties during optimisation of the loss function [30]. For an underfitting model, the opposite can be tested in order to improve the model [29].

2.4 Handling imbalanced datasets

In the context of machine learning and artificial intelligence, datasets with more or less perfectly equally distributed features are wanted. Unfortunately, the datasets are to a varying extent usually imbalanced, meaning there are more data objects of one feature compared to another [31]. If the difference is small, this might not be a problem. However, if it is large, the imbalanced dataset must be handled

before being used in order to compensate for the lack of data that has one or more specific features. There are different ways to handle an imbalanced dataset and some common ones are described in Section 2.4.1, 2.4.2 and 2.4.3 below.

2.4.1 Sampling

There are two ways to balance an imbalanced dataset using sampling; undersampling and oversampling. Undersampling is done on the majority class of the dataset, in which samples are randomly selected to make the number of samples match the size of the minority class. The selected samples will be used in the dataset and the non-selected are discarded. This can be carried out when the minority class is large enough to be used. The disadvantage of undersampling is that useful data is thrown away and the dataset becomes smaller, which might affect the training of a classifier. [31, 32]

Oversampling is the opposite of undersampling. When oversampling, samples of the minority class are copied until the number of samples matches the size of the majority class. The advantage of oversampling is the increase in size of the dataset. However, since there will be copies of the datasamples, overfitting is more likely [31, 32].

2.4.2 Cost-Sensitive Learning

When training a classifier, the misclassifications are often equally costly, i.e. misclassifying one class is just as bad as misclassifying another. As a consequence, the classifier will adapt towards the majority class of the imbalanced dataset. This can be fixed by adding a cost function that gives a higher value when misclassifying a sample from the minority class compared to the majority class. This is known as adding class weights. As a result, the classifier will be forced to prioritise the minority class more [32].

2.4.3 Data Augmentation

Data augmentation is a way to increase the size of a dataset without collecting more data. This is done by modifying samples in the data. Images, for instance, can be rotated, flipped and magnified, which has been shown to increase the accuracy of the classifier [33].

When rotating an image of for example a digit, the digit will remain. However, for time series it is not possible to easily tell if the data remains or is completely changed when such a transformation is applied [34]. What can be used for time series data is window slicing. Window slicing is when the data sample is split into two subsamples and each subsample is treated independent of the other. Another advantage with window slicing is that it makes sure all the data is of the same length [35].

3
Related work

This chapter contains approaches from related work and presents the methods used as well as the results that were achieved. The related work composed the foundation of the investigation of this thesis, as well as for the choice of technique for the implementation.

3.1 Signal processing

Below, results that have been achieved, as well as advantages and disadvantages of different approaches for signal processing of time series for usage in classification, are presented.

3.1.1 Fourier Transform

In the context of computer vision, faster classification of images in neural networks has proven to be possible by using FT as a preprocessing step on the images used as the data [4], [5]. Moreover, it has been carried out without decreasing accuracy [5].

The great improvement in terms of speed is also emphasised in Mathieu et al.'s [36] and Rippel et al.'s respective work [37]. Mathieu et al.'s work is based on using convolutionals as products in the frequency domain as well as reusing feature maps multiple times. In addition, Rippel et al. demonstrated in their work the boost in efficiency when creating spectral representations in the frequency domain and claimed that using Fourier functions leads to faster convergence in the optimisation part [37].

Even though the results above were achieved when applying Fourier transforms on images and not on time series, it is still possible that a similar effect could be achieved for time series. This is since both cases make use of the Fourier transform, plot the result of it and treat the plots as images in a neural network.

3.1.2 Conversion into spectrogram

CNN is optimal for image recognition. In the article *Audio Classification Using CNN — An Experiment* [12] the author described an experiment that used speech data as input to a CNN. The experiment was carried out by first converting all the audio files into spectrograms and then training the network on those plots. The conversion was done by application of STFT. Thereafter, each resulting window was

plotted as a vertical line in the spectrogram. They trained two networks, one on the spectrograms and one on the plain audio files. The result had an accuracy of 97% for the spectrogram compared to an accuracy of 94% for the plain audio model. This is related to our work since the data to be classified consists of time series.

In addition, Salamon et al. claim in the paper *Unsupervised feature learning for urban sound classification* [38] that using raw audio signals as input to a classifier does not only result in a very high dimensionality, but also that similar sounds might not be seen as related. As a result, they suggest transforming the signal into a spectrogram before feeding it to a classifier.

As mentioned in Section 2.2.2, spectrograms can be used the same way as images. This led to the insight that examples from image classification and their respective accuracies gained can be studied and compared further. This knowledge is used to a high extent in the following sections.

3.2 Neural networks in machine learning

There are three different kinds of neural networks that are studied in this thesis. Achieved effectiveness and performance in related work is described in the sections below. These measurement values show the potential and usage of each network architecture in the context of time series as input and for the purpose of solving classification tasks.

3.2.1 Convolutional neural network

Several attempts at classifying different types of audio streams and sounds using CNN have been made throughout the years. A paper written by Piczak [39] evaluated the potential of using CNN for classifying environmental sound. The results obtained showed high performance, despite limited datasets and usage of simple data augmentation. The mean accuracy for the model which gave the highest accuracy was around 80% and data that had been converted into spectrograms had been used. One of the conclusions drawn by the author in the paper was that for the case of classifying sounds in the city, simple methods for data augmentation did not improve the accuracy a lot, but mainly just increased the training time.

Furthermore, similar accuracies as for Piczk's experiment were obtained by Salamon et al. [38]. They tried classifying urban sounds using unsupervised learning, but achieved an accuracy of no higher than around 73.6%. Before classification, the data was transformed into spectrograms.

In contrast to Piczak's conclusion about data augmentation for urban sounds, Salamon et al. got state-of-the-art accuracies when applying it for classifying environmental sounds in another experiment [40]. A mean accuracy of approximately 79% was obtained in that test.

Moreover, in *Classifying environmental sounds using image recognition networks* [41], V. Boddapati et al. classified environmental sounds by transforming the data into three different image representations; spectrograms, mel-frequency cepstral coefficients and cross recurrence plots. The classification was carried out using AlexNet and GoogLeNet which are two well-known CNNs for image recognition. Lastly, they implemented a CRNN. The best result was obtained from a combination of spectrograms and GoogLeNet, which gave an accuracy of 91%.

3.2.2 Recurrent neural network

Kouba et al. trained a RNN to classify radar targets into five classes from a sequence of radar signatures [42]. The radar signatures were described by the estimated target azimuth, the estimated target width and four noisy amplitude values. The network achieved an accuracy of 96% on a test set. However, the radar signatures were generated and not extracted from a real scenario.

In addition, Xu et al. used CNNs for extracting features from mel-filter banks (MFBs), spectrograms, and raw waveforms for the purpose of classifying audio [43]. The resulting image representations were then fed into a GRU. The resulting model was tested in *Detection and Classification of Acoustic Scenes and Events 2016 (DCASE 2016)* challenge with great results; the equal error rate, or EER for short, obtained for the development dataset was 0.11 when not using spatial features. When using spatial features they achieved an EER value of 0.10 and a comparable result for raw waveforms. For the evaluation dataset they gained 0.12 as the EER value, which can be compared to the previously best value 0.15.

3.2.3 Convolutional recurrent neural network

Many human-computer interfaces such as Google Assistant and Apple's Siri usually run in the cloud due to the heavy computation required for speech recognition [44]. In *Aneural attention model for speech command recognition* [44], Coimbra de Andrade et al. introduced a lightweight *convolutional recurrent neural network* (CRNN) with attention for the purpose of processing simple commands, such as "stop" and "go", locally. The input to their model was 1s long raw WAV-files, from which spectrograms were created using DFT. Their resulting attention-CRNN model had an accuracy of 94.5% for one test and 93.9% for another, which were significantly better than the ones compared against.

Moreover, a CRNN was used for solving the task of identifying emergency sounds and detecting the time at which it was identified as part of a competition. The CRNN and its results was presented in a paper written by Lim et al. [45]. When creating the CRNN, a one-dimensional CNN and a RNN with LSTM were used. On the development set Lim et al. achieved an error rate of 0.07 and a F1-score of 96.26. For the evaluation dataset they got almost equally good results. 0.13 and 93.1 for the error rate and F1-score respectively were obtained.

3.3 Handling imbalanced datasets

As mentioned in Section 2.4, a balanced dataset is preferred in order to create a robust classifier. A related project to this one was done in Joakim Strandberg's master thesis [46]. His problem was to classify radar echos into two different classes using different machine learning approaches and by studying movement patterns of tracks already created from the data. The dataset given was quite small and imbalanced, where one class had 100 samples and the other 30 samples. However, the data samples, which were in the time domain, were fairly long. Therefore, in order to increase the size of the dataset and make it more balanced, samples were split into several non-overlapping subsamples.

Moreover, sampling as well as cost-sensitive learning have been presented as potential solutions to imbalanced datasets for classification problems [47], which show possible areas of usage for both techniques. Which of the two that performs best is, however, not clear according to Weiss et al. [32]. They claim that they are fairly equal, but that cost-sensitive learning performs a little better on larger datasets (in their experiments 10000 samples were used), whilst oversampling is better to use when having small datasets, even though oversampling increases the risk of overfitting.

In addition, data augmentation has been tried in several experiments for handling imbalanced datasets. However, the impact it has on the accuracy relative to the training time varies a lot, as can be seen in Section 3.2.1.

4
Methods

This thesis aimed to find a combination of useful signal processing steps and a classifier for making fast and accurate predictions on radar echoes. This chapter describes the choices made for the implementation part, with a foundation in theory and related work, see chapter 2 and 3, as well as studies about possible different techniques and approaches. It also describes the implementation itself and parameter choices.

In general, airborne vehicles may include several different types of objects, including different types of air planes, helicopters and unmanned aerial vehicles (UAV). In this thesis, the focus is only on jets and helicopters due to the time constraints. For the same reason, missiles are not classified or handled either. Including missiles, and all different types of objects regarded as airborne vehicles when classifying radar echoes can, however, be an alternative in future work.

The data used was three dimensional and the three dimensions are bearing (horizontal angle), range bins (distance interval) and lobes (height) seen from the radar's perspective. In this project, only one lobe has been used – the one with the target in. This gives a two dimensional matrix where each value corresponds to the amplitude and phase of the returned echo from the pulse. A high amplitude indicates that there is a target there. The bearing axis can in some way be seen as a time axis, since the radar echoes are collected as the radar is rotating and sending out pulses. Thereby, it is possible to form range doppler plots and spectrograms.

In addition, the data used was already prepared to some extent for this thesis. The data had been marked using the ADS-B signal, which is described further in section 2.1. As a result, all the training data was labelled with its correct class.

In the implementation part of the project, Python and its standard libraries, such as sklearn and Keras (included under Tensorflow), were used.

4.1 Signal processing

Since neural networks perform best, in terms of classification accuracy and efficiency when training on images, range doppler plots and spectrograms were chosen as input data to the classifiers tested in this thesis. The spectrograms, as well as the range doppler plots, were created from radar echoes. A more thorough motivation as well as a description of how this was done is given below.

4.1.1 Choice of technique

As Salamon et al. explained in their paper *Unsupervised feature learning for urban sound classification* [38], converting the data from the time domain to the frequency domain is ideal. Many different experiments and research about classifying audio have been made and almost all of them share the insight that converting the signals into spectrograms gives a better result (see Section 3.1.2). However, creating spectrograms takes a lot of data and in this project there was not quite enough. As a result, spectrograms were produced in order to see some of their potential, even though their full capacity could not be reached.

Moreover, due to the shortage of pulses for each sample, colormaps showing the range doppler plots were also created. Range doppler plots were tested since they demand less data compared to spectrograms to be good. One drawback with range doppler plots compared to spectrograms is that they are not as detailed and therefore might result in worse classifications.

4.1.2 Implementation

To be able to generate the range doppler plots, the data was first filtered with a MTI-filter (Moving Target Indication-filter). The filter separates moving from non-moving targets, like the ground, in radar echoes in order to then be able to remove the non-moving ones. Thereafter, FFT was applied on the bearing axis of the data. The FFT function used came from a Python standard library named SciPy. The result from the FFT was plotted as range doppler plots and saved as images. Examples of such plots, generated from the dataset, can be seen in Figure 2.2.

To generate spectrograms the range axis is replaced with a time axis and therefore only one range bin can be used. Since it was unknown in which range bin the target was in, all the bins were summed together. An MTI-filter was also used before spectrograms were created from the data. After filtering, SciPy's spectrogram function was applied, together with adjustment of the arguments to the function. The different settings that were tested were oversampling the data, how large window to apply the FFT on and what type of window to apply. Oversampling resulted in no difference, whilst varying the window size gave different spectrograms, but more or less the same result when classifying, regardless of the window size. Changing the window type, however, gave varying results and the best was obtained when using 'hamming'. The result from the spectrogram function was plotted and saved as images. Examples of created spectrograms can be seen in Figure 2.3.

4.2 Preprocessing of the dataset

The dataset consisted of raw video data, or radar echoes, and showed discovered jets and helicopters. It also included ADS-B signals, as mentioned before. All of the information contained in the signal was not of use for the thesis statement in this project and as a result it had to be filtered. In addition, all generated images

were normalized by first rescaling the images so that all of them had mean 0 and variance 1. Thereafter, all images were resized to the same size. This was done to be able to classify them.

On top of this, the dataset was small and imbalanced and included many more jets than helicopters, which had to be compensated for. The technique used to make the dataset more balanced is described in Section 4.2.2.

4.2.1 Handling noisy data

Noisy data is in this thesis interpreted as data without any targets in it, or with unidentifiable targets in it, such as echoes from the ground or other surroundings. In this dataset there was a lot of noisy data, which does not make the dataset optimal to train a network on. This is since the network will classify the noise as for example a jet when the noise actually should be classified as an unidentified object. An example of a noisy range doppler plot and a spectrogram is given in Figure 4.1 below.

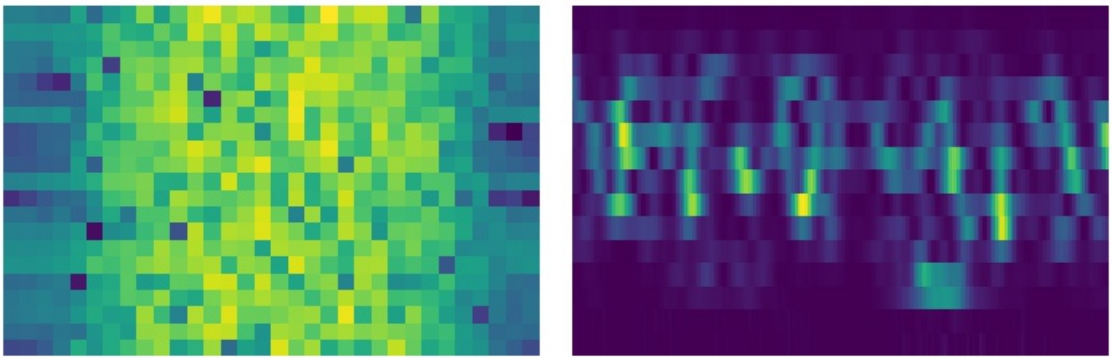

Figure 4.1: To the left is an example of a noisy range doppler plot and to the right a noisy spectrogram, both generated from the dataset used in this thesis. The velocity is represented on the x-axis and range on the y-axis of the range doppler plot. For the spectrogram, the x-axis shows the time dimension and y-axis the frequencies.

In order to handle this, the k-means clustering algorithm, which is described in Section 2.3.2 was used. Plots were formed from all the data and then fed to the algorithm. Given that k-means clustering is an unsupervised machine learning algorithm, it will only cluster according to the data and not the label. Hence, the data was put into k clusters, where k was set to 2. Only one class was fed to the algorithm at a time and the algorithm split the data into two categories; 'good data' and 'noisy data', independently of the labels of the data, by studying the patterns and structures of the images. Some of the noise data was put back to the dataset but with the label 'noise' instead of its original label. This implicated that the resulting dataset consisted of three classes; jet, helicopter, and noise.

4.2.2 Handling imbalanced data

The selected technique for handling the imbalanced dataset was cost-sensitive learning, or in other words using a cost function to increase the number of predictions of the minority class helicopters. There was approximately 32000 recorded jet samples in the original dataset, but only 3600 helicopter samples and after filtering only around 7200 jet and 1300 helicopter samples remained. The reason for why cost-sensitive learning was selected was that it is a common way of handling imbalanced datasets and, as the literature study shows (see Section 3.3) is useful for classification problems.

Furthermore, data augmentation has not been used in this project even though it could be useful for an imbalanced dataset. This is because plots of spectrogram and range doppler are not regular images. Regular images are invariant under the transformation of mirroring, translation and rotation, which is not the case for the plots in this project. Since a range doppler plot or spectrogram is not invariant in this way, these kinds of transformations would create a plot with different information.

Moreover, since the dataset was relatively small, there was a high risk of overfitting the model. Thus, sampling was not used to produce a larger dataset as it most likely would cause even more overfitting of the model.

4.3 Creating the neural networks

Several different kinds of neural networks were implemented and evaluated during the project. Which ones as well as a motivation of hyperparameter choices made are provided in the following sections. All models were tuned by testing.

4.3.1 Choice of classifier

From the results of experiments on data structures similar to the one in this project, presented in Section 3.2.1, 3.2.2 and 3.2.3, the conclusion that RNN or CRNN will most likely perform the best can be drawn. In this thesis it was decided to test RNN and CRNN, but also CNN.

The choice of using RNN and CRNN was due to two reasons. Firstly, the high accuracies obtained in related work showed great potential for RNN. Secondly, sequences of data (in this case sequences of images) can be fed to RNNs, which might improve accuracy. The different versions of RNNs that were implemented and compared were LSTM and GRU (see Section 3.2.2) as well as CRNN using LSTM and CRNN using GRU (see Section 3.2.3). In addition, a CNN was implemented despite the poor performance results in related work. The reason for why it was still implemented was that CNN is outstanding in image classification.

4.3.2 Implementation

The selected loss function for all tested models was categorical cross-entropy since it can be used for classifying multiple classes. There are two versions of the cross-entropy function; sparse and non-sparse and the difference between them is the form of the input. Sparse takes only an integer, which symbolises the class of the sample, as input, whilst non-sparse takes an array where the class of the sample is marked with 1 and the other values are set to zero. [48]

Moreover, the optimizer was the same for all models. The selected optimizer was Adam. The optimizers Adadelta and RMSprop were also tested, but Adam performed best and was therefore used for all models. Adam is an update to the RMSProp optimizer and also includes momentum [49]. The standard settings with a learning rate of 0.001 were used in the implementation.

Also, in the final layer of each model, the activation function softmax was used. Softmax was used since it makes sure the final probabilities in the output layers sum to one, which otherwise might not be the case when classifying multiple classes and having one output neuron for each class [19].

4.4 Evaluation

Several methods were used when evaluating the implementation and the resulting classifier. The classification error achieved when passing the test data through the network was studied. If an accuracy of approximately 90% or more was obtained, the network was seen as good.

In addition, the optimal classification rate for the network was to be able to classify around 1000 samples per second on a computer with typical laptop performance. This was not a strict demand, though. The running time for testing the model was also measured for the different models implemented.

Moreover, a confusion matrix was used for evaluating the classifiers. A confusion matrix measures performance and effectiveness of a classifier with an input of two or more classes. The matrix shows four different cases of a prediction. Either the classification is predicted to be positive and it is (referred to as true positive) or it is not (referred to as false positive), but the classification could also be predicted to be negative and that could also be either true (referred to as true negative) or false (referred to as false negative). [50] This is illustrated in Figure 4.2 below.

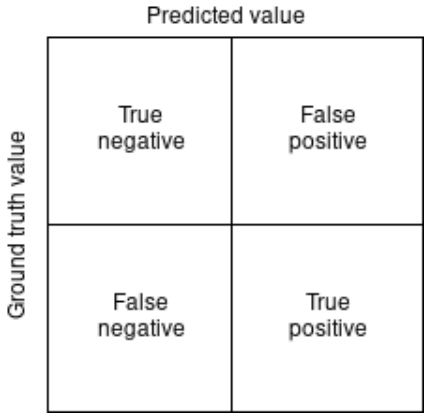

Figure 4.2: A confusion matrix used to evaluate performance for a model.

There are five different calculations that can be carried out from a confusion matrix; the recall, precision, accuracy, error rate, and F-score, which are calculated according to equation 4.1, 4.2, 4.3, 4.4 (and its rewritten form in equation 4.5) and 4.6. In the equations, true positive is denoted as TP, false positive as FP, false negative as FN and true negative as TN. [51, 50] The best possible value for each of them, apart from the error rate for which it is the other way around, is 1 whilst the worst is 0 [52, 53, 54, 55].

$$recall = \frac{TP}{TP \ + \ FN} \tag{4.1}$$

Recall is interpreted as the number of samples of the positive class that were predicted correctly. [50] For example, in the case of helicopter and jet, helicopter could be seen as the positive class whilst jet as the negative class. Recall is in that case the fraction of all correctly predicted helicopters over all helicopters in the dataset.

$$precision = \frac{TP}{TP \ + \ FP} \tag{4.2}$$

Precision indicates the number of samples that were correctly predicted to be positive. [50] Therefore, in the helicopter versus jet example, it would be the fraction of all correctly predicted helicopters over all samples that were predicted as helicopters.

$$accuracy = \frac{TP \ + \ TN}{Total \ number \ of \ samples} \tag{4.3}$$

Accuracy is simply how many samples that were correctly predicted, independently of class, out of the total number of classes and should ideally have a value that is as high as possible (maximum value is 1). [50]

$$error \ rate = \frac{FP \ + \ FN}{Total \ number \ of \ samples} \tag{4.4}$$

Error rate is the same as the classification error and shows the frequency of wrong predictions the model has done. The error rate is closely related to the accuracy and can therefore also be written in the following way: [51]

$$error \ rate = 1 - accuracy \tag{4.5}$$

$$F1\text{-}score = \frac{2 * recall * precision}{recall \ + \ precision} \tag{4.6}$$

If precision is low whilst recall is high or precision is high when recall is low, it will be hard to evaluate the model. Therefore, F1-score is used. F1-score makes use of harmonic mean instead of arithmetic mean and in that way penalises extreme values, creating a comparable value. [50]

5
Results

This chapter presents the performance results of each of the implemented and tested models. Confusion matrices, accuracy and loss diagrams, as well as a comparison of the final accuracies and performance metrics calculated from the confusion matrices is provided. The results are analysed and discussed with regards to usability, reliability and relevance in Section 6.1.1.

5.1 Performance results with filtering

This section presents the results for a CNN model using filtered data – that is, data where either all or most of the noise has been removed using the unsupervised algorithm k-means clustering. For the case where some of the bad data is kept, the noisy data has then been marked with a noise label. The results for the models using a filtered dataset can be compared with the ones displayed in Section 5.2.

5.1.1 CNN

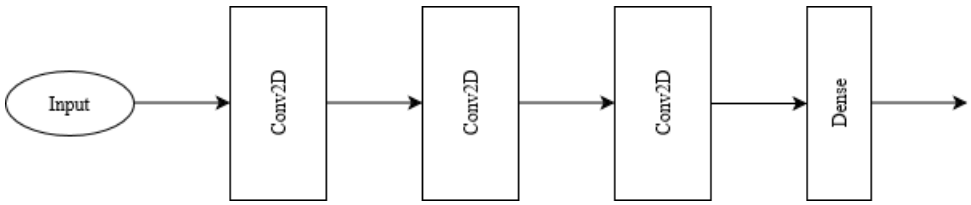

Figure 5.1: The figure visualises the layers used for the CNN model.

Figure 5.1 shows the different layers of the CNN model used. Before every Conv2D layer there is a ZeroPadding2D layer and after a BatchNormalization, a MaxPooling2D and a Dropout layer. For the Conv2D the number of filters was set to 20, the kernel size to 5 and the activation function was ReLU. For the Dense layer the activation function was Softmax. The dropout was set to 0.3 and the pooling size to 2. The input to the model was one image at a time with size $64x64$ and 3 channels with a batch size of 30.

5.1.1.1 CNN using filtered spectrograms with noise class

The CNN architecture presented in 5.1.1 was trained on the filtered spectrogram dataset, which made use of three classes; jet, helicopter, and noise. Table 5.1 shows a

confusion matrix of how the model classified the test samples of the dataset. Figure 5.2 describes the accuracy and loss for each epoch during the training of the model. It can be seen from the diagrams that the model is possibly overfitting.

Confusion matrix			
	Predict helicopter	Predict jet	Predict noise
Actual helicopter	257	152	8
Actual jet	619	1536	46
Actual noise	3	14	867

Table 5.1: The table shows that the model is good at recognising noise as well as jets, even if it classifies many jets as helicopters. The model does not do that well on classifying helicopters since the majority of the predicted helicopters are actually jets.

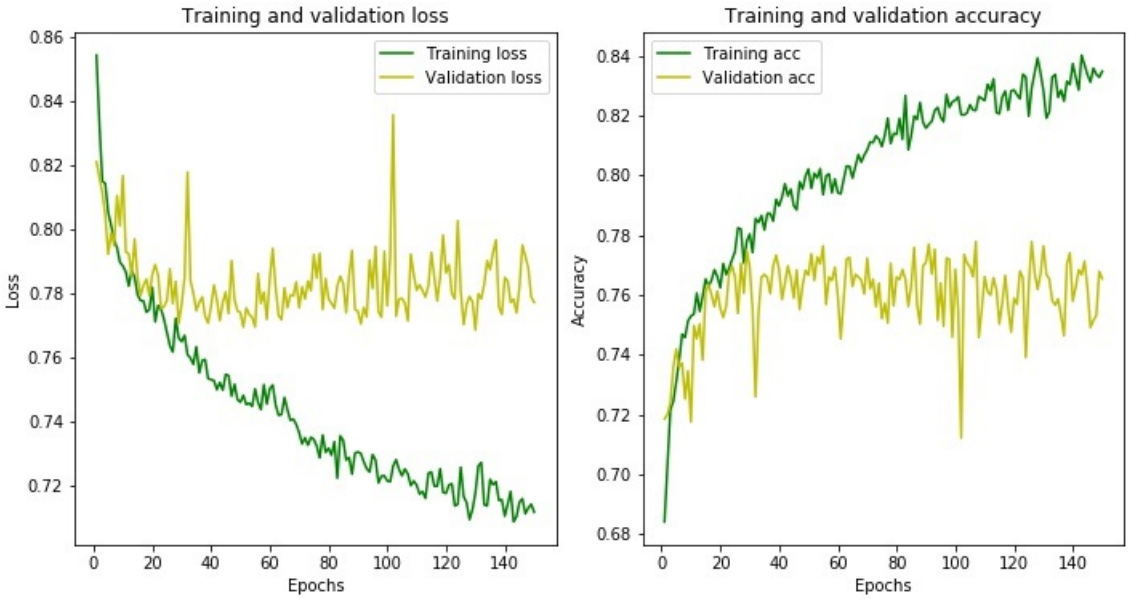

Figure 5.2: Diagrams showing how the loss and accuracy changes for every epoch during training and validation for a CNN model fed with filtered spectrograms, resulting in a quite large difference between training and validation.

5.1.1.2 CNN using filtered spectrograms without noise class

The CNN architecture presented in 5.1.1 was trained on the filtered spectrogram dataset, which made use of two classes; jet and helicopter. Table 5.2 shows a confusion matrix of how the model classified the test samples of the dataset. Figure 5.3 describes the accuracy and loss for each epoch during the training of the model. It can be seen from the diagrams that the learning is very unstable and the model tends to overfitting.

Confusion matrix		
	Predict helicopter	Predict jet
Actual helicopter	152	85
Actual jet	215	1199

Table 5.2: The table shows that the model is good at recognise jets. However, one third of the helicopters are classified as jet and most of the predicted helicopters are actually jets.

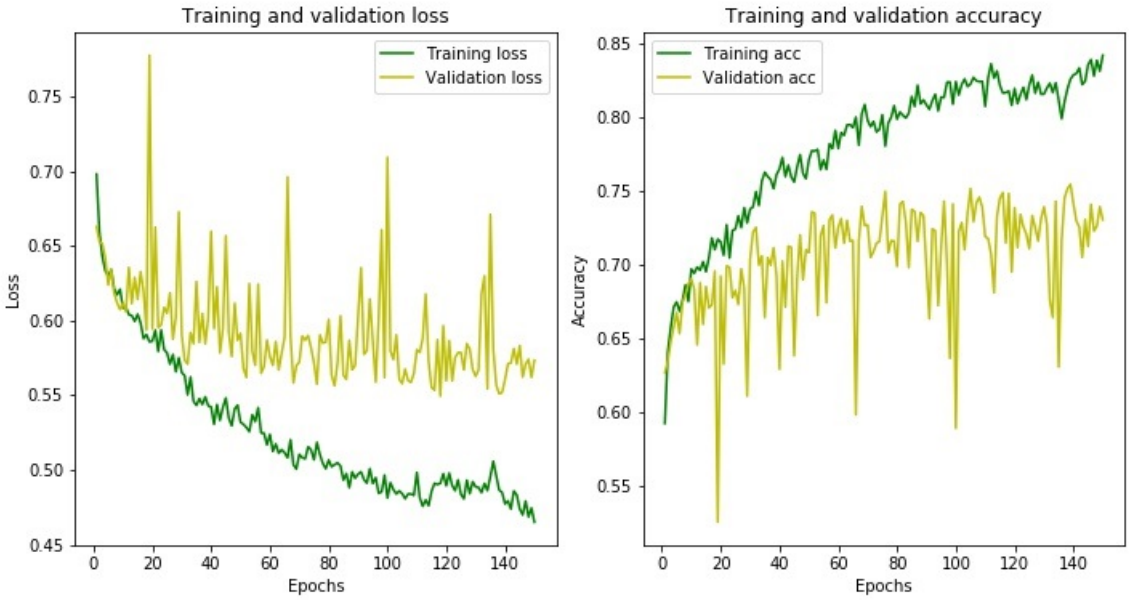

Figure 5.3: Diagrams showing how the loss and accuracy changes for ever epoch during training and validation for a CNN model given filtered spectrograms without a noise class as input.

5.1.1.3 CNN using filtered range doppler plots with a noise class

The network architecture for a CNN presented in 5.1.1 was also trained using range doppler plots as input data. Table 5.3 provides the confusion matrix, whilst Figure 5.4 describes the accuracy and loss diagrams of the model. From the figure, it can be seen that the model seems to get a good performance result, although there is room for improvement.

Confusion matrix			
	Predict helicopter	Predict jet	Predict noise
Actual helicopter	168	84	5
Actual jet	159	1269	15
Actual noise	4	11	1107

Table 5.3: The confusion matrix shows that the model using filtered range doppler plots and a third class noise is great at classifying noise (see third row, last column), but also at classifying jet (see middle value). On top of this, it is decent at correctly predicting helicopters (see first row, first column).

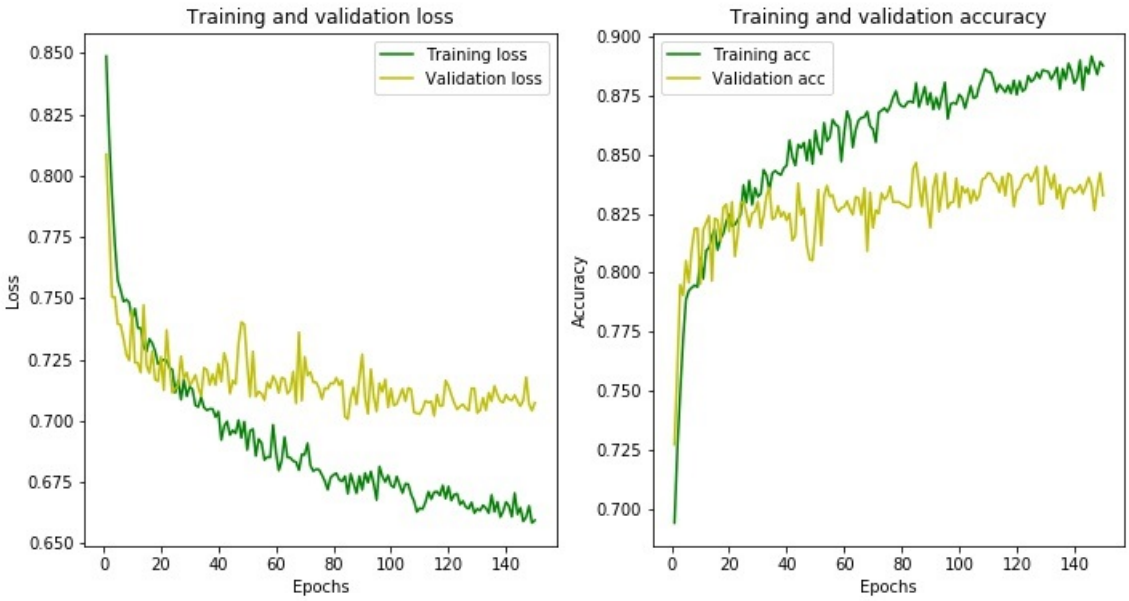

Figure 5.4: The diagrams show the loss and accuracy for each epoch for training and validation respectively for a CNN model using filtered range doppler plots as input data.

5.1.1.4 CNN using filtered range doppler plots without noise class

The CNN architecture described in 5.1.1 was also trained on filtered range doppler plots without a noise class. Table 5.2 shows a confusion matrix of how the model classified the test samples of the dataset and Figure 5.3 shows the accuracy and loss for each epoch during training of the model.

Confusion matrix		
	Predict helicopter	Predict jet
Actual helicopter	187	70
Actual jet	260	1183

Table 5.4: The confusion matrix shows that the model using filtered range doppler plots without the noise class is good at classifying jet. However, it predicts more jets than helicopters as helicopter.

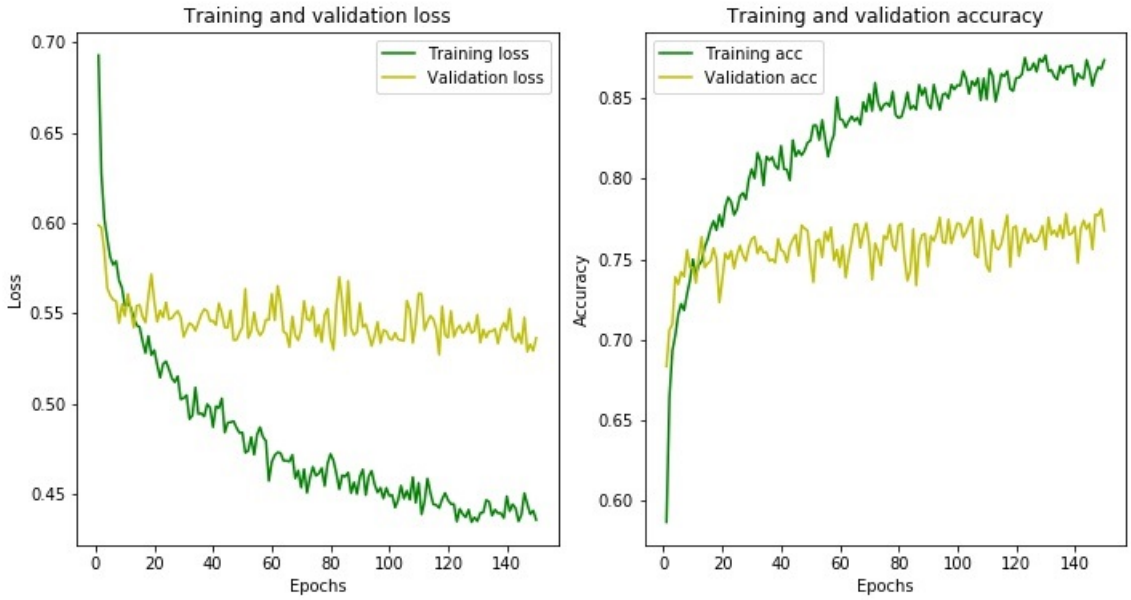

Figure 5.5: The diagrams show the loss and accuracy for each epoch for training and validation of a CNN using filtered range doppler plots without a noise class as input.

5.2 Performance results without filtering

Models for CNN, GRU, LSTM, CRNN with LSTM and CRNN with GRU were tested on unfiltered datasets for both range doppler plots and spectrograms. However, the performance was very poor for GRU, LSTM and CRNN with GRU and therefore the results for these three models are not included in this report. The reason for the bad performance is discussed in Section 6.1.

The results described in this section ought to be compared to the ones in Section 5.1.

5.2.1 CNN

This section describes the results obtained for a CNN with spectrograms and range doppler plots as input data, both of which use the network architecture illustrated

in Section 5.1.1.

5.2.1.1 CNN using unfiltered spectrograms

The CNN architecture presented in 5.1.1 was trained on the unfiltered spectrogram dataset. The generated classifier's performance can be studied in Table 5.5, which shows a confusion matrix of how test samples of the dataset were classified. It is important to recall, though, that the dataset includes lots of noisy data which the model will be forced to classify as either jet or helicopter, leaving a confusion matrix which might not entirely show how well the classifier is doing.

Furthermore, Figure 5.6 describes the overall performance for all classes considering loss values calculated using the activation function and accuracy. The diagrams evince that there is a quite large gap between training and validation and that it potentially will increase for coming epochs as well. This might indicate overfitting of the model.

Confusion matrix		
	Predict helicopter	Predict jet
Actual helicopter	473	238
Actual jet	2126	4221

Table 5.5: It can be seen from the table that the model predicts a lot of the jets to be helicopters and many of the helicopters to be jets.

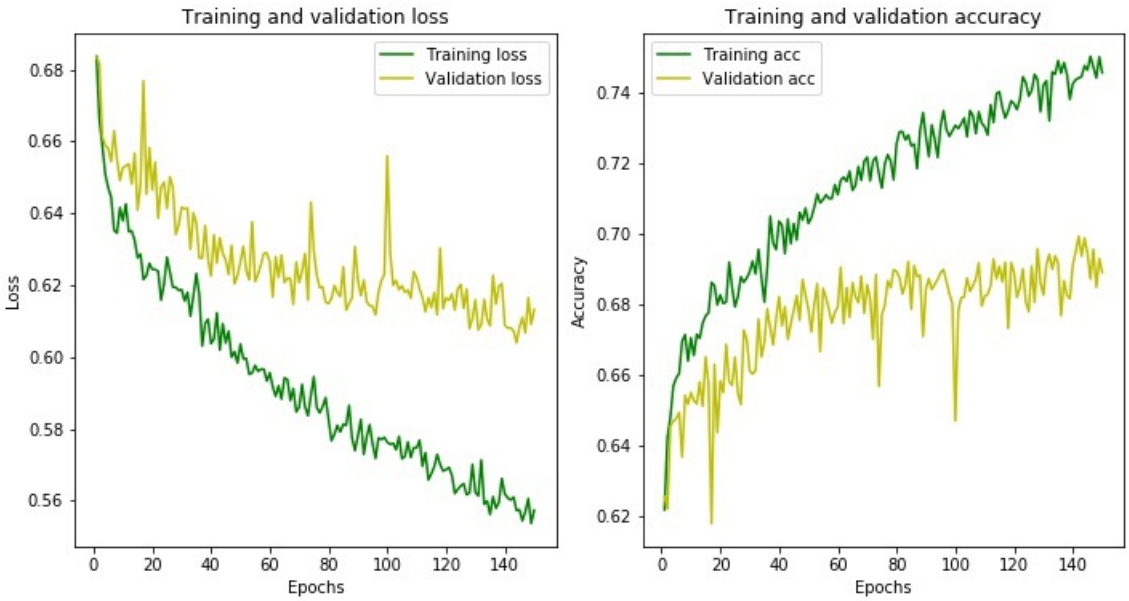

Figure 5.6: The leftmost diagram compares the training and validation loss, whilst the rightmost compares training and validation accuracy.

5.2.1.2 CNN using unfiltered range doppler plots

The CNN architecture (see Section 5.1.1) was not only trained on unfiltered spectrograms, it was also trained on unfiltered range doppler plots. The confusion matrix corresponding to the model created is provided in Table 5.6 and the accuracy and loss diagrams of the model in Figure 5.7 below. Again, however, it is important to remember that noise is also classified as jet or helicopter for this model.

The table and figure shows that the model is not doing well on unfiltered range doppler plots because there is too much noise.

Confusion matrix		
	Predict helicopter	Predict jet
Actual helicopter	364	352
Actual jet	1319	5038

Table 5.6: For the case of CNN with range doppler plots as input data, the table shows that the model is good at classifying jets (see lower right) but quite bad at classifying helicopters since only half of the helicopters are predicted correctly (see upper left and upper right).

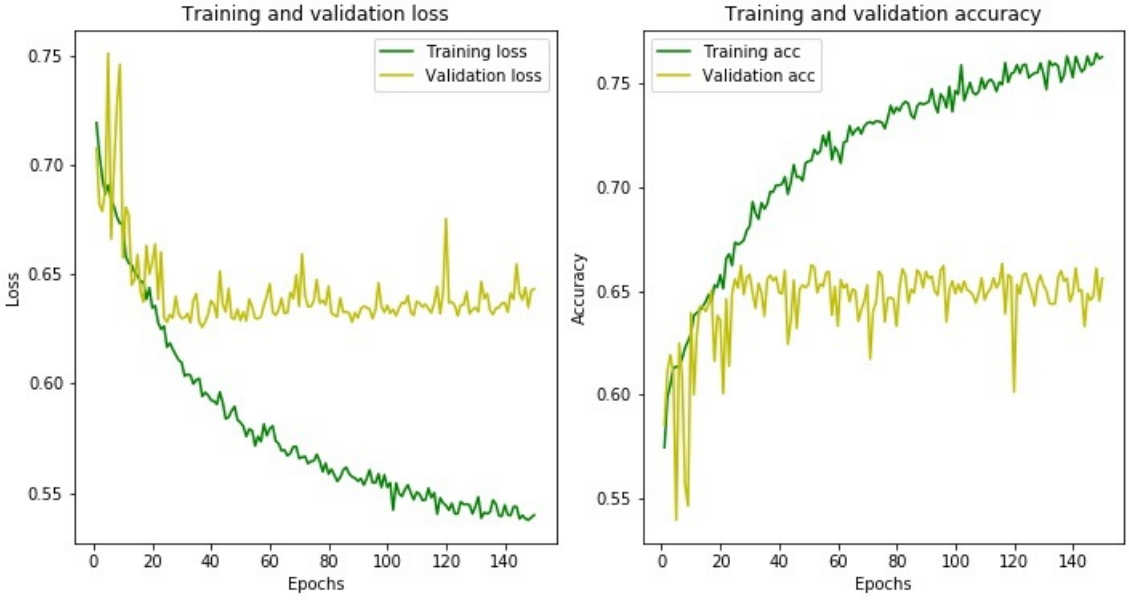

Figure 5.7: The diagrams visualise how the loss and accuracy adjusts for every epoch when training and validating for a CNN model trained on unfiltered range doppler plots.

5.2.2 CRNN with LSTM

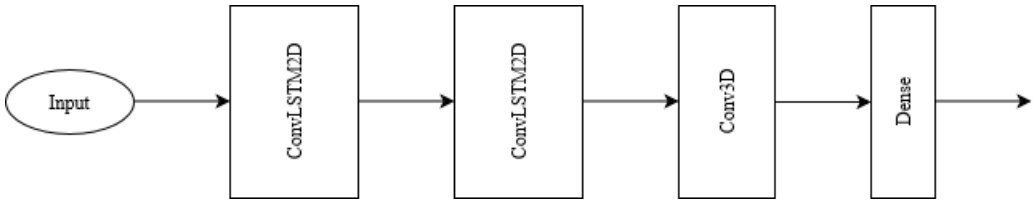

Figure 5.8: The figure illustrates the layers used for the CRNN with LSTM model.

Figure 5.8 shows the different layers of the CRNN with LSTM model used. After every ConvLSTM2D layer there is a BatchNormalization and a Dropout layer. The number of filters used was 32 followed by 16 and the kernel sizes were 3 at all times. The activation function was *tanh* for both ConvLSTM2D layers. The Conv3D layer was also followed by a BatchNormalization and a Dropout layer. The dropout was set to 0.4. The Dense layer made use of Softmax as its activation function. The input to the model was a sequence of images, each with a size of $64x64$ and 3 channels together with a batch size of 20.

5.2.2.1 CRNN with LSTM using unfiltered spectrograms

The CRNN with LSTM architecture presented in 5.2.2 was trained using unfiltered spectrograms, i.e. where noisy data is included and therefore classified as either jet or helicopter. The performance of the generated model can be seen in Table 5.7 and Figure 5.9. The table describes how test samples from the dataset was classified and the figure the model's overall performance considering loss and accuracy.

The diagrams in Figure 5.9 show that the model has great difficulties with learning the difference between helicopters and jets and as a result mainly just guesses at jet, as can be seen in the confusion matrix in Table 5.7 above.

Confusion matrix		
	Predict helicopter	Predict jet
Actual helicopter	36	99
Actual jet	292	913

Table 5.7: For the case of CRNN with LSTM using spectrograms as input data, the table shows that the model is quite good at classifying jets (see lower right), but terrible at classifying helicopters (see upper left).

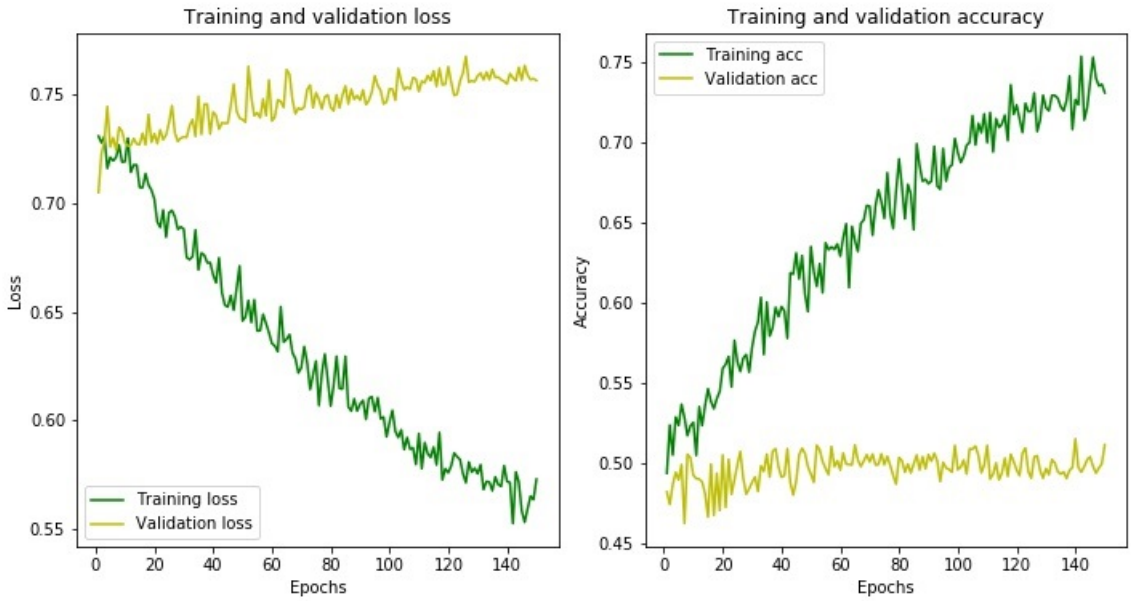

Figure 5.9: Diagrams showing how the loss and accuracy changes for every epoch when training and validating on a CRNN with LSTM model.

5.2.2.2 CRNN with LSTM using unfiltered range doppler plots

The network structure given in Section 5.2.2 was also fed with unfiltered range doppler plots, resulting in another CRNN with LSTM model. The metric values and performance diagrams for this model are given in Table 5.8 and Figure 5.10 respectively.

As for the case with unfiltered spectrograms as input to a CRNN with LSTM model, the diagrams for a model using unfiltered range doppler plots instead of spectrograms, show that the model is not learning much. This can be seen in both the diagram to the left which provides the changes in loss for each epoch and the diagram to the right with the changes in accuracy for every epoch when training and validating.

Confusion matrix		
	Predict helicopter	Predict jet
Actual helicopter	68	69
Actual jet	610	593

Table 5.8: For the case of CRNN with LSTM using range doppler plots as input data, the table shows that the model is very bad at classifying both jets (see lower right) and helicopters (see upper left).

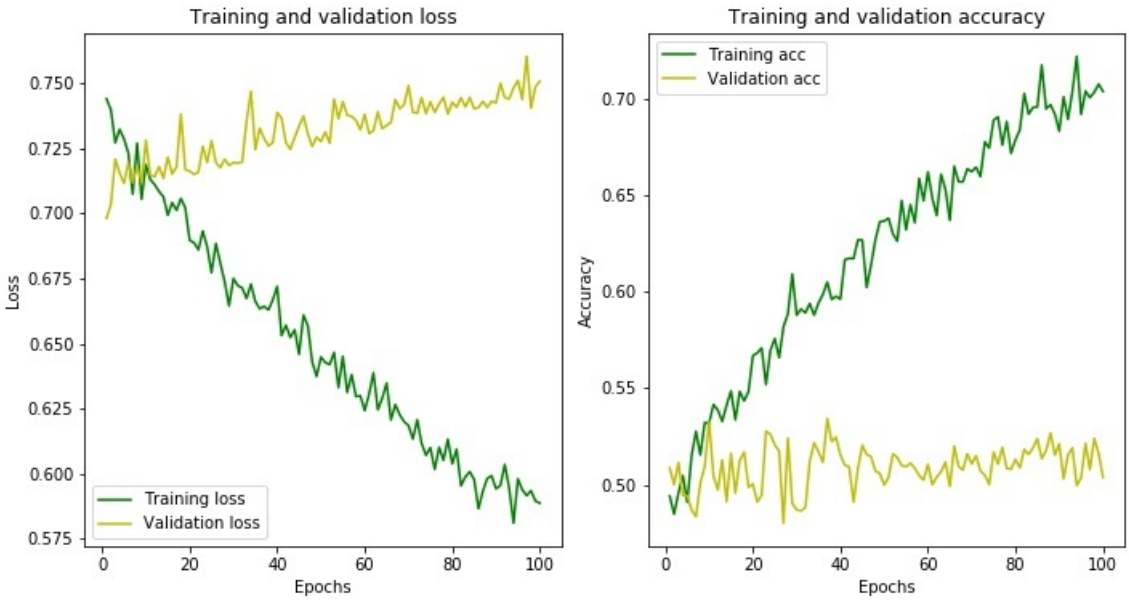

Figure 5.10: Performance results for a CRNN with LSTM model that is given unfiltered range doppler plots. The results shown are with respect to loss and accuracy for training and validation.

5.3 Comparison of the models

Diagrams describing changes in loss and accuracy for training and validation data respectively give a good idea of how well a model performs, but not the entire picture. It is possible that a model is only good at classifying some classes and terrible at others and the diagrams might not show this. Therefore, performance metrics such as recall and precision were calculated according to the formulas presented in Section 4.4. Remember that for all metrics below, apart from error rate, the optimal value is 1 and worst possible value is 0. For error rate it is the other way around.

Table 5.9 and 5.10 below show that the models finds it most difficult to classify helicopters, but is good at classifying jets and noise for the CNN models. It can also be concluded that filtered data works best. Best performance was achieved for the dataset with filtered range doppler plots and a noise class. This can be seen when comparing the F1-scores for the respective classes and confusion matrices, as well as when comparing the overall test accuracies which are also slightly higher when compared to not using the noise class. The recall and precision values indicate the same.

Performance metrics (spectrograms)						
Model	Class	Recall	Precision	Test acc.	Error rate	F1-score
CNN [1]	jet/heli/noise	0.70/0.62/0.98	0.90/0.29/0.94	0.76	0.24	0.79/0.40/0.96
CNN [2]	jet/helicopter	0.85/0.64	0.93/0.41	0.82	0.18	0.89/0.50
CNN [3]	jet/helicopter	0.67/0.67	0.95/0.18	0.67	0.33	0.78/0.29
CRNN LSTM [3]	jet/helicopter	0.76/0.27	0.90/0.11	0.71	0.29	0.82/0.16

Table 5.9: The table shows all metric values obtained for the CNN with and without filtered data and with and without using a noise class. It also shows metric values for a CRNN with LSTM model which was fed with unfiltered data. All four models used spectrograms as input. Recall, precision and F1-score are given as per label values.

Performance metrics (range doppler plots)						
Model	Class	Recall	Precision	Test acc.	Error rate	F1-score
CNN [1]	jet/heli/noise	0.88/0.65/0.99	0.93/0.51/0.98	0.90	0.10	0.90/0.57/0.98
CNN [2]	jet/helicopter	0.82/0.73	0.94/0.42	0.81	0.19	0.88/0.53
CNN [3]	jet/helicopter	0.79/0.51	0.93/0.22	0.76	0.24	0.86/0.30
CRNN LSTM [3]	jet/helicopter	0.49/0.50	0.90/0.10	0.49	0.51	0.64/0.17

Table 5.10: The table presents all metric values obtained for the CNN model, which was given filtered data with a noise class, filtered data without a noise class and unfiltered data respectively. The table also provides metric values for the CRNN with LSTM model which was given unfiltered data. All four models were given range doppler plots as input. Recall, precision and F1-score are given as per label values.

[1] The data has been filtered using k-means clustering before the model was trained, but some noisy data has been kept.

[2] The data has been filtered using k-means clustering and all noisy data has been removed before the model was trained.

[3] The data has not been filtered before the model was trained.

6
Conclusion

In this chapter a summary and conclusion of the results is provided in addition to the next step to be done in the investigation. The chapter also presents a thorough analysis of the results obtained and how they stand in relation to the data being used.

6.1 Discussion

This section provides an analysis of obtained results and how they are affected by the data used. It also presents an analysis of the data itself and how different choices of how to handle it could have affected the result. Finally, an evaluation of the methods used to evaluate the performance of the models is made and whether they fulfilled their purposes and were good techniques for measurements.

6.1.1 Analysis of the data and the results

As explained in Section 2.3.3.1, features are extracted from the input data which for this work consisted of spectrogram and range doppler plots and were handled as images. Usually when handling images in machine learning, the features extracted can be quite easily discovered manually, like when having images of faces for instance. An example of a feature for that particular case would be a nose or eyes and where in the face the eyes are located in relation to the nose. For spectrogram and range doppler plots, on the other hand, identifying exactly what the neural network will look for and see as a feature is not that obvious. Hence, on one hand finding patterns a human cannot see is one of the benefits with applying machine learning techniques, but at the same time it also creates an uncertainty about the reliability of the model. Is the network looking for patterns it only believes exists, but which in fact does not exist?

Furthermore, the results presented in Chapter 5 show that using spectrograms as input data gave a worse result than range doppler plots and even seemed to cause overfitting of the model, despite spectrograms' great potential according to related work. This can be explained by the fact that quite few pulses were used when generating the data. When classifying sounds, which is similar to this problem, many pulses are sent and thereafter spectrograms are created with much higher time resolution than what was possible in this thesis. In addition, more data to train on is necessary to get good resolution on the spectrograms, but it would most likely also

improve the results when using range doppler plots as well. In conclusion, spectrograms can still be an option for this kind of classification problem if the number of pulses is greatly increased and more data is created, but with the current dataset they are not.

Moreover, using the spectrograms, but also the unfiltered range doppler plots as they are now, caused a dreadful result on the LSTM, the GRU and the CRNN with GRU models. The huge amount of noise in the data, and for spectrograms also in combination with low resolution, resulted in classification of the datasets using these models being an impossible task. It is therefore necessary to filter the data to remove most of the noisy data.

6.1.2 Imbalanced data

The dataset used was highly imbalanced, which could have made an impact on the obtained result in several different ways.

Firstly, the imbalance could potentially cause the model to be good at classifying jets, but not anything else since the majority of the samples were jets and thereby the network's purpose fails. This effect can be seen in the metrics values shown in Chapter 5. On the other hand, the dataset is more realistic with many more jets than helicopters. However, the imbalance causes the test accuracy to be higher than the validation accuracy for the models since class weights are not used when testing. This affects the reliability of the test accuracy and as a result, both the test and validation accuracy is pointed out when presenting the final results for this thesis.

Secondly, trying to fix the major problem with imbalance using standardized techniques for handling imbalanced datasets, such as cost-sensitive learning, might only cause overfitting. For this work, however, using cost-sensitive learning seems to have improved the result.

6.1.3 Noisy data

The dataset used was not only imbalanced, it was also very noisy (the interpretation of noisy data used for this thesis is given in Section 4.2.1). An attempt was therefore made to filter it using the unsupervised, i.e. without considering labels, algorithm k-means clustering. Filtering the data using this technique improved the results substantially. A third class called noise was also taken into account and added by re-labelling all data found in the cluster with noisy data.

Having a noise class has both advantages and disadvantages though. One disadvantage is that having a noise class might cause the network to classify jets and helicopters incorrectly as noise and in that way it might take longer time to find a sample that is classified as an actual vehicle. On the other hand, the model might

be more accurate if some samples are discarded as noise. This could result in a more correct picture of how well the model will perform in reality; that is, if the model knows some data is bad, will it be able to classify jets and helicopters correctly? Moreover, a noise class is needed in order to prevent the neural network from believing that a very noisy image corresponds to for instance a helicopter and thereby causing the neural network to not become a useful classifier.

Remark though that not all noisy data could be kept since a great part of the dataset was noisy and therefore keeping all noisy data would have resulted in an even more imbalanced dataset. However, removing all noisy data and therefore not having a noise class does not completely solve the problem with having noisy data either as k-means clustering will not result in two perfect clusters. There will always be some noise left in the 'good' cluster and true jet and helicopter samples in the 'bad' cluster. When studying the discarded and saved images respectively, it seemed like this was not a huge problem in our case though. Without noisy data, the neural network would also perform very poorly if used in reality.

To be able to have a noise class, some of the noisy data had to be added to the filtered dataset before training the model, but this results in another challenge; not adding too much or too little noisy data. Furthermore, it was unfortunately not possible to use RNNs on the filtered data as a consequence of too little data remaining after clustering and since there would be quite a lot of time passing between two 'good' samples showing the same jet or helicopter. Therefore, not enough sequences of images could be formed.

6.1.4 Evaluation methods

All models implemented were evaluated by calculating the loss and accuracy for each epoch, but also by calculating different metrics such as recall. The number of samples that could be tested per second were also computed to be able to compare with the 1000 samples per second minimum. However, using the number of samples per second as a metric for evaluating a model might not give a fair value. This is since using a computer with better performance will give a better value and selecting a more suitable computer for doing the classification is possible. Hence, due to the rates' irrelevance, they were excluded from this report.

The other ways of evaluating the models give a broader picture of how well the models are doing. It is crucial, though, to remember that the models were trained on highly imbalanced data in addition to very few samples and therefore a much better performance is likely to be achieved if there was more data available, especially helicopter samples, see Section 6.1.1.

6.2 Conclusion

To conclude, the CNN model which was given filtered range doppler plots with a noise class in addition to the jet and helicopter classes performed best seen to both

overall performance and metrics, as can be seen in Section 5.1.1.3. The final test accuracy obtained for that model was around 90% and the final validation accuracy was 83%. We believe CNN performed the best due to using images and since CNNs are outstanding at classifying images, but also due to the data being used (see Section 6.1.2 and 6.1.3), even though the potential for CRNN with LSTM, as well as for using spectrograms, remains (see Section 6.3).

This result shows that it is most likely possible to classify jets and helicopters using only raw video data and a neural network in a real-time scenario, even though it is not completely certain since the precision and F1-score is only just above 0.5 for helicopters. Therefore, some more work has be done, as described in Section 6.3 below, including training and testing on a much larger dataset. In addition, the data must be filtered and a computer with high performance must be used, as seen and explained in Section 5.10 and 6.1.4.

6.3 Future work

In this project, the only preprocessing of the dataset was creating range doppler images and spectrograms using FFT and normalizing the images, followed by filtering. However, another possible technique that transforms time domain data into the frequency domain is the Hartley Transformation which has computational benefits compared to FFT since it, given real values as input, outputs no imaginary values [56]. It would also be interesting to try other techniques for filtering the data in addition to k-means clustering, as well as training and evaluating the current models on more data, including more helicopters.

Furthermore, only CNNs, RNNs using LSTM or GRU and CRNNs using LSTM or GRU were carried out due to the time constraints of the project and the potential of these methods as seen in the related work, see Section 3.2.1, 3.2.2 and 3.2.3. It is possible though that other neural network architectures, not to mention transfer learning which is well studied [57], [58], [59], [60], [61], could perform equally well or even better than the ones implemented and tested in this project. Also, more fine-tuning of the models tested could improve their respective performance results.

Examples of extended fine-tuning could be adjusting the number of hidden layers, which layers to use, the parameters of the layers, as well as the layers' sizes. Adjusting parameters at the generation of the data, such as how many images to include in one sequence or the number of windows to apply FFT on, could also be potential fine-tuning.

Moreover, all different kinds of vehicles included in the term airborne vehicles as well as missiles and birds can be added to the network so that it is able to recognise everything moving in the air.

Lastly, and perhaps most important of all, the final model should be trained and tested on much more, and preferably less noisy, data using more pulses. It could

result in for instance CRNN with LSTM finding typical movement patterns for different vehicles since it is fed with sequences of data, but it would most likely also improve the performance of the currently best model. We believe that the CNN model using filtered data which includes a noise class, and possibly also the CRNN with LSTM model using unfiltered data, then could become very useful classifiers and be used in (almost) real-time.

Bibliography

[1] Creative Commons, "Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)." https://creativecommons.org/licenses/by-sa/4.0/.

[2] L. Cannon, "2011 Ocean City, MD Air Show." https://www.flickr.com/photos/9763931@N04/5906653857, 2011.

[3] G. W. Stimson, *Introduction to Airborne Radar*. Aerospace & Radar Systems, SciTech Pub., 1998.

[4] J. Li, S. You, A. Robles-Kelly, "A Frequency Domain Neural Network for Fast Image Super-resolution," in *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, July 2018.

[5] H. Pratt, B. Williams, F. Coenen, Y. Zheng, "FCNN: Fourier Convolutional Neural Networks," in *Machine Learning and Knowledge Discovery in Databases*, (Cham), pp. 786–798, Springer International Publishing, 2017.

[6] F. Franzen, and C. Yuan, "Visualizing image classification in Fourier domain," in *ESANN 2019 - Proceedings*, pp. 535–540, April 2019.

[7] K. Samuelson, E. Valovage, D. Hall, "Enhanced ADS-B research," in *2006 IEEE Aerospace Conference*, pp. 7 pp.–, March 2006.

[8] Mathworks, "Airplane Tracking Using ADS-B Signals." https://se.mathworks.com/help/comm/examples/airplane-tracking-using-ads-b-signals.html.

[9] G. Stimson, *Introduction To Airborne Radar.* Mendham, NJ, USA: SciTech Publishing, INC., 2 ed., 1998.

[10] K. Chaudhary, "Understanding Audio data, Fourier Transform, FFT and Spectrogram features for a Speech Recognition System." https://towardsdatascience.com/understanding-audio-data-fourier-transform-fft-spectrogram-and-speech-recognition-a4072d228520, 2020.

[11] D. Shulga, "Speech Classification Using Neural Networks: The Basics." https://towardsdatascience.com/speech-classification-using-neural-networks-the-basics-e5b08d6928b7, 2018.

[12] AI Graduate Admin, "Audio Classification Using CNN — An Experiment." https://medium.com/x8-the-ai-community/audio-classification-using-cnn-coding-example-f9cbd272269e, 2019.

[13] A. Pant, "Introduction to Machine Learning for Beginners." https://towardsdatascience.com/introduction-to-machine-learning-for-beginners-eed6024fdb08, 2019.

[14] R. Sathya, A. Abraham, "Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification," *International Journal of Advanced Research in Artificial Intelligence (IJARAI)*, vol. 2, no. 2, pp. 34–38, 2013.

[15] D. Soni, "Supervised vs. Unsupervised Learning." https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d, 2018.

[16] Mathworks, "Unsupervised Learning." https://se.mathworks.com/discovery/unsupervised-learning.html.

[17] Dr. Michael J. G., "Understanding K-means Clustering in Machine Learning." https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1, 2018.

[18] M. Jain, "The Basics of Neural Networks." https://medium.com/datadriveninvestor/the-basics-of-neural-networks-304364b712dc, 2019.

[19] L. Shukla, "Designing Your Neural Networks." https://towardsdatascience.com/designing-your-neural-networks-a5e4617027ed, 2019.

[20] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way." https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, 2018.

[21] C. Thomas, "An introduction to Convolutional Neural Networks." https://towardsdatascience.com/an-introduction-to-convolutional-neural-networks-eb0b60b58fd7, 2019.

[22] S. Kostadinov, "How Recurrent Neural Networks work." https://towardsdatascience.com/learn-how-recurrent-neural-networks-work-84e975feaaf7, 2017.

[23] TensorFlow, "The Keras functional API in TensorFlow." https://www.tensorflow.org/guide/keras/functional#models_with_multiple_inputs_and_outputs, 2020.

[24] B. Chandra and R. K. Sharma, "On improving recurrent neural network for image classification," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 1904–1907, May 2017.

[25] M. Nguyen, "Illustrated Guide to LSTM's and GRU's: A step by step explanation." https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21, 2018.

[26] D. Coimbra de Andrade, "Recognizing Speech Commands Using Recurrent Neural Networks with Attention." https://towardsdatascience.com/recognizing-speech-commands-using-recurrent-neural-networks-with-attention-c2b2ba17c837, 2018.

[27] E. Alese, "The curious case of the vanishing & exploding gradient." https://medium.com/learn-love-ai/the-curious-case-of-the-vanishing-exploding-gradient-bf58ec6822eb, 2018.

[28] N. Arbel, "Attention in RNNs."
https://medium.com/datadriveninvestor/attention-in-rnns-321fbcd64f05, 2019.

[29] G. Yibo, "Machine Learning Cheat Sheet — Model Evaluation and Validation."
https://towardsdatascience.com/machine-learning-cheat-sheet-model-evaluation-and-validation-b67565df6075, 2019.

[30] F. Chollet and others, "Regularizers - Keras Documentation."
https://keras.io/regularizers/, 2015.

[31] W. Badr, "Having an Imbalanced Dataset? Here Is How You Can Fix It.."
https://towardsdatascience.com/having-an-imbalanced-dataset-here-is-how-you-can-solve-it-1640568947eb, 2019.

[32] G. M. Weiss, K. McCarthy, B. Zabar, "Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs?," *Dmin*, vol. 7, no. 35-41, p. 24, 2007.

[33] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," 2017.

[34] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, "Data augmentation using synthetic data for time series classification with deep residual networks," 2018.

[35] Z. Cui, W. Chen, Y. Chen, "Multi-Scale Convolutional Neural Networks for Time Series Classification," 2016.

[36] M. Mathieu, M. Henaff, Y. LeCun, "Fast Training of Convolutional Networks through FFTs," 2013.

[37] O. Rippel, J. Snoek, R. P. Adams, "Spectral Representations for Convolutional Neural Networks," 2015.

[38] J. Salamon and J. P. Bello, "Unsupervised feature learning for urban sound classification," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 171–175, April 2015.

[39] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, Sep. 2015.

[40] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, pp. 279–283, March 2017.

[41] V. Boddapati, A. Petef, J. Rasmusson, L. Lundberg, "Classifying environmental sounds using image recognition networks," *Procedia Computer Science*, vol. 112, pp. 2048 – 2056, 2017.

[42] E. T. Kouba, S. K. Rogers, D. W. Ruck, K. W. Bauer Jr., "Recurrent neural networks for radar target identication," in *Applications of Artificial Neural Networks IV* (S. K. Rogers, ed.), vol. 1965, pp. 256 – 266, International Society for Optics and Photonics, SPIE, 1993.

[43] Y. Xu, Q. Kong, Q. Huang, W. Wang, M. D. Plumbley, "Convolutional gated recurrent neural network incorporating spatial features for audio tagging," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 3461–3466, May 2017.

[44] D. Coimbra de Andrade, S. Leo, M. Loesener Da Silva Viana, C. Bernkopf, "A neural attention model for speech command recognition," 2018.

[45] H. Lim, J. Park, K. Lee, Y. Han, "Rare Sound Event Detection Using 1D Convolutional Recurrent Neural Networks," in *Detection and Classification of Acoustic Scenes and Events 2017*, November 2017.

[46] J. Strandberg, "Time series classification for identification of radar targets," Master's thesis, Chalmers University of Technology, 2015.

[47] Z. Wang, "Practical tips for class imbalance in binary classification." https://towardsdatascience.com/practical-tips-for-class-imbalance-in-binary-classification-6ee29bcdb8a7, 2018.

[48] F. Chollet and others, "Losses - Keras Documentation." https://keras.io/losses/, 2015.

[49] S. Lau, "Learning Rate Schedules and Adaptive Learning Rate Methods for Deep Learning." https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1, 2017.

[50] S. Narkhede, "Understanding Confusion Matrix." https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62, 2018.

[51] S. Rawat, "Is accuracy EVERYTHING?." https://towardsdatascience.com/is-accuracy-everything-96da9afd540d, 2019.

[52] scikit-learn developers, "sklearn.metrics.recall_score." https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html, 2007-2019.

[53] scikit-learn developers, "sklearn.metrics.precision_score." https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html, 2007-2019.

[54] scikit-learn developers, "sklearn.metrics.accuracy_score." https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html, 2007-2019.

[55] scikit-learn developers, "sklearn.metrics.f1_score." https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html, 2007-2019.

[56] Wolfram Research, Inc., "Hartley transform." http://mathworld.wolfram.com/HartleyTransform.html.

[57] J. Brownlee, "A Gentle Introduction to Transfer Learning for Deep Learning." https://machinelearningmastery.com/transfer-learning-for-deep-learning/, 2019.

[58] S. Gupta, "An introduction to Computer Vision using transfer learning in fast.ai — Aircraft Classification." https://towardsdatascience.com/an-introduction-to-computer-vision-using-transfer-learning-in-fast-ai-aircraft-classification-a2685d266ac, 2019.

[59] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, "How transferable are features in deep neural networks?."
https://arxiv.org/abs/1411.1792, 2014.

[60] P. Marcelino, "Transfer learning from pre-trained models."
https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751, 2018.

[61] K. D. N., "Audio classification using transfer learning approach."
https://medium.com/@krishna_84429/audio-classification-using-transfer-learning-approach-912e6f7397bb, 2018.