Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

Ingrid Grimstad

Classification of Wader Birds' Vocalisations using Neural Networks

Master's thesis in Electronic Systems Design and Innovation
Supervisor: Guillaume Dutilleux
June 2021

NTNU
Norwegian University of
Science and Technology

Ingrid Grimstad

Classification of Wader Birds' Vocalisations using Neural Networks

Master's thesis in Electronic Systems Design and Innovation
Supervisor: Guillaume Dutilleux
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

NTNU
Norwegian University of
Science and Technology

Abstract

Monitoring birds' activity is an essential part of recognising birds behaviour patterns and estimating their population size. Doing such monitoring through long time audio recording enables us to study when a bird vocalises over smaller and longer time windows, in a way physical monitoring is not adequate. To do such , the Norwegian Institute for Nature Research (NINA) has recorded 146 days of recordings at known breeding sites for three less documented birds in Norway: the Jack Snipe, the Spotted Redshank and the Broad-billed Sandpiper. To help their research, a system was built to detect specific vocalisations in wav-files.

The system was built using neural networks with supervised learning on self-made training and test sets, mainly made using NINA's data. The top-performing solution extracted spectrograms from the audio data and used a trained 34-layered ResNet model to output whether or not the desired vocalisations were detected. If detected, the system would output what bird was detected and the timestamps for the start and end of the detection.

Several tests were done, where the true positive rate (TPR), precision and number of false positives (FP) were calculated from the number of vocalisations the system found. Two of the tests used 72 hours of NINA's data, where one looked at single sound expressions, and the other looked at longer sound sequences which are the most relevant for monitoring the birds' activity. The first test gave a TPR of 87.87% and a precision of 96.56%, while the second test gave a TPR of 89.19% and precision of 95.64%. When it comes to the true negative rate (TNR), the first test got a TNR of 99.98%, and the second got a TNR of 99.97%, while both got a total of 23 cases of FP cases. From the same data, it was estimated that a human would label with a TPR of approximately 91%. Indicating that the system with a processing time of four minutes per day of audio using a Geforce RTX3090 graphics could offer NINA a considerable speed up and similar TPR as a human could offer.

Sammendrag

Monitorering av fuglers aktivitet er viktig for gjenkjenning av adferdselsmønster og estimering av populasjon. Gjennom monitorering i form av lengre lydopptak er det mulig for oss å studere når en fugl vokaliserer, i forhold til døgn og sesongvariasjon, på en måte som fysisk monitorering ikke ville være istand til. For å studere slik oppførsel har Norsk institutt for naturforskning (NINA) gjort lydopptak som tilsvarer 146 dager med audio på kjente hekkeområder til de tre fuglene: kvartbekkasin, sotsnipe og fjellmyrløperen. For å hjelpe til med deres forskning ble det laget et system for å detektere spesifikke vokalisasjoner i wav-filer.

Systemet ble laget ved å bruke nevrale nettverk og veiledet læring på et selvlagt trening- og testset, som hovedsakelig bestod av NINAs data. Den mest lovende løsningen transformerte lydopptak om til spectrogram og brukte et 34 lags trent "ResNet" modell til å si om en av fuglelydene av interesse var til stede eller ikke. Dersom de var til stedet, ble fuglens forkortelse, sammen med start- og stopptidspunkt for lydytringen sendt ut av systemet.

Flere tester ble gjort hvor den sanne positive raten (engelsk forkortelse: TPR), presisjonen, og antall falske positive (FP) ble regnet ut fra antall vokalisasjoner systemet fant. To av testene brukte 72 timer av NINA sin data. Den ene så på enkle lyduttrykk, mens den andre så på lengre lydsekvenser, hvor den sistnevnte er mest interessant for studie av fuglene sin oppførsel. Den første testen gav en TPR på 87.87% og presisjon på 96.56%, mens den andre testen gav en TPR på 89.19% og presisjon på 95.64%. Når det kommer til den sanne negative rate (engelsk forkortelse: TNR) hadde det første systemet en TNR på 99.98% og den andre testen hadde en TNR på 99.97%, mens begge fikk 23 FP tilfeller. Fra samme data ble det estimert at et menneske ville markert dataen med en TPR på ca 91.0%. Dette tilsier at systemet, som har en prosesserings tid på fire minutter per dag med opptak når den bruker et Geforce RTX3090 grafikkort kan tilby NINA en betydelig raskere metode, som har tilnærmet like god TPR som et menneske kan tilby.

Acknowledgements

First of all I would like to thank my supervisor, Guillaume Dutilleux, for giving me the opportunity of working on this project, as well as contributing to some good discussions and guidance.

I want to thank John Atle Kålås at NINA, for providing data, information and interest for this project.

Then I would like to thank my close ones for fruitful discussions and support during the project.

- Ingrid Grimstad

Contents

Abbreviations

BS Broad-billed Sandpiper

CCN Convolutional neural network

CSV Comma-separated values

FFNN Feed forward neural network

FN False negative

FP False positive

JS Jack Snipe

MFCC Mel-frequency cepstral coefficients

ML Machine learning

NINA Norwegian Institute for Nature Research

NN Neural network

ResNet Residual neural network

SMOTE Synthetic Minority Over-sampling Technique

SNR Signal to noise ratio

SR Spotted Redshank

TN True negative

TNR True negative rate

TP True positive

TPR True positive rate

1 Introduction

The world has a rich diversity of animal species, of all kinds of colours and shapes, divided across the globe. Some live in the hot deserts, while others prefer cold Antarctica or humid rain forests. A big percentage of the human population lives in buzzing and busy cities, which is not an ideal habitat for most animal species. This is one of several reasons why most of us only get to see a small number of species throughout our lifetimes. Looking at birds specifically, more than 10 600 different species have been documented across the globe [1] as a result of centuries of studying them [2, 3, 4, 5]. However, some researchers estimates that there are over 18,043 species in total [6], leaving a world of new discoveries and information open for us to explore.

Collecting bird data has traditionally been manual labour by professional bird experts and bird enthusiasts, where they spend time out in the wild observing and taking notes[7]. Fieldwork like this is costly, time demanding and can lead to uncertainties as a result of human error in the field, without the possibility for others to validate the results. These errors could be due to lack of concentration, subjectivity and tiredness. The accessibility of cheap and high quality recording devices has opened up a new way of gathering data and reducing the effect of human error drastically, making it far easier to gather more data that can be shared and discussed with experts across the globe.

Nowadays, a microphone can be dropped out in the wild for a longer time period, to do long-time recordings of animals, also at locations that for long have been hard to reach, like deep in the ocean [8]. This is enabling us to replay an incident and have it validated or discussed with peers, leading to higher certainty and quality of data. It also opens up new doors by giving us the possibility to monitor species over longer time periods and thereby ask new questions. Nevertheless, with big amounts of data you need ways processing it. One option is to inspect the audio manually, but this is both inconvenient and time-consuming in most cases. Another option is to use technology that can help us process the data. One of these technologies is called *machine learning*. It has been around since the '60s [9], blossoming in the last decade as a result of the breakthroughs of AlexNet in 2010 [10] and by ResNet 2015 [11], which has lead to even newer methods and breakthroughs [12]. A way to describe what the technology does is that it uses data to learn patterns through experience [13]. Machine learning has been shown to be useful in solving complex problems if applied right [14]. However, good results with machine learning is heavily correlated with the amount of data available[15], which has driven people and companies to save and collect data. Some even sell the data they generate and make a good profit from it [16, 17].

The data machine learning is applied on can be almost anything, from images of pets, stock marked data, people talking, music or even what you and your friends do on social media. The need for data to solve difficult problems has inspired the creation of many free, open-source datasets and databases, made with a common interest in solving these difficult problems and educate people who want to learn about machine learning. Two examples of

well-known datasets are *the MNIST dataset*, which has over 60 000 images of handwritten numbers between zero and nine [18], and *Google Audioset*, containing 2,084,320 labelled sound clips lasting for ten second, taken from Youtube videos [19]. There are also some databases and datasets that hold pictures, videos or sound clips of birds or other animals in addition to other sounds [20]. Some are even are dedicated to only animals, examples on this are the macaulay library[21], and xeno- carton [22]. However, even the worlds largest public sound database for animals, the Macaulay Library[23], have significant gaps in the amount and the quality of data it holds for each species. This is especially true for audio recordings in the database, some species have thousands of examples, while others have barley any.

All this shows that we have merely gotten started and that there is a lot of work left that can be done. In an attempt to contribute a step further, this thesis will, in cooperation with Norwegian Institute for Nature Research (NINA), look into three specific bird species with limited information about their behaviour. These species are named Jack Snipe, Spotted Redshank and Broad-Billed Sandpiper.

1.1 The problem

The Norwegian Institute for Nature Research (NINA) wish to know more about wader birds circadian and seasonal variation in Norway. They are generally interested in getting to know all of the birds, however, the three birds; *Jack Snipe*, *Spotted Redshank* and *Broad-Billed Sandpiper* are of particular interest. These three species live mainly in a specific part of Norway during their breeding season, Finnmarksvidda [24, 25], and are considered rare in Norway by NINA's ornithologist John Atle Kålås. This has led to limited information about these species and their activity in Norway, pointed out in a report done by NINA in 2013 [24]. To get more information about the birds' activity during the breeding season, they have chosen to look at the three birds territorial vocalisations used for mate attraction, which will be presented in Section 2.1, over a more extended period.

To achieve this, they did audio recordings in Kautokeino, the western parts of Finnmarksvidda, in June of 2016, 2017 and 2018. A total of eight different locations has been used as recording site over the three years. An overview of the locations used, comments about them, and time of recording at each location can be found in Appendix A1, while the equipment used and pictures of the equipment's setup can be found in A2[1]. When adding all audio recordings together, it gives a total of approximately 146 day's worth of data [2]. All this data would be quite a time demanding to go through manually. Therefore, NINA soon realised that they had to look into other options to get the job done, which is how this project was created.

NINA wants to have a system or a tool that can take in the audio recordings (WAV-files)

[1]The reason why some of the text in these appendixes are blue will be explained in Section 1.3.
[2]Approximately 29 days recorded in 2016, 96 days in 2017, and 22 days in 2018.

and give predictions on when one of the three birds has vocalised in the sound file. A black-box version of such a system is presented in Figure 1.1. The system takes in a wav-file, processes it, and returns what is called a comma-separated value (CSV) file with timestamps of when a bird has vocalised. The CSV file as a output format was chosen for convenience as it is a widely supported format.

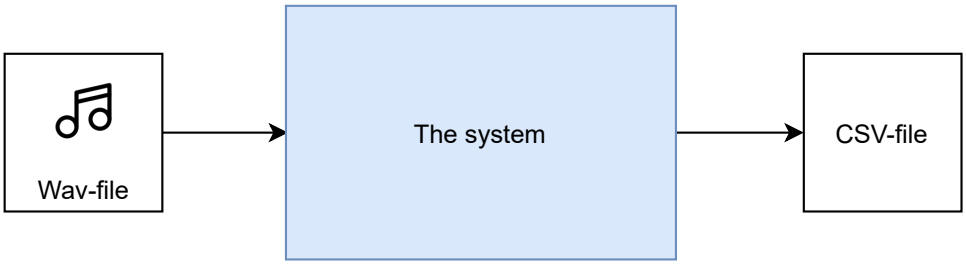


Figure 1.1: A simplified version of the system NINA needs.

To evaluate the system, some minimum requirements are set. First of all, the system should compute at least a couple of hours of data at a time. In addition, it should be able to predict whether the birds are present or not better than random. Nevertheless, the system would have to be evaluated further by NINA before it could be used in their research. NINA's main interest is to know if one of the birds of interest has been near the recorder and at which time of day. Hence, since they mainly need the time of day, one of the minimum requirements will be to say with a minutes accuracy when a vocalisation has occurred. Since they are mainly interested in whether the system found a vocalisation, the system will be tested on whether it found a vocalisation or not, not how accurate a vocalisation was predicted from start and end.

An important thing to mention is that a similar system was built for detecting the Jack Snipe as a preliminary project for this thesis, which will be presented in Section 1.2. Even though a Jack Snipe detector is already built, this thesis will try to make a system that can be able to detect the Jack Snipe, the Broad-billed Sandpiper and the Spotted Redshank all together.

Another thing worth mentioning is that even though NINA are generally interested in having a system that can provide good predictions, their wish is to have as few false positives as possible. Meaning they would rather have more false negatives than false positives. The reason why false negatives are not as bad for them is that they already expect that the recording equipment will not pick up on all vocalisations due to distance, sound propagation, and the fact that all the three birds are flying while making the vocalisations of interest, which we will come back to is Section 2.1. Nevertheless, they view false positives as bad since it requires more supervision of the system and its output.

1.2 Previous work

In this section we will take a look at previous work relevant to this project. Section 1.2.1, will look into two previous project on the Jack Snipe using NINA's data. Followed by Section 1.2.2, that will look into some other technical solutions for detecting birds.

1.2.1 Previous work with NINA's recordings

The first project done on detecting the Jack Snipe's mating vocalisation using NINA's data was the master thesis *"ANN for classification of Jack Snipe"*, written by Maja Sofie Stava. To accommodate the project, an ontologist from NINA worked through eight days from one of the locations and labelled 264 Jack Snipe vocalisation. These were divided into a training and test set used to train and test a convolutional neural network (CNN) and a long short-term memory(LSTM) network. For preprocessing, the data was divided into 4-second segments, bandpass filtered from 400 Hz to 2000 Hz and then transformed into Mel-frequency cepstral coefficients (MFCC) before Dynamic Time Warping (DTW) was applied. The best result came from a four-layered CNN model, when testing with a shorter test on 1 hour and 49 minutes containing 22 Jack Snipe vocalisations. The test gave a true prediction rate (TPR) of 0.88 and a true-false rate (TNR) of 0.91. However, it was pointed out that longer tests of 9 and 15 hours had been done, where it was concluded that predictions were more unstable due to windy weather conditions.

Due to non optimal results from the thesis written in 2020, the preliminary project mentioned in Section 1.1, was created. The project report's title was "*Detection of Jack Snipe Vocalisation using Neural Networks*" [26] and had access to the same data, but did not use any of Maja S. Stava work. When making the dataset, the eight days with the 264 pre-labelled vocalisations were gone through a second time, leading to 376 additional vocalisations being found. One of these days, containing 120 vocalisations, were used for testing, while the rest were used for training. Among the 521 vocalisations left for training, 20 were removed for being too weak and uncertain. In addition, two additional test days were labelled from another year than the rest, and from different locations to give diversity. The final solution for the project contained a system divided into three parts; a preprocessing step, a trained neural network/model, and a post-processing step of the model's data. The first part would take in the wav file, resample it if necessary before dividing the audio into 1-second segments, and then transform it into RGB spectrogram images plotted between 0 and 2 kHz. These images would then be sent into an 18-layered ResNet model, made using pre-trained weights and the dataset. For post-processing, the output is checked for vocalisations that are three seconds or longer, as these would be "accepted" as vocalisations. These would then be written to a text file with timestamps of the start and end of the vocalisations. The final test testing all 72 hours of test data had approximately 29 minutes of the vocalisation of interest. Out of the 188 vocalisations the system was able to detect 180, leaving five undetected, giving the system TPR=0.96 and TNR=1.00 [26]. Six of the 188 cases were first found by the model, showing that it only found 1 case less than a human being.

1.2.2 Other previous work

Both within bio-acoustics and machine learning have a tremendous amount of work been done, so going through it all would not be feasible. Therefore, some examples of classification of bird vocalisations have been hand-selected to give a feeling of what type of work has been done before.

The first example is Warblr, an app that can recognise over 80 different British bird species [27]. The co-founder Dan Stowell published the paper "Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning" together with Mark D. Plumbley, where they talk about the technology behind Warblr. This indicated that the app uses unsupervised feature learning in the form of Spherical k-means and Random forest with raw Mel-spectra [28]. This method requires a large dataset to be viable. Another example is the smaller app Whatbird, which can recognise 22 different Norwegian bird species [29]. It is developed by ablemagic, in Norway together with the help of the Norwegian Ministry of the Environment and Norwegian University of Science and Technology (NTNU), and has over 35 000 active users [29]. According to a master student's, Kristian Selboe, thesis from 2015 the solution consists of or at least consisted of; using MFCC as features and one Gaussian Mixture Model (GMM) per bird [30].

A third app solution for recognising bird vocalisations is BirdNET, developed by Stefan Kahl, Shyam Madhusudhana and Holger Klinck [22], in an cooperation between Cornell Lab of Ornithology and the Chemnitz University of Technology. BirdNET is the name of the app, a neural network model and the project, which is under the MIT license. The network supports 984 of the common bird species in North America, and Europe [31]. This includes the three species of interest, which no other public available solution does. The model is based on a ResNet structure, using spectrograms features as input and is trained on approximately 80 000 different labelled vocalisations [32].

The last example we will look at is a competition called "Cornell Birdcall Identification" arranged in 2020 by Cornell Lab of Ornithology [33]. The goal of the competition was to detect birds vocalising as well as possible, by using data from mainly xeno-canto [22], and a couple of other sources. The ones scoring the highest did mostly use data features like MFCC, Mel-spectrogram and spectrogram, in a combination with a supervised deep learning model [33]. The models mostly were tweaked and newer versions of ResNet, where some used pre-trained weights, while others trained the models from scratch. In addition, many of the competitors did use some form of data augmentation.

Of all work mentioned in this section, it is only BirdNET that recognise any of the species of interest in this thesis, however, it does not solve NINA's problem described in Section 1.1. Nevertheless, the solution presented in Section 1.2.1 solves the problem of detecting the Jack Snipe.

1.3 This report's structure and assumed pre-knowledge

The report is divided into six main chapters: Chapter 1 introduces the problem and previous work related to it. Chapter 2, "background material", is meant to give a brief explanation of the theory used and help a reader understand the the rest of the report. This is followed by Chapter 3 and Chapter 4 presenting how the project was implemented and the results it has given. Chapter 5 discusses different aspect of the project, what could have been done differently and future work. The thesis is concluded in Chapter 6.

As this thesis is a continuation of a preliminary project (see Section 1.2.1) some of the material from the prior report will be relevant. This is especially the case for the background chapter, as much of the theory remain the same. Therefore are some parts of the previous report be reused. To provide an proper overview of were larger amounts of text have been reused directly or it have heavily inspired by the preliminary project report [26], this text has gotten this colour instead of black.

No prior knowledge of deep learning is assumed, which is why the background chapter will go through some of the basics within machine learning and neural networks. However, it can be an advantage to be somewhat familiar with the concept. Nevertheless, it will be expected to be familiar with basic signal processing, this includes knowledge of how audio is digitised, how a filter works, what a spectrogram is and what a mel-spectrogram is.

2 Background material

This chapter contains ten sections. Section 2.1 contains more information about the three birds of interest, while the second Section 2.2, will explain some concepts and theory within bio-acoustics relevant for this project. The rest of the sections will look at different machine learning related topics; like what machine learning is (Section 2.3 and 2.5), datasets and feature extraction (Section 2.4), and neural networks (Section 2.6).

2.1 The birds of interest

In this section, we will take a closer look at the three bird species of interest. Figure 2.1 shows a picture taken of each of the birds. The first picture, in Figure 2.1a, shows a Broad-billed Sandpiper *(Limicola falcinellus)* standing in moss bog and grass, while the second one, Figure 2.1b, shows the Jack Snipe (*Lymnocryptes minimus*) held in hand by an ornithologist from NINA. The last photo, Figure 2.1c, shows a Spotted Redshank (*Tringa erythropus*) nearby the water.

(a) Broad-billed Sandpiper **(b)** Jack Snipe **(c)** Spotted Redshank

Figure 2.1: Photos of each of the birds, taken by the ornithologist John Atle Kålås, from NINA (used in this thesis with permission).

All three birds are waders, also called shorebirds, of the order Charadriiformes[34, 35]. They belong to the family Scolopacidae [34], which has around 90 different species on a global scale [5, 35]. The three birds we will look at in particular are all migratory species [34], meaning they will move to different areas throughout the year [36]. Even though they do not live in the same places all year round, they all breed in Northern Europe and Russia, in open land areas with marsh and bogs with swampy grounds [37, 38, 39].

All birds have multiple vocalisations [40]. However, what is special about the breeding season is that the male of each of the three species vocalises a particular vocalisation used for mate attraction only at this time of the year. It is this particular type of vocalisation we will be looking at throughout this thesis. There is limited information about the frequency features of the three species' mate attraction vocalisations, which is why the spectrograms (Figure 2.2, 2.3 and 2.4) and the information about them are supplied as a part of this thesis contribution.

Figure 2.2 shows a spectrogram of the Jack Snipe's mate attraction vocalisation lasting for 10-12 seconds between 500-1500 Hz. This particular vocalisation is often referred to as sounding like a galloping horse in the distance [41]. The male makes the vocalisation, where he starts by flying to a higher altitude before he dives down while vocalising from a height of 50-150 meter, often in a circular or semi-circular like pattern[34, 42].

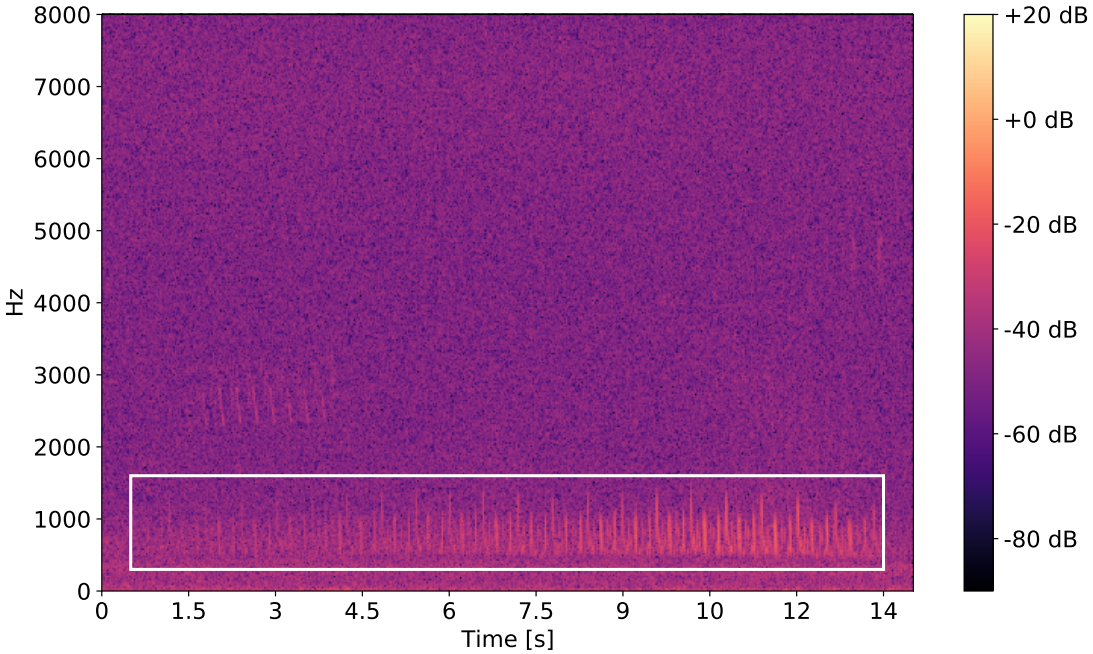


Figure 2.2: A spectrogram of the Jack Snipe mate attraction vocalisation.

The second bird we will look at is the Spotted Redshank, which has two different vocalisations that we are interested in. According to one of the ornithologists at NINA, John Atle Kålås, the first vocalisation is an alarm call often displayed before the second one. The second one is the territorial vocalisation used for mating, making it the most interesting of the two. Figure 2.3 shows an example of both vocalisations, which lie around 2.1-3.0 kHz. However the vocalisations can be found up to 3.6 kHz, which we will come back to in Section 3.2.4 The first vocalisation (the alarm call) is displayed between 1.5-3.5 seconds in 2.3, while the second one is between 3.5 and 8 seconds. The latter vocalisation is made while flying in dipping circles from a height of 20 meters down to 6 meters before it climbs again [34]. The alarm call is included because they are often vocalised sequential, and NINA wished to detect both vocalisations, for this very reason.

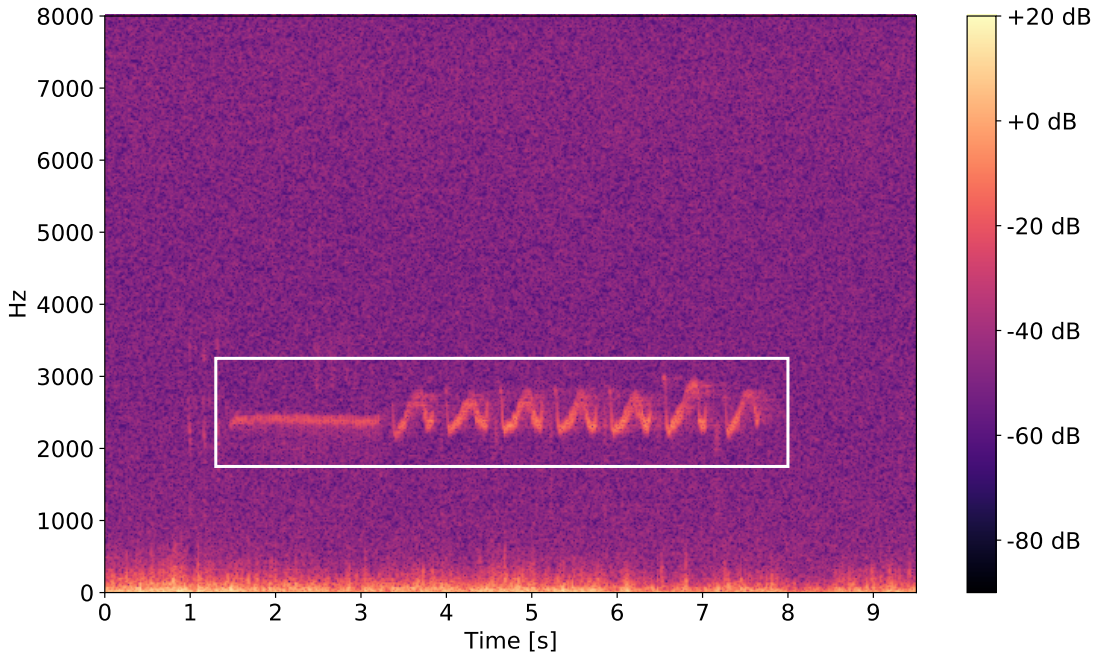


Figure 2.3: A spectrogram of the Spotted Redshanks vocalisations of interests.The first type of vocalisation (an alarm call) can be found between 1.5 and 3.5 seconds. The second vocalisation (the mate attraction vocalisation) can be found between 3.5 and 8 seconds.

The last bird's mate attraction display is somewhat more difficult. This is because the Broad-billed Sandpiper has two different mate attraction vocalisations, which can be found both separated, and together. These vocalisations are made during flight approximately 10-20 meter above the ground [34]. Figure 2.4 shows a spectrogram of both vocalisations.

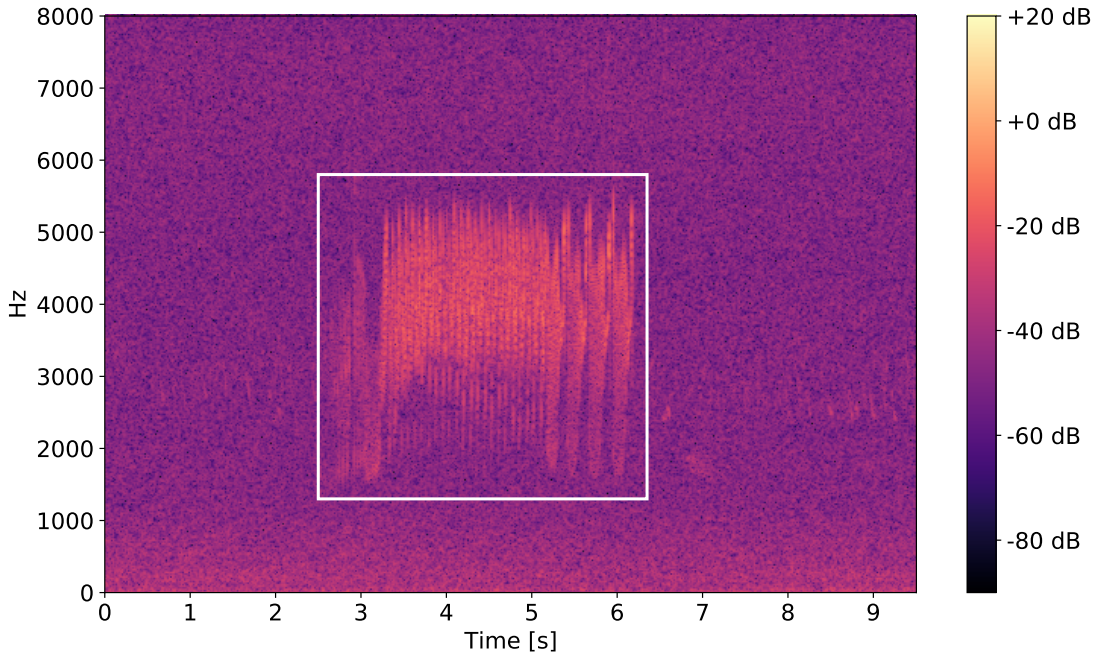


Figure 2.4: A spectrogram of the Broad-billed Sandpipers's territorial vocalisations used to mate attractions.Even though only one sound can be seen, the sounds made from 2.1-5.1 second and the 5.1-6.1 seconds appear both together and apart and are therefore viewed as two separate sound expressions.

The first one start with an "introduction" sound lasting 0.5-1 seconds followed by sound displayed as dense lines in a spectrogram, between 2.3-4.2 seconds in the Figure 2.4. Followed by the second vocalisation, which starts around 4.2 and lasts to 5.2 seconds in Figure 2.4. From Figure 2.4 we can also see that the vocalisation(s) have a broader frequency bandwidth, in this case between 1.6 kHz and 5.5 kHz, but can in some cases be stronger and therefore be between 1.5 kHz and 6 kHz.

An important thing to mention is that the spectrograms that are shown in this section only belong to one individual per species in a specific area. For some species, it has been documented that a bird, just as humans, can have dialects meaning that their vocalisations can be somewhat different from one area to the next [43] even though they sound quite the same. However, no literature was found to support any of the three birds having or not having any dialects. Another thing that is even more likely to happen is some variation in the vocalisations due to various effects happening while the sound is travelling from the source to the received, which we will look closer at in the next section.

2.2 Bioacoustics

When stepping outdoors, one encounter often a sea of different sounds and sound sources at once, either it is in the city or out on a country side. When adding all the sound sources produced in a landscape together it forms what is called a **soundscape** [44].

A soundscape can be divided into three main categories; *geophonies*, *biophonies* and *anthrophonies* [44]. Geophonies are nonbiological nature sounds, such as running water or wind. Biophonies are sounds produced by living organisms that are not human, like a bird vocalising or the sound a rattlesnake makes with its tale. Anthrophonies are human-made sounds, not limited to speech and movement from a human body, but also sounds from human-made things like cars, planes, windmills, church bells, children's toys etc.

When making audio recordings in a soundscape, there is often one or multiple sound sources of interest, often referred to as the signal. The signal could be anything; a dog bread, cats majoring or sounds made by the wind. Everything else is background noise, also referred to as ambient noise[45]. The relationship between the signal and noise is called the *Signal-to-Noise ratio* (SNR) [46]. The ratio alone does not necessarily give enough information to tell if the signal is strong or weak, but gives an indication of how easy it may be to detect the signal. If the ratio is high, it indicates a strong signal compeared to the background noise, while a low ratio indicates that there might be a lot of noise present, a weak signal or both.

Before a signal gets to a receiver, the signal has to travel through a medium from the sender. The signal will then be attenuated due to distance, atmospheric attenuation and ground effect [47], other effects are considered as negligible. The attenuation due to distance is not frequency dependent, while this is not the case for attenuation due to atmospheric attenuation and ground effect. This can therefore change the vocalisation form it leaves the sender, until it gets to the receiver, affecting how the spectrogram will look. Another thing to think about is whether a bird is flying while vocalising, like in the cases presented in Section 2.1, it can lead to the Doppler effect causing the signal to shift frequency, thereby changing the signal's frequency range. Another effect to take into consideration is if a bird is flying below the height of the microphone when it has Plexiglas plate as NINA's had during their recordings it could lead to diffraction of the signal. However, in the case of the three species of interest this does not seem to be a problem as they all fly above 3 meter while vocalising (see Section 2.1), which is approximately the hight of the equipment (see Appendix A2). The last effect we will mention is refraction due to distance from the microphone in heterogeneous medium, can cause the signal to change direction and thereby end up not reaching a microphone, or end up reaching the it because of the refraction [48].

However, for the signal to actually be detected by a microphone or another individual the signal must be strong enough, and not masked by noise. One source of noise can be wind, which can contribute to high levels of noise. Even a small breeze with a speed of 4 m/s in an open grassland can alone exceed a noise level of 35 dB(A), and accumulate a noise level between 45 and 55 dB(A) in the forest[49]. When it comes to frequencies, the wind tends to be greatest between 0 and 6 kHz but can exceed to higher frequencies as well [48]. Another important geophony sound source is rain, which can also generate high levels of noise. It has a frequency range from 0.5 kHz to 8 kHz and can generate a noise level of at least 50 dB in a forest [49]. In addition to these two geophonic sound sources, there are many others, including bioacoustic and anthropogenic sources, that can

contribute to a lot of noise, all of which can cause a sender to be masked.
Masking over time can lead to a sender not being able to communicate and hunt as well [50, 51], which has lead some species to adapt. One of these adaptations is called the Lombard Effect, where the sender increases its signals amplitude to avoid being masked [48]. Other methods are to increase the duration of their vocalisation or shift the frequency of the signal. Hence, it is important to point out that not all animals can do this; some will also adapt by changing the time of their vocalisation, or some might even leave the area if the noise level continues over longer time periods [48].

2.3 Intro to Machine learning

Machine learning (ML), is a branch of Artificial intelligence that has existed since before the 60's [52]. It is however, in the last decade that the technology has seen its true raise in popularity, as mentioned in the introduction of Chapter 1. Simply put, it can be said that machine learning is used to recognise patterns in data, by using algorithms. Algorithms, which both take input, and provides output. What is special about machine learning when comparing to other methods, is that these algorithms will learn and improve through experience, much like humans do. Moving recognition algorithms away from being solved through hard-coded rules, but instead through trial and error.

To learn, these algorithms need data to "practice" on. This is done by showing them examples, preferably data that is similar to the real problem. Not only do they need data, but they also need to be able to distinguish between when they do something right and wrong. This is done by giving the algorithm some feedback on how well it is doing. The process of showing data examples and giving feedback is called training. When an algorithm is training on data, it builds what is referred to as an model [53]. It is through the training process that the patterns in the data are found. It is this "experience" the algorithm uses when exposed to new data. However, for the algorithm to learn well it is crucial to have high quality data. Fixing the data after gathering it, and feature extraction are both a part of an important step (if not the most important) to get good results from machine learning.

2.4 Datasets and feature extraction

As mentioned in the introduction chapter, the input data of machine learning models/algorithms can take many forms; it can be audio, images, sensor data, or data from the stock market etc. A *dataset* is simply put a collection of related data [54], which is needed during training and testing of a model. Each data example in a dataset is called a *data point*[55], which could be an image of a stop sign in a dataset of images of road signs. For the algorithm to learn well, it is necessary to have a dataset with quality data, and to give it a large enough number of data points. More data often leads to better results, or a more "correct" model [15, 56]. It can be difficult to define what "good" and "enough" data is. If we were to create a classifier that should classify if a sound is a dog barking or a cat miaowing, we could likely do a fine job by recording the same cat and

dog many times. However, if we then redefine the problem to not only tell us if the sound is a dog or a cat, but to also tell us what breed the animal is, the data would not be nearly sufficient enough. A high quality dataset in the redefined problem would require us to gather sound clips from as many dog and cat types/breeds as possible, big, small, schafers, chihuahuas etc. Defining what makes up high quality data is difficult, but the data should at least have enough information for the model to learn about the defined problem. Another example of things to think about in the cat and dog example is; where we want it to work? Should the model recognise the data from any soundscape, or should it just work on recordings done in a music studio? If it should work in several soundscapes, the training data should also have different soundscapes, while if not, it might not be needed nor beneficial.

The datasets can be labelled or unlabelled. A labelled dataset assigns each data point to a specific class by giving it a label, while unlabelled datasets do not have any such information[53]. Both cases need data, but making a labelled dataset requires one to go through the data and assign appropriate tags or classes to each data point.

Data can be represented in countless formats and sizes. Finding features in the data that makes the problem easier to solve for the algorithm is a time consuming and difficult process, but can have huge impact. To look at one example, could spectrograms and Mel-frequency cepstral coefficients (MFCC) be features that promotes the frequency features in audio data. Processing the data and finding useful information requires a good understanding of the problem at hand, and often some field expertise around the data type to transform the data into something useful. Data can be anything from audio, images or social media engagement, all requiring a different asset of skills and understanding.

When gathering data for training a model, it can have different format and sizes. However, the data that goes into a model often has to be of a particular format and shape. As we will see later, some ML methods take in 1D data, while others take in 2D or 3D data of a specific format and size. This means that the data given as input to a model, should be standardised if it is not already on the required format. For getting the data to a specific shape and format, data transformation, also called pre-processing is used. However, it can be important to know that the format of the data is not necessarily chosen just because of the model. It is chosen based on what data format combined with the model seems more likely to give successful predictions.

The data transformation can be so many different things depending on what type of data it is. This may involve resizing or normalising the data. It can transform audio into a spectogram, Mel-frequency cepstrum coefficients (MFCC), or waveforms. Changing an image's lighting, or rotating it, or taking the image and "flatten" it out which would mean to go from 2D to 1D by changing the shape from Height$\times$Width to 1$\times$(height$\times$width). All these transformations are usually done by a programming script, which gives the "final" dataset that is the input of the model. Many of the things listed above are so-called *feature extraction* methods. They can be used to change the format of the data and extract a

particular feature or "part" of the data. The spectrogram and MFCC is an example of extracting frequency features of audio segments. Feature extraction is an essential part of ML. It determines what the model has to work with, and can be the difference of failure or success.

2.5 Learning types within machine learning

Now that we know about the concepts of training an algorithm and preparing data for it to use, let us go through some different ways an algorithm actually learns. There are three main categories:

- Supervised machine learning
- Unsupervised machine learning
- Reinforcement learning

Supervised machine learning: Is where you use a labelled dataset for training it. As the name suggests, the algorithm is trained with a "supervisor", or a "teacher" that gives feedback on what was right and wrong. The training process is a repeating cycle where the algorithm makes a prediction on each data point, and gets feedback from the supervisor on how well it performed. The latter part is done by comparing the predictions with the labels. The algorithm can then use this to make small tweaks to try to improve its performance for the next cycle [53].

Unsupervised machine learning: On the other hand, uses unlabelled data, and the algorithm has to try to find some pattern itself without a teacher[53]. This is typically used for problems such as clustering, where it is possible to get feedback to the system in the form of a closeness measure of data. Other example on where unsupervised learning could be useful in abnormalities, which maybe could find something the human eye could not see on its own.

Reinforcement learning: Is based on learning in an environment through feedback given in the form of a reward, where the goal is to make the model for satisfactorily interacting with the environment [57, 58]. An example of an application is that it could be used in games to learn how to play. It can then be "rewarded" or not, by how far it gets. So, in other words, if it gets a new high score, it should get rewarded.

Within all of these categories different types of machine learning can be applied, like the Support Vector Machines (SVMs), Decision Trees and Random Forest, clustering, and neural networks (NN) [53]. From here on and out we will focus on the the latter method, called neural network, which can be applied both within supervised and unsupervised learning. However, we will only focus on supervised learning.

2.6 Neural networks

Neural networks (NN), also called artificial neural networks (ANN), are inspired by the biological neurons in the human brain [53, 58]. Like the brain does neural networks have several layers, but how many can vary. A Neural Networks have an structure of one input layer, at least one hidden layer and one output layer. Within these layers, are what we call an artificial *neuron*, but can also be referred to as a node. Figure 2.5 illustrates a basic network with one hidden layer.

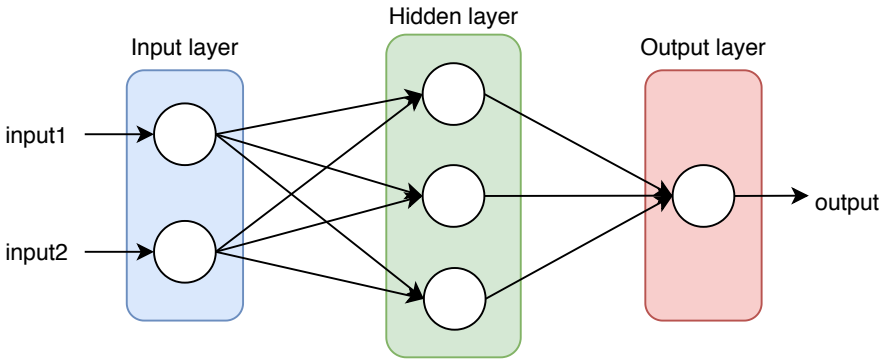


Figure 2.5: A simple neural network with one hidden layer.

The circles in Figure 2.5 are the neurons. They are the points of communication between the layers, allowing us to propagate data from the input- to output layer. Each such neuron holds a number. What this number is or represents, depends on what layer we are looking at.

The input layer is what takes in the data from a dataset. The format of the data decides the number of neurons in this layer. In the Figure 2.5 the data point sent in would have two numbers, since there are two neurons. It could, for example, be used to hold one number that tells about the texture of a fruit (represented by a number), and one to tell about the fruit's weight. Example on numbers of the first and second node in the input layer could have been two and 231. Where the two would maybe represent a smooth surface, and 231 stood for how many grams it weighed. An input layer could also for example have taken in a 24×24 black and white image, which would have needed 576 (24×24) nodes in the input layer.

The number of nodes in the output layer, on the other hand, is determined by the number of classes the input data could belong to. Where if the input data belonged to the class a node represented, it should ideally hold a "high" number that is sent out of the node. If the data, however, did not belong to that node, it should give out a "small" number. The output ranges typically from 0 to 1, where 0=No and 1=Yes, but this does not have to be the case. In the example about the fruit data, the network could have one node, as in Figure 2.5. This could for example have been used to determine if it is an apple or not. However, it could also have five nodes if we wanted it to say whether it was an apple, banana, kiwi, orange or pineapple.

The numbers held by the nodes in the hidden layers are a bit more complicated. To help explain it we have Figure 2.6, which shows how one node is connected to all the nodes in the previous layer. The first layer is in this case, the input layer, with a length N, which have N connection to the node of interest.

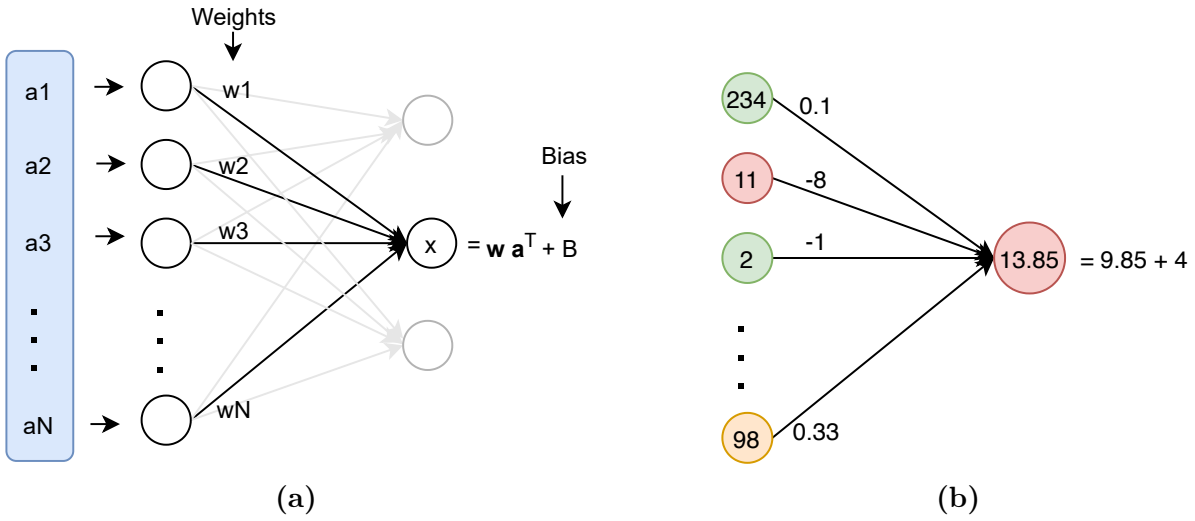


Figure 2.6: Two figures that demonstrates how the hidden layer nodes gets their values.

The connections between the neurons, will have various strengths, just as the connections between the biological neurons in the brain. The way these strengths are "simulated" is through numbers, which is why they are called *weights*. In Figure 2.6a, they are symbolised with an $\mathbf{w}$ in the equation, and $w_i, i = 1, 2..., N$ per connection. Each of these connections, is represented by an individual weight/number. In addition to the weights between the connections of the layers, there is an additional weight, called the *bias*. Each neuron has a bias, which is symbolised by a B in the figure.

When all the inputs goes into the first layer, they will have a value called the activation value, $\mathbf{a}$. This value tells how "active" a node is, or how high number it has. Then these values will "travel trough" the connections to the node, and be "formed," i.e. multiplied with a weight. All these values are then summed up, and forms the node value, x, together with the bias, B, as shown in Equation 2.1 and Figure 2.6a.

$$x = \sum_{i=1}^{N} W_i a_i + B = \mathbf{W}\ \mathbf{a}^T + B \tag{2.1}$$

This happens for all the nodes in a hidden layer. Figure 2.6b, illustrates how Figure 2.6a could look like with values. The example also shows that an activation value will have little impact if the weight is low. However, this is also one of the reasons why "neighbour" nodes in the same layer, can have such different values, and extract different information.

As mentioned in the introductions of this section, does a model at least have one layer but can have several as well. When a model have many layers they are often referred to as deep models/networks. If we look at the weighted sum in Equation 2.1, and think of deep neural networks, one can imagine that the sum can grow quite large. One of the reasons why these numbers do not "grow out of hand" is *feature scaling* of data, and *activation functions*. **Feature scaling**, is often applied on the input data because ML algorithms usually do not work that well when the input have very different scales [53]. The feature scaling is done in the data transforming step, before the data enters the algorithm/model. One of the common ways to do scaling is by using min-max scaling, also referred to as normalization [53]. What it does is rescale and shift the input data between the range from zero to one.

Activation functions are another way to "scale" the data, but inside the model. These functions are found at the node's output, as Figure 2.7 shows. They take in the weighted sum and the bias from the node and scale them in certain matters to determine what will be sent out and "how active the node is".

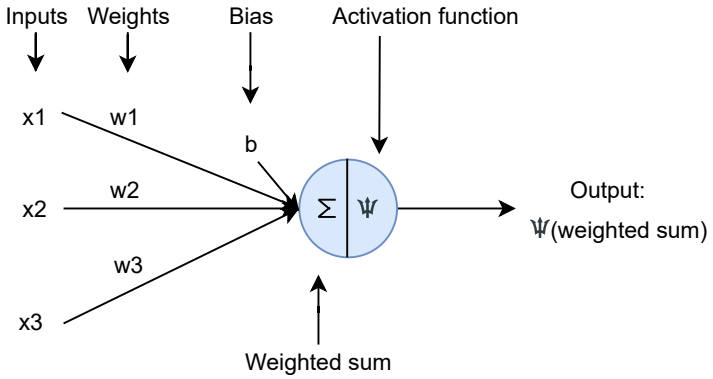


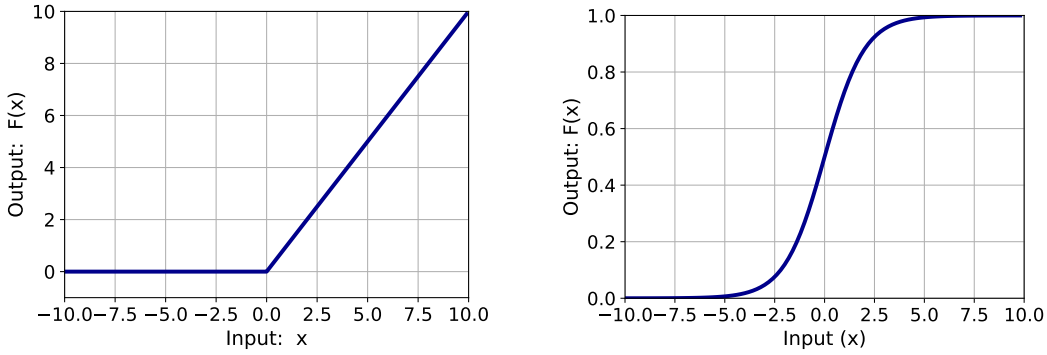
Figure 2.7: A node showing how the activation function sits on the end of the node, taking in the weights and bias.

Activation functions can both be linear and non-linear, where the latter is most used. This is because it introduces non-linearity, which is often needed to solve complex problems humans have a hard time with. Two well-known examples of non-linear activation functions are *ReLU* and *Sigmoid*. The ReLU-function scales the data such nothing is below 0, like Equation 2.2 shows. Hence sigmoid scales the numbers between zero and one, as Equation 2.3 shows.

$$ReLU(x) = max(0, x) \qquad (2.2)$$

$$Sigmoid(x) = \frac{1}{1 + e^x} \qquad (2.3)$$

Figure 2.8 shows both these activation function plotted with input vs output.



(a) (b)

Figure 2.8: The two activation functions ReLU and Sigmoid plotted with input vs output. Figure 2.8a shows the activation function called ReLU, while Figure 2.8b shows Sigmoid

Throughout the model will often the same activation function be used, except from the case of the output-layer. The activation function referred to as the output function, shapes the model's output. This is why one often have to look at what type of problem one have, and what type of output one want when choosing this function, since all activation functions has its advantages and disadvantages. This means that they can work great for one type of model and problem, but not necessarily for all. Throughout the model will often the same activation function be used, except from the case of the output-layer. The activation function referred to as the output function, shapes the model's output. Besides, some functions suits some problems better than others, as some will depend on the other classes outputs while others will not.

2.6.1 Training the model

When training a neural network through supervised learning, the model's weights and biases start by being initialised with random values, if not configured differently. This is the reason why an untrained model often outputs relatively useless results, and that it has to be trained. During training, the model will first be given data, that propagates through the network. Then the results/predictions will be compared with the labels in the case of supervised training, used to calculate a measure called *loss*. The *loss* describes how far the model's predictions were from the truth, and is calculated by the use of a loss function, also referred to as a *cost function* [59]. Which cost function one choose to use depends on the problem. Some cost function will compute the loss of the output independently of potential other outputs while others will not. The latter is a problem if it is a multi-label classification problem, as several outputs might be "True".

The loss is then used in a step called the back-propagation to improve the model. This step involves using gradient descent, to compute in which direction the model should change the weights and biases for getting the most significant loss reduction. After finding the "best way", the model parameters are shifted in that direction by the amount the

learning rate allowed. The learning rate says how big steps the backpropagation step is allowed to take in the direction the decent gradient algorithm suggests. After this is new data "feed" to the model and "worked" trough, were the backpropagation scheme is applied and so on.

For not letting abnormalities in the data affect the model too much, as well as save computation resources, the model does not only go trough one data point at a time. It goes through a **batch** of data, calculating the loss average, instead of the loss of one data point. When the model has gone through all the data points in the training data, it is said to have gone through one **epoch**. When training a model the data is gone through Y number of epochs, which is a self-chosen number set when choosing the parameters for training. The batch size and learning rate is also self-chosen number. Of these are the learning rate particularly important, and have a big impact on whether the training will succeed or not [53].

For each time the backpropagation scheme is applied, the hope is that the loss will decrease, and the accuracy increase. However, we do not actually know how the loss would look like as a function of initialised values (weights and biases values), but it is often illustrated by a graph similar to the one in Figure 2.9.

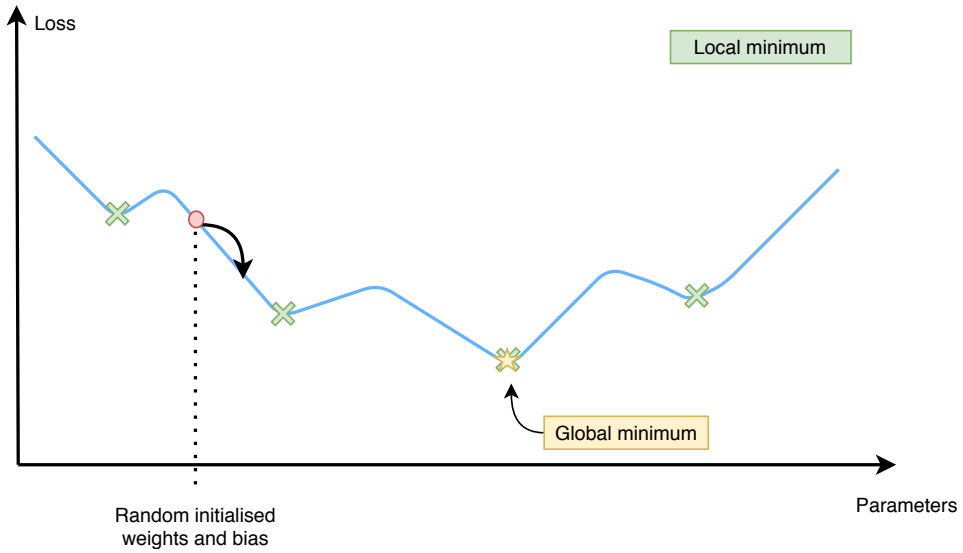


Figure 2.9: A simplified graph symbolising how the relationship between the loss and the model's parameters can be illustrated. The graph demonstrates how the model will start with some parameters, which will place it close or far from one or multiple local minima and the global minimum. The arrow symbolises how the training will change the model's parameters and move the model along the x-axis.

The idea is that the random initialised values will put the model somewhere along the blue line, which symbolises the variation of loss. Along this line, there are multiple *local minimums* and a *global minimum*. The goal is to get as close as possible to the global

minimum, and it is here the learning rate comes in. If the learning rate is too big, then the model can end up taking steps over a local or the global minimum. This is not necessarily a problem, however, if it is starting to stagnate over a local minimum, the model can have missed out on having a smaller loss. On the other hand, if the learning rate is too low, it can end up in a local minimum which is not the global minimum. Therefore, the learning rate value is important when it comes to achieving accurate models. There are several approaches to deal with this, by changing the learning rate during training. This can either be after a particular time or after the change in the loss has reached a value.

The examples we have seen upon until now are networks where each node in one layer are connected to all the nodes in the next, which are called a fully connected layer. When all the layers in a model are fully connected, it is called a fully connected model. The network we have so far, where the data only goes forward is such a model and is called a feed-forward neural network (FFNN). However, this does not have to be the case. A model can have both none or some fully connected layers. In the next section we will look at an example of the latter one.

2.7 Convolutional Neural networks

Convolutional Neural networks (CNN), also called ConvNet, are considered a feature extractor, and are typically applied on images. The model's layers extract information about the image; while at the same time compressing the data. These models can both be used as feature extractor, or as a pre-processing step before feeding data into another network structure or a bigger system. A CNN typically has three types of layers; convolutional layers, pooling layers and fully connected layers, which will be explained in the Section 2.7.1 to Section 2.7.3 [60]. We will go briefly through all of them to get an idea of how they work. However, the latter one has been introduced at the end of last section, and will therefore not be mentioned in great details.

2.7.1 The convolutional layers

Each of these layers has what we call filters or a kernels. These are the elements carrying out the convolution to extract the information from the data. The filters are composed of an array of weights, which is trained to give the best possible feature extraction, with a corresponding bias per filter[53]. Figure 2.10 shows how the filter would be used on the input array, to get an output, called a feature map[53]. The filter's values are multiplied with some of the input values at a time, before being multiplied with some of the next values, and so on. Each time are these values summed up, and the filter's bias value is added, as the figure shows. Just as for FFNN the weights and biases are randomly initialised, and not "handpicked". Then during training, backpropagation is used to improve the filters.

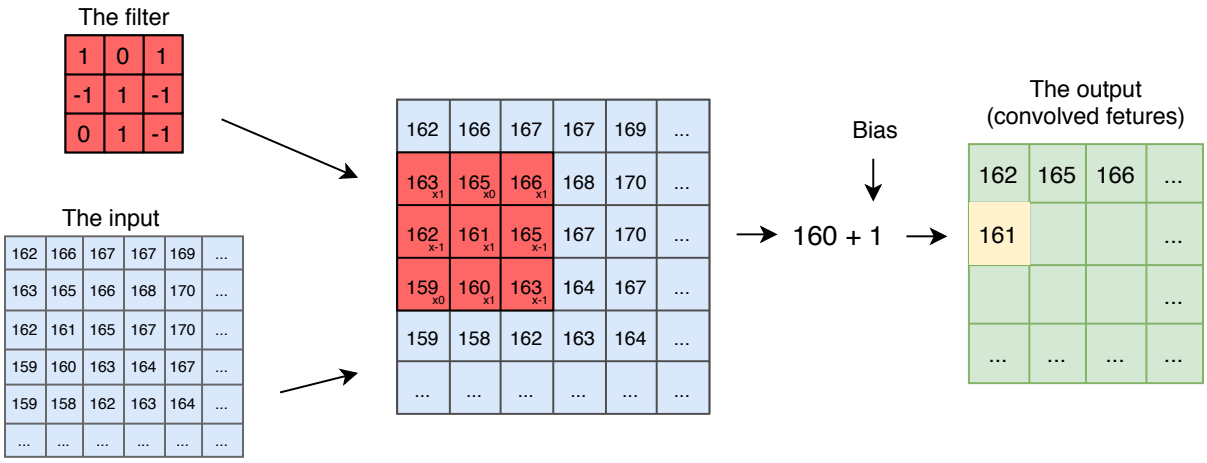


Figure 2.10: An illustration of how the the convolution is done by a filter, on the input.

The size of both the input, the filter and the output can vary, but the filter usually is not too big[53]. In the example in Figure 2.10 the filter is moved just one pixel to the right at a time, then down one row at the end, until it has gone through the whole image. Nevertheless, this does not have to be the case. The filter could also have gone two steps at a time. How many steps it takes is a parameter called striding [53], this parameter will therefore affect the output size. Another parameter used, that will affect the output is called padding or zero padding [53]. The name comes from that zeroes are padded around the input (image) if the padding is anything else than zero. Each layer has typically multiple different filters, with different parameters(weights and biases) that are applied to the same data. These extract different features, resulting in different feature maps that are stacked upon each other[53].

2.7.2 Pooling

The pooling layers "mission" is to shrink the data size, to reduce the computational load[53]. These have some similarities to the filters. Their size has to be defined and how they should "move" (stride and padding size)through the data. However, they do not have randomly initialized weights. They are either calculating the average or find the highest number within the pooling kernel [53]. Figure 2.11 demonstrates how the two pooling methods work. In this case it is a 2x2 kernel, with two in stride and zero in padding.

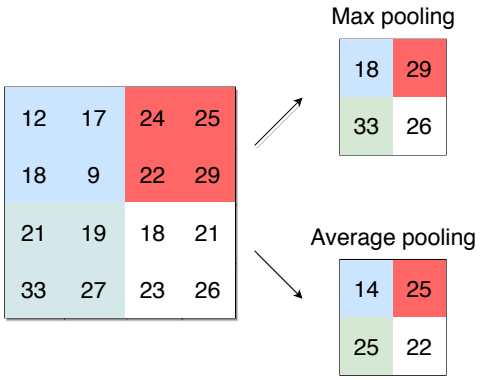


Figure 2.11: An example of how both max pooling and average pooling works, using a 2x2 pooling kernel, with two in stride.

2.7.3 Fully connected layers

As mentioned earlier, is fully connected layers are where all the neurons in one layer are connected to all the neurons in the next layer. After the convolutional- and pooling layers are often one or multiple fully connected layers used. They are used to classify the data from the features extracted in the previous layers, and to give the final output of the model as Figure 2.12 shows. How many convolutional or pooling layers used can differ, and the same goes for the fully connected layers. Before the fully connected layer(s), the data is usually flattened to a 1D array. Either by using a pooling layer, or a flattening function. It should also be mentioned that in between these layers are typically activation functions like ReLU used [53].

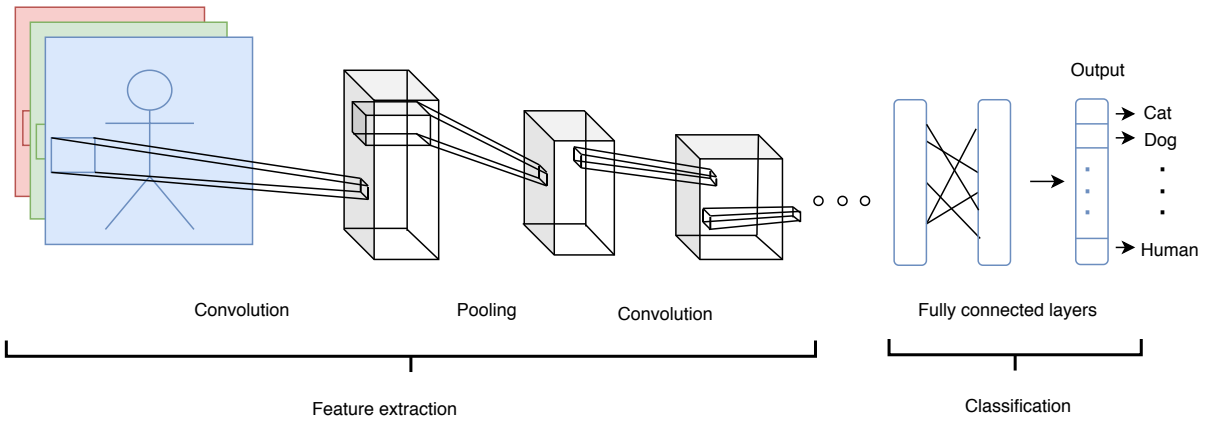


Figure 2.12: An overview of the different steps a CNN model can have.

One last thing that should be mentioned is when talking about the number of layers a model has, it is a common practice to only count the layers with parameters [53]. For CNN, this means the convolutional layers, and the fully connected once, but not the input and output layer, the pooling layer, or a flattening step (if used).

2.7.4 Residual neural network

Residual neural network, ResNet, was the type of network *Kaiming He* used for winning the the ILSVRC 2015 challenge with [11], using an 152 layered, deep CNN. However, it is important to point out that smaller versions exist (18, 34, 50, 101 layers) [11]. The key point about the ResNet compared to other deep CNN, is that it "skip connections", or uses "shortcut connections" [53], as Figure 2.13 show. F(x) in Figure 2.13, is how the layers will affect the data, while X is the input data. By letting the input "bypass" in addition to go through the layers, and adding it to F(x), one gets what is called "residual learning" [53]. By using shortcut connections in deep model, like in Figure 2.13, one can solve one of the big problems deep neural networks are facing [61].

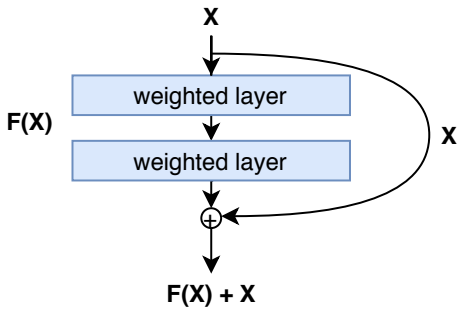


Figure 2.13: A building block using residual learning.

2.8 Transfer learning

Transfer learning is a method within machine learning where you re-use a model that is trained for a similar problem, to your own. This is done by either reusing the trained weights and biases from parts of the model or the whole model, in a new model. Then retraining it on your own dataset. This teqhnice is especially usefull when the dataset is not to big [53].

Within transfer learning there are two approaches for training:

(i) The first one, is where one freeze some, or all of the layers from the pre-trained model. Preventing the training from changing any parameters in the frozen layers, and only change the other layers.

(ii) The other approach is to continue training the network with the pre-trained weights, on the new dataset.

2.9 Overfitting and underfitting

Overfitting, is when the model is trained too well on the training data, to be able to generalise well [62]. It happens when the model is to complex for the problem [63], and the model becomes to specialised on the training data to be able to generalise well. Because

the model is too complex, it makes extra adjustments to make the model "fit" the training data better. This can make it seem like the model works better than it actually does. The risk is then having good training results, but bad test results. If the training and test data's accuracy and loss are measured during training, and then plotted, the plot can sometimes show if a model is overfit. However, overfitting does not have to give poor test results if for example the test data is not different enough from the training data. This is one of many reasons why one should think through what data one uses for both training and testing of a model. There are several measures that can be taken against overfitting, some examples are getting more training data, and having a model with fewer parameters [53].

Underfitting, on the other hand, is the opposite of overfitting. It happens when the model has too little training to generalise the problem [62]. It happens when the model is to simple for the problem [53]. When a model is underfit it does not mean that running the model over more epochs will necessarily help. When a model is underfit, it could indicate that the training data does not has enough data or good enough "examples". Approaches to deal with underfitting is for example choosing a more powerful model, and choosing a better features to feed the model [53].

Both overfitting and underfitting are big problems within machine learning, with many more approaches to avoid them than the ones mentioned above. However, it does not mean that it is not a problem anymore, and is something one have to consider when looking at a model's results.

2.10 Balanced and unbalanced datsets

The last topic of this chapter will be the balancing of data between classes in a dataset. When there is approximately the same amount of data points per class in a dataset, it is referred to as a balanced dataset. However, if the amount of data is not divided equally, which is often the case in the real world, it is called an unbalanced or imbalanced dataset[64]. If a dataset is unbalanced, it can become a problem because when one or multiple classes accounts for the majority of the data training the model, the model can become biased against the majority class[65], leading it to fail to understand the underlying pattern in the data, which is supposed to help differentiate the classes. However, it is important to emphasise that an unbalanced dataset does not automatically lead to this problem, meaning it does not have to be a problem.

Nevertheless, if it seems to have become a problem or one wants to take precautions against it, several preventative techniques are out there. Examples of strategies are; collecting more data, resampling the data, or using techniques that penalise wrongful predictions of the minority classes heavier than the majority class [66]. This part will focus on the second example, resampling of data, which means to up-or downsample the dataset: Downsampling is when data is removed from the dataset either randomly or by choice, while upsampling entails adding examples by duplicating existing examples in the original dataset or creating new examples from existing ones [65]. Both can be used to

make the dataset more balanced, but the latter can also be used to make more data. A technique often used for this is data augmentation, which takes existing data and modifies it slightly in a realistic matter, so it becomes a "new" data example[53]. Examples of data augmentation techniques on an image could be to rotate or resize, while on audio, it could be adding noise or other sounds.

3 Methodology

This chapter contains nine different sections. The first section, Section 3.1, gives an overview of a suggested system to solve NINA's problem described in Section 1.1, followed by Section 3.2, which will try to give a better overview of the soundscape, the challenges related to the data and how the data available was utilised. Continuing with the labelling process in Section 3.3, before Section 3.4, Section 3.5 and Section 3.6 presents the pre-processing, the model's structure and the post-processing of the data. Section 3.7 shows the software and hardware used to build the system, before Section 3.8 and Section 3.9 rounds of the chapter by talking about how the models were trained and how the final system was tested.

3.1 System overview

To solve the problem described in Section 1.1, we need a system that can detect the vocalisations presented in Section 2.1. Figure 3.1 shows a simplified overview of such a system containing three main parts or steps; a pre-processing component, a trained model, and a post-processing component.

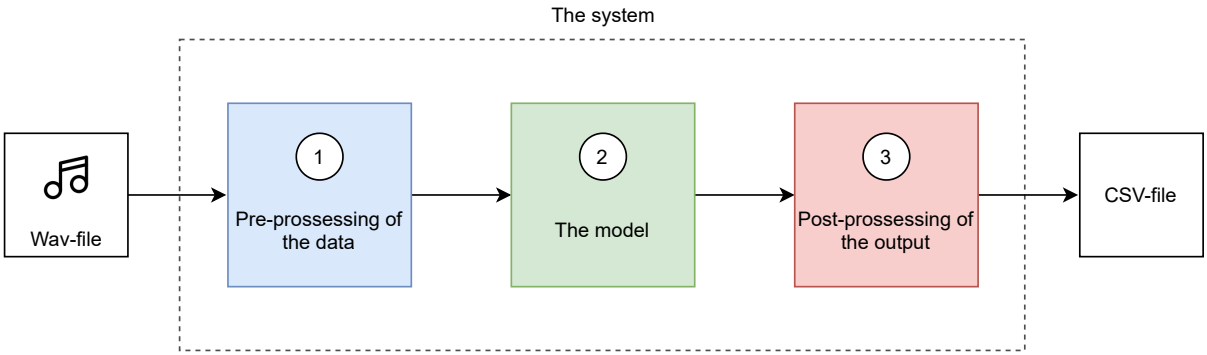


Figure 3.1: The suggested system contains three parts or components: A pre-processing part that will take in wav-files and prepare the data for the model, the model, and a post-processing part that process the model's output and transform it into a human-readable format, in the form of a CSV-file.

The components can be listed as follows:

1. **Pre-processing of the data:** The pre-processing component reads raw wav-files, and is responsible for formatting the data to be used by a machine learning algorithm. The wav-files can have different durations and sampling rates, so this component ensures that all the data has the same sampling rate of 16 kHz, which is done by up-or down sampling the data if needed. Other sampling rates could likely have been used as well, but it was chosen since all of NINA's data had a sampling rate of 16 kHz. This pre-processing component is also responsible for transforming the data and extracting features if necessary, to collect valuable information and prepare the data for the model. This is explained further in Section 3.4.

2. **The model:** This component contains a trained neural network that utilises the output from the pre-processing step. From this data, it produces an output for each data-point to say if this data is a known bird, or if it is something else entirely. This data is then sent to the post-processing step.
3. **Post-processing:** The third part of the system uses the model's output and transforms the data into a convenient, human-readable format that the researchers at NINA can work with; a CSV-file. The file contains a list of vocalisations the model predicted to have found in the wav-file. Each vocalisation is tagged with the abbreviation of the bird (*JS*, *SR* or *BS*) and a timestamp of when the vocalisation was found in the wav-file. Details about this post-processing component will be presented in Section 3.6.

The system described above illustrates a simplification of how the system looks like in "production". The system behaves a bit different when training: The two first blocks in Figure 3.1 are more or less identical, but the model's output is not post-processed when training. It is instead used to give the model feedback to learn. A choice was made early on in the project to use neural networks and supervised learning to try to solve the problem. This was motivated by good results from both the preliminary project and other similar projects, some of which are listed in Section 1.2. In addition, some of the data was also already labelled as a result of the preliminary project.

When looking at the problem defined in Section 1.1, we stand between two types of classification: *multi-class classification* and *multi-labelled classification*. A multi-class classification problem only has one class for each data point, while a multi-label problem can have multiple classes that belong to one data point [53]. The fact that multiple of the vocalisations of interest can be present at the same time is important to have in mind when defining the classification problem. This scenario was rarely found in the inspected data, so one could likely have chosen to ignore these cases without losing much information. However, since viewing the problem as a multi-label classification problem would represent the problem better, and make it easier to expand the dataset with more frequently overlapping classes if ever desired, this thesis views it as a multi-label classification problem.

3.2 A closer look at the data

This section starts by looking at what audio data is available on the three birds, both from publicly available sound databases and NINA's recordings done in Kautokeino. Continuing by presenting the soundscape in NINA's data, and looking into potential challenges in the data such as the SNR of the vocalisations. Before rounding off this section with how the data available was utilised.

3.2.1 Public available data

As mentioned in the introduction of Chapter 1, there are multiple sound databases containing bird vocalisations. Most of these are under different creative commons licenses, making it possible to use them for research such as this project. However, since this project is looking at less common birds, there is a limitation to what is available when it comes to quantity and quality. It is also not enough to find just any audio of the birds, but it has to be specific vocalisations, making it even harder to find interesting data. Nevertheless, there are a few examples of the specific vocalisations in the three databases: *Macaulay library* [21], *Xeno-Canto* [22] and *Tierstimmen Archiv* [67]. All data tagged with the birds of interest in these databases were inspected. The number of vocalisations gathered from these databases will be listed together with the other datasets in Chapter 4. This data is mostly recorded by bird enthusiasts, with no requirements to recording equipment used, audio quality or duration of the audio. Some of the audio is even modified with filters or edited so that parts of the sound clip has been removed. Most of the data was stored in a mp3 format, a compressed format [68], instead of a lossless format like wav-files[69]. All this means that the publicly available data has a huge variety in the quality of data, sampling rates and durations of the audio clips, often with high noise levels.

3.2.2 The available data from NINA

The amount of data from NINA was tremendous in comparison. As introduced in Section 1.1, there were about 146 days worth of data made available to this project, all with a sampling rate of 16 kHz, recorded on a total of eight different locations over three years (see Appendix A1). The recordings were carried over different durations. Sometimes only over a few hours within a day, while other times they were carried out over several days almost continuously. Due to the equipment's limitation, approximately 18 hours and 38 minutes is the max duration of a recording before needing to start saving data in a new recording file. Therefore, many of the recordings carried out over a full day (24 hours) were split up into one 18 hour and 38-minute file and one 5 hour and 20-minute file. When it comes to the equipment chain, the only variation of the equipment used across all the recordings was that the microphone used at location 6 in 2016 and location 7 in 2017 was different, and that an additional wind-screen was tested on top of the microphones in 2018 (see Appendix A2). This implies that the data from NINA might have little diversity in the data recorded when it comes to the equipment used. However, the recordings are done at all hours of the day, over many days, with changing weather conditions and variations in other parts of the soundscape, resulting in large variety of background noise, which we will look closer at in Section 3.2.3.

3.2.3 The soundscape in NINA's data

If we look back at Section 2.1, we saw that each bird had its frequency range. The Jack Snipe had between 0.5 kHz and 1.5 kHz, the Spotted Redshank had between 2.1 kHz and 3.6 kHz, and the Broad-billed Sandpiper had a frequency range of 1.5 kHz and 6.0

kHz. However, they are far from the only sounds in their frequency ranges. The following part will look at the main sound sources in each of the soundscape categories presented in Section 2.2; geophonies, biophonies and anthrophonies. Table 3.1 shows all the main sources found within each of the categories. The reason for looking into all these sources is that they can all mask the vocalisations when they are in the same frequency range or create false positives for the system by having similar data characteristics/properties as a vocalisation.

Table 3.1: An overview of the most dominating sound sources within each soundscape category, as introduced in Section 2.2.

Soundscape catagory:	Most common sounds:
Antrophony	Motors and gunshots.
Geophony	Wind, rain and running water.
Biophony	Other birds in the area.

Of the three categories, anthropogenic sounds is the least dominant at most of the locations. The exception is location five and sometimes at location six, where a lot of traffic noise is a big part of the soundscape. Distant motor sounds does, however, appear occasionally at the other locations as well. An example of a motor sound is shown in Figure 3.2.

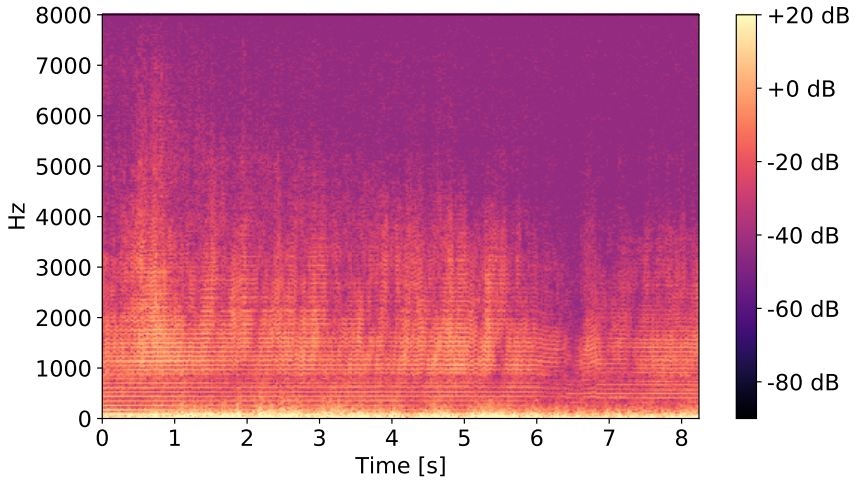
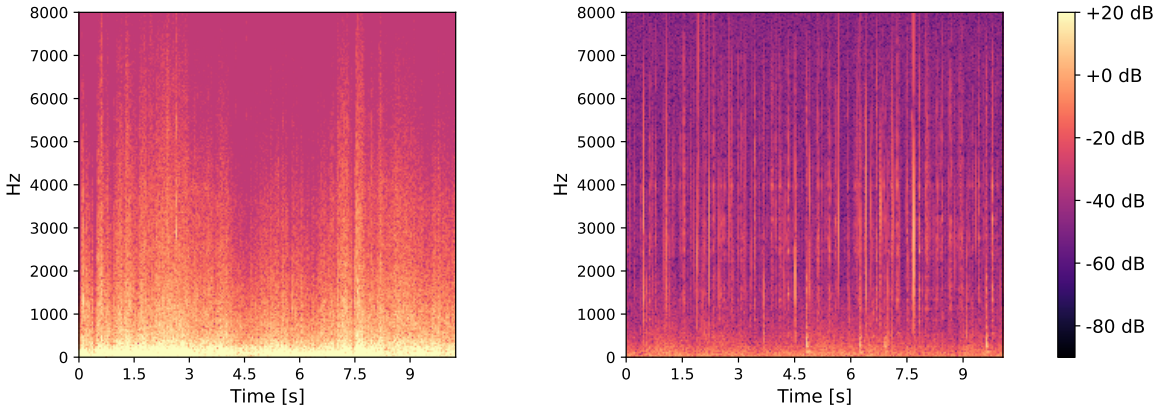


Figure 3.2: A spectrogram of motor noise in a sound clip taken from NINA's data.

The motor sound can cover significant parts of the spectrogram from 0 Hz to around 5 kHz, which can cause problems as it overlaps with the frequency range of all vocalisations of interest. Other anthropogenic sound sources found are; humans talking, gunshots and animal bells.

When it comes to geophony sound sources, wind and rain are the most dominant. Figure 3.3 shows two spectrograms of both wind and rain.



(a) **(b)**

Figure 3.3: Two spectrogram of the strongest geophony sound sources. Figure 3.3a shows a spectrogram of strong wind between 0 Hz and 5-8 kHz. Figure 3.3b shows a spectrogram of rain covering the the whole frequency range of 0-8 kHz.

Some degree of wind is often present within the frequency range of 0 Hz to 0.1-0.2 kHz. However, there are cases of stronger winds having a frequency range of up to 5-8 kHz, as shown in Figure 3.3a. Rain covers a broader frequency range and often occurs from 0 to 8 kHz, while the density of the sound it produces and its strength will vary much more. Figure 3.3b shows an example of a spectrogram during a rainy sound clip. Another geophony source, found several times but not as often as the two prior once is running water, mostly generated sound between 0.2-2 kHz.

When it comes to biophonic sounds, other birds are by far the most significant, even though there are other sources such as insects. Most of the birds vocalise within the frequency range of 1.5 kHz to 7 kHz, but some vocalise outside this frequency range. Based on data from previous observations, registered at *artsdatabanken* [70], confirmed by the ornithologist John Atle Kålås, there are approximately 100 birds species that can potentially be heard in the area in the recording periods of June. Many of them are less common and will have little anticipated activity. According to the ornithologist, approximately 30 of them will be heard regularly, while twelve are expected to dominate the soundscape at the recording sites in June. However, of these twelve, five of them will dominate the most. For more information look at Appendix A3, which list how the observations were extracted and each of the bird species with which group they belong to. Among the most frequent, we have the *Bluethroat*, which is particularly interesting to look at as it likes to imitate other birds [71]. This can be a problem both when trying to label bird vocalisations or detect them. Even though no cases were found where the Bluethroat tried to mimic the Jack Snipe, several instances were found where it was believed to mimic the Spotted Redshank and Broad-billed Sandpiper. There are also several cases of other sounds and vocalisations that are quite similar to the three vocalisations. All of which can bring challenges for data labelling and for a model to predict them accurately. Figure 3.4

shows six sound samples taken from the data, where two of them shows the *Blue Throat*, while all the others belong to different birds.

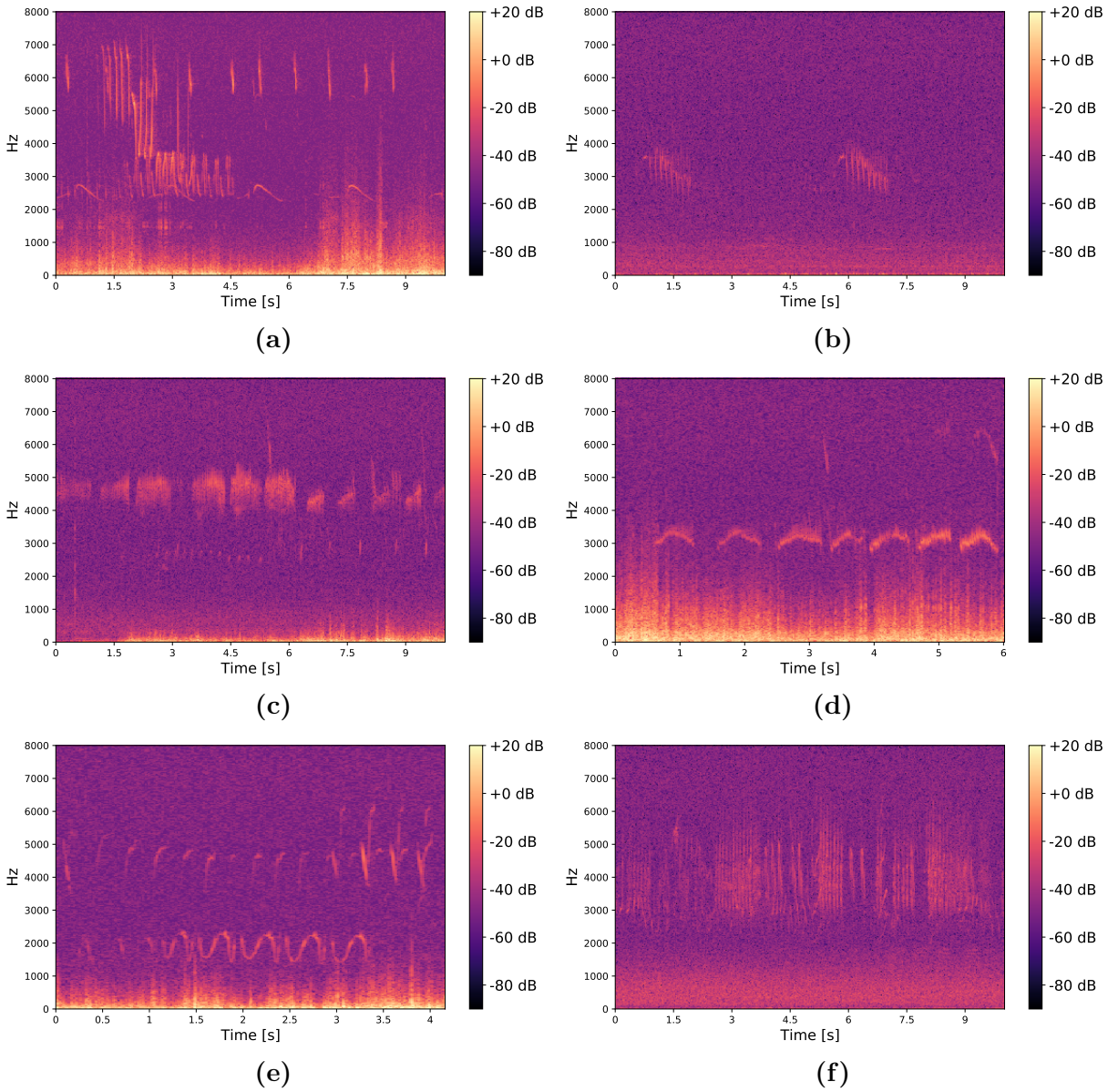


Figure 3.4: Six different sound samples taken from NINA's data to give insight to how the different soundscapes can look like. Figure 3.4a shows a *Golden Plover*, *Wood Sandpiper*, and *Meadow Pipit*. Figure 3.4b shows a *Willow Warbler* vocalising. Both Figure 3.4c and Figure 3.4d show two different examples of the *Bluethroat*. Figure 3.4e shows the *Ringed Plover* and Figure 3.4f shows a *Common Redpoll*.

The first figure, Figure 3.4a shows a medium noisy sound clip with the *Golden Plover* in the frequency range of 2.2-3.0 kHz, the *Wood Sandpiper* in 2.5-3.9 kHz, and what is probably a *Meadow pipit* between 3.5-7.5 kHz. Figure 3.4b, Figure 3.4c shows less noisy sounds clips. Figure 3.4b shows a *Willow Warbler* between 2.5 and 4 kHz, and Figure 3.4c and Figure 3.4d contains the Bluethroat. In Figure 3.4d, between 2.7 and 3.5 kHz, it is

suspected of imitate the Spotted Redshank or the Golden Plover (Figure 3.4a) which both have some similarities in there vocalisations. Figure 3.4e between 2.5 kHz and 2.3 kHz we have another vocalisation the bird *Ringed Plover*, which can also look a bit like the SR, but unlike the Golden Plover, it vocalises in a lower frequency range. The last sound clip in Figure 3.4f has a *Common Redpoll* between 2.8 kHz and 6.5 kHz, a vocalisation that can resemble the first of BS vocalisation presented in Section 2.1.

3.2.4 Signal to noise ratio for the vocalisations

Up until now, only strong versions of the vocalisations of interest have been presented. This section presents some weaker examples to give a better picture of the problem, and the variations of the strength of the signal. In addition to looking into how the level of background noise from the soundscapes presented in Section 3.2.3 affects the signals of interest, and the signal-to-noise ratio (SNR).

Let us start by looking at some weak signals of the birds of interest. Figure 3.5 shows a weak SR signal, Figure 3.6 a weak JS signal and Figure 3.7 a weak BS signal. All the vocalisations of interest are highlighted with white boxes.

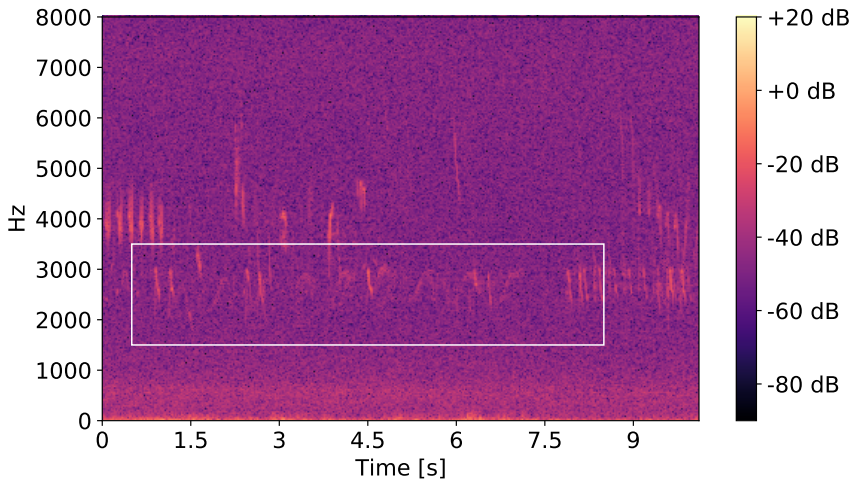


Figure 3.5: A weak Spotted Redshank mating vocalisation.

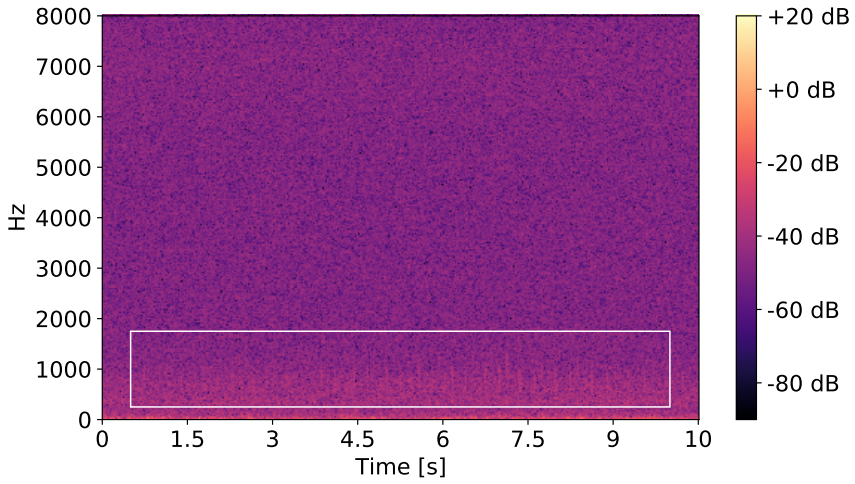


Figure 3.6: A weak Jack Snipe mating vocalisation.

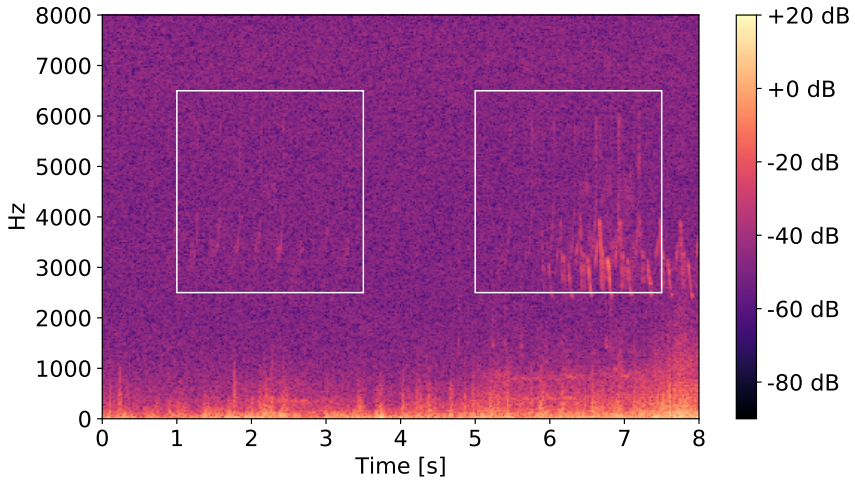


Figure 3.7: A weak Broad-billed Sandpiper mating vocalisation.

When comparing the spectrograms in Section 2.1, one can see that these are much harder to detect and that the SNR is lower for the weaker vocalisation than the stronger ones. The weaker vocalisations are likely due to the birds having a greater distance to the equipment when vocalising. However, as we saw in Section 2.2, there is more than distance that will effect how the sound propagates, and the fact that the birds are flying while vocalising the vocalisations of interest, as mentioned in Section 2.1, will probably have something to say as well. Something not illustrated in the figures above is that the vocalisations also can be much shorter, particularly when the signal is weak. When going through parts of the data, it was discovered that all three vocalisations had cases where a vocalisation lasted no longer than a second. Compared with the stronger JS vocalisations that lasted for 8-13 seconds, the SR for 3-7 second, and the BS for 2-6 seconds.

High levels of noise also leads to lower SNR, as mentioned in Section 2.2. Nevertheless, it is important to remember that it is noise with the same frequency range as the vocalisations of interest that gives the most trouble, effectively masking the vocalisations. When it

comes to the different sound sources introduced in Section 3.2.3 and the three birds, it was observed that the Jack Snipe is the most exposed to masking by wind and running water, which makes sense when comparing their frequency ranges. The Broad-billed Sandpiper and Spotted Redshank are much more vulnerable to being masked by other bird vocalisations, as most of them vocalise between 1.5-7 kHz, which is the same frequency range as the BS and the SR. Figure 3.8a shows an example where an the SR mate attraction vocalisation has a medium to strong strength but is masked by a Wood Sandpiper, making it much harder to detect. Figure 3.8b shows a sound clip of a powerful SR vocalisation between 2.4 kHz and 3.2 kHz, with a high level of noise, due to strong winds up to approximately 2 kHz and other bird vocalisations. Due to the strong SR vocalisation it is still not that difficult to detect, even with a lot of background noise.

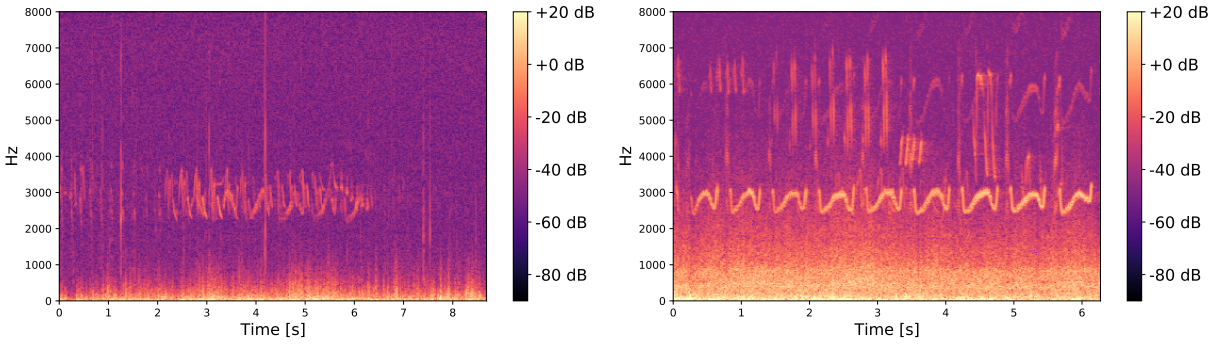


Figure 3.8: Two examples of SR vocalisations in strong level of noise. Figure 3.8a shows an example of where the SR is being masked by a Wood Sandpiper. Figure 3.8b shows a much more intense SR, with a lot of background noise.

An important thing to point out with the spectrogram in Figure 3.8b is that the shapes between 4.9 kHz and 6.4 kHz looks like harmonic replicas of the SR vocalisation between 2.4 and 3.2 kHz. This phenomenon was observed several times with multiple types of vocalisations in the soundscape, including the BS. The reason behind it was not determined.

If we recall Section 2.2, it was mentioned that some birds adapts to masking in different ways. Figure 3.8 illustrates one of many cases were the Spotted Redshank seems to vocalise stronger at times during high levels of noise due to wind and rain. This is the only one of the birds of interest that does this, and has been observed throughout the project. Even though it does not happen every time, it could be due to the Lombard effect (see Section 2.2) or it could be that many of the SR vocalisations are made at a closer range to the microphone. However, it is hard to say any of this for certain.

Throughout the project, it was observed that the Spotted Redshank's vocalisations could have a shifting frequency ranges, regardless of strength. Two examples we have seen up to now show this as the SR example in Section 2.1 had the frequency between 2.1 and 3.2 kHz, while Figure 3.8 had the frequency range 2.4 and 3.2 kHz. This could be due to one of the other masking avoidance techniques, presented in Section 2.2, called frequency shifting. It could also be due to the double effect but since this has not been observed for

any other species, it is more likely to be vocalisations made by different individuals having different sizes, as there is a correlation between the frequency range a bird vocalises in and their body size [48].

3.2.5 How to utilise each data source

As presented in Section 3.2, most of the data used in this project came from NINA, in addition to some examples from publicly available datasets. The number of examples from the public datasets were heavily outnumbered by NINA's data, making it worth thinking about how to best utilise this data. Using this data during training would result in a few more data points and a bit more diversity, but no way to "validate" the effect of including this data when training. Instead it was chosen to only use these samples when testing the model, where it could be used to say something about the model's ability to generalise on data from other sources than NINA. When only training on NINA's data, one could suspect the model to overfit, only giving good results with a specific equipment chain or in a specific type of soundscape. Using other soundscapes and other equipment chains when testing helps us to validate or invalidate overfitting. However, the quality of the public data is as mentioned in Section 3.2 of variable quality. It is also often prone to high noise levels, which is important to consider when looking at the test results. We will come back more to how the system will be tested in Section 3.9.

Due to the fact that NINA's recording was done in a total of eight locations over three years, the locations were carefully selected and divided into training and test locations. The reasoning behind this was to ensure that the test data would not come from the same locations as the data the model was trained on. Attempting to create a model that could support placing a microphone in a new location, without the need to retrain the model, as would be the practical use of this project for NINA. It was determined to choose three test locations, to ensure diversity in the testing. Ideally, there would be a lot of data used for testing, but labelling the data for testing is quite time consuming. Therefore, one day form each location was used for testing. An overview of how each location was used when training and testing is shown in Table 3.2.

Table 3.2: An overview of how the different locations were utilised, showing which location was used for training and which was used for testing.

Location	1	2	3	4	5	6	7	8
Training	x		x		x	x		x
Testing		x		x			x	

The reasoning behind each selected location and year differs. Some recordings includes variations in the equipment chain,and other recordings were used in the preliminary project, giving the possibility to directly compare how well this system is at detecting the JS compared to the system made in the preliminary project. The equipment chain is much more than just the microphone or the analog-to-digital converters (ADC), but everything between when the signal "hits" the microphone until it is digitised. All of which can contributes to small variations in the data recorded in the form of noise. This

means that there are quite small variations in the equipment chain to NINA, as we saw in Section 3.2.2. However it is tried to utilise the little variations that exist. Let us have a closer look:

Testing
The test location was selected first and was based on tying to use the diversity that was available in NINA's data. Different reasons which will be listed up in the following part:

- **Year 2018:** In 2018, NINA added a component to the equipment chain that was not used in the other years, a windshield. Instead of including this data as part of the training, it was decided to use data from 2018 to test the system. This is to see if the model would generalise, still giving good results if adjustments to the equipment chain were made, and new recordings were done another year than the model had trained on.
- **Location 2 and 4:** One day from each of these locations was processed and labelled in the preliminary project. It was chosen to use this data to test this project's model as well, to get a direct comparison of the quality of JS detection. This is valuable as there are no other known models that classifies this bird, effectively creating a benchmark for the JS classification in this project. Both locations are known breeding sites for both JS and BS, increasing the chance of finding the vocalisations of interest. This data was recorded in 2018.
- **Location 7:** One day from location 7 in 2017 was used. From a quick look at the data from 2018, none of the data had the Spotted Redshank vocalisation of interest, which according to the is probably due to the recordings were done later in June this year. Therefore was it particularly important that this day had a good amount of cases, which the day selected had. It is also important to have the fact that only one of the three test days holds SR examples, when looking at the results.
  Regarding the equipment chain used at location 7 in 2017, it was only this location and year, and location 6 in 2016 used another microphone in the equipment chain. Making this test day an opportunity to test with this change as well.
- **Known breeding sites:** If we look in Appendix A1 we can also see that by choosing the combination of the locations listed above, we also ensured that there are at least one known breeding site of each bird. As location 2 is a known breeding site of JS, location 4 is a known BS breeding site, and location 7 is a known SR breeding site.

Training
All the remaining locations were marked as training data, and used by choosing at least some days from each. This was so one would take advantage of the data collected, and that the recordings were carried out at different locations. In the following part will we list some comments to the test data used, while Appendix A4 lists up all the specific days and time spans selected from each location.

- **Location 1:** This data was already processed and labelled as a result of the preliminary project. This data included eight full days of data recorded in 2017, saving a lot of manual processing time in this project.

- **Location 6:** This location is a bit special as it really contains five "sub-locations". It is however viewed as one location by NINA, as they are close and the recording equipment was moved between them. In this case were one day from four of the sub-locations picked out.
- **Location 5:** This location was only recorded at in 2016 and were first thought almost only contain car noise. This was the case for about the first half of the data, however, the rest turned out to contain all three species in between the car noise. One can also notice that in Appendix A4 were two of the longer 18.6 hours recording from this site marked with being added after the original labelling was done. This was done as a measure to look for more 'BS' vocalisations when the data set were discovered to be unbalanced with 'BS' as the minority class. This we will come back to in Section 3.4.3 and Section 4.1.

3.3 Labelling of the data

The software Audacity[72] was used to label all the data (version 2.3.3 for ubuntu). It was chosen because it is a well supported and used free open source software, which had all the necessary tools needed. The tool offers the ability to import both mp3- and wav-files, and to view the audio as either a spectrogram or waveform while listening to the sound clip. The spectrogram view was chosen as it is not possible to spot the vocalisations in a waveform-format with the data in this project. But it is fully visible in a spectrogram view, as shown in Section 2.1 and Section 3.2.
Audacity also offers the possibility of zooming in and out on both the time and the frequency axis, and then highlight parts of the audio one wants to label. Figure 3.9, shows a screenshot of this software in use while marking and labelling a Jack Snipe vocalisation.

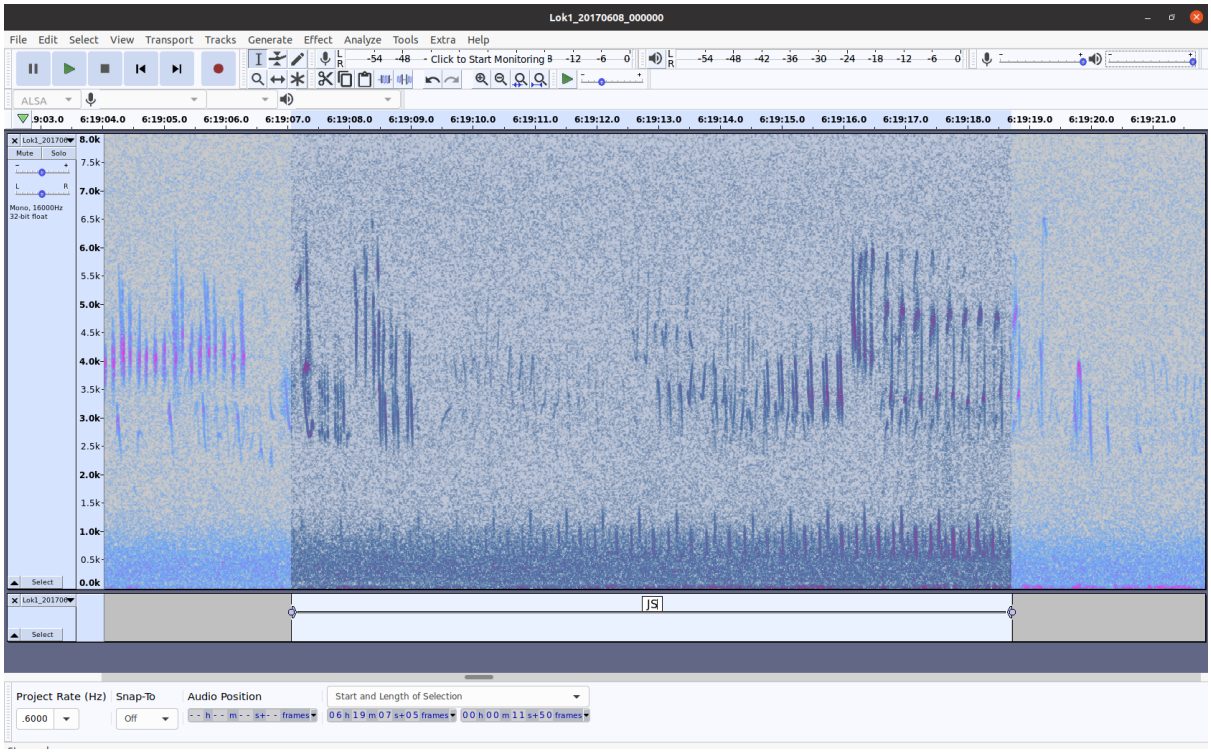


Figure 3.9: A picture of Audacity in use, while marking and labelling a Jack Snipe vocalisation.

An important thing to point out is that the time resolution and frequency range used to view the audio has a big effect, as it can make it harder or easier to spot something. A second by second time scale was used for the whole project, as shown in Figure 3.9, to ensure high visibility of weak or masked vocalisations. Using this resolution also made it easier to accurately label the vocalisations. When the labelling of an audio file was done, a method to extract the data was needed. It was chosen to extract the labels in txt-files, which included the labels and start/stop timestamps, given in seconds with a six decimal accuracy, of each vocalisation. How these txt-files were used to make the complete training and test datasets will be explained in Section 3.4 and Section 3.9.

The data consists of four classes:

- The Jack Snipes class: **JS**
- The Spotted Redshank class: **SR**
- The Broad-billed Sandpiper class: **BS**
- All data that was not the vocalisation of interest: **other**

All bird vocalisations of interest were labelled with the bird's abbreviation. When there were multiple vocalisations from the same species in the same sound clip, only one label would be assigned. If the sound clip included vocalisations from two or more classes, the sound clip would be labelled with each of the classes present. An important aspect of the

classification task is to tell when there is no bird in the sound clip. A selection of many different sounds were therefore assigned to the *other* class, for the training data. While selecting this data, the focus was to create a diverse and large set of examples of how the vocalisations of interest does not sound like. The idea was that the more examples from different sources was assigned to this class, the less prone the model would be to wrongly predicting a bird from some random noise. This labelling of the *other* class was only an important aspect when training the model. As when testing the model, and when used in "production", everything that is not a vocalisation of interest is considered to be the *other* class. This will be explained further in Section 3.9.

As the author of this report had no previous specific knowledge of bird-audio before the preliminary project, and that the project consists of a lot of manual labelling, it was important to get familiar with the data. The ornithologist John Atle Kålås helped by giving a short introduction on each vocalisation, as well as working as a "validator" in the start of the project, to confirm that the author was understanding what characterised each vocalisation of interest. After getting more familiar with the vocalisations the ornithologist only functioned as a second opinion, to discuss specific vocalisations when they were difficult to define. This resulted in a higher quality data-labelling process, but also made the process more time consuming. An important thing to consider when evaluating the quality of the data-labelling process is that the ornithologist was foremost familiar with the Jack Snipe's vocalisation, but was also somewhat familiar with the other birds as well. The Broad-billed Sandpiper turned out to be a challenge to identify in many cases, for both the author and the ornithologist, as the vocalisations were often weak and frequently masked by other stronger vocalisations. The vocalisations that were masked was particularly challenging to verify for the ornithologist, as the audio source masking the vocalisation of interest often would dominate the sound segment. When inspecting the sound clips for labelling, the author would foremost look at the spectrogram, combined with listening in difficult cases, while the ornithologist primarily listened to the sounds. Even though the ornithologist came with his opinions, it was the author of this project that was responsible to make the final decision on how to label the specific segment. Ideally one would be able to classify each vocalisation with a high degree of certainty. However, the method used for gathering data combined with manual labelling, as done in this project, will always introduce cases with some degree of uncertainty.

Another thing that is believed to have an impact on the quality of the data is that each day was inspected in two rounds, first to look for JS vocalisations, and the second time to look for the BS and SR vocalisations. A measure done to not miss any vocalisations, especially the weaker ones, as the JS is in another frequency range than SR and BS. After all the days were labelled, the training data's labels were double checked to see if they were correctly labelled. This was to reduce the chance of any mislabelling that could have taken place, as human labelling is assumed to have an error rate of around 5% [73]. To reduce the chance of any mistakes in the labels belonging to the test data, the data were often inspected when checking the results of a trained model in the system, especially when observing false-positive cases. After the test data was inspected and labelled for all three classes for the second time, all cases where a vocalisation was found during the

second iteration was given an extra tag. This was done so one could measure the error rate of the manual labelling process.
How many cases found per class, both used in the training and test data, will be listed in Chapter 4 where we will take a closer look at the dataset.

3.4 Pre-processing

In this section, we will look at how the data was pre-processed, both for training and testing. Figure 3.10 shows what the first part of the system in Figure 3.1 looks like when in production.

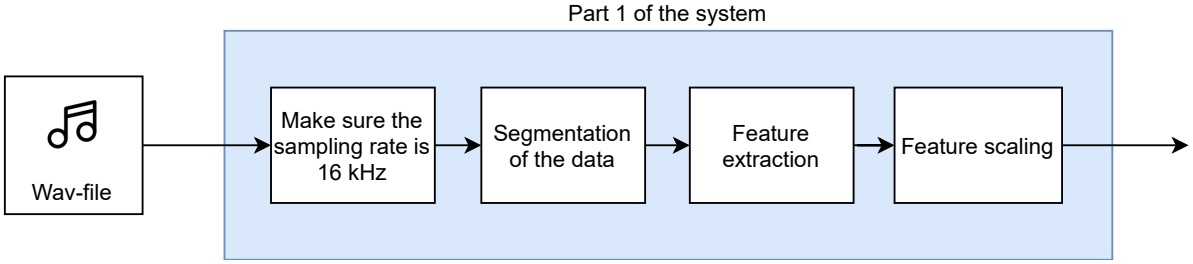


Figure 3.10: The pre-prossesing part of the system when in production.

The first step for the pre-processing component is to check if the wav-file has a sampling rate of 16 kHz. If not, it up-samples or down-samples the data to 16 kHz as mentioned in Section 3.1. The second step is to get the data in a standardised format with a specified given length, so that the data can be used as input in a neural network. As the input layer of a neural network has a fixed number of nodes to take in data. The vocalisations are therefore divided into smaller segments. It was determined to use one-second segments in this project, as it showed great results in the preliminary project, and accurate enough to catch some of the patterns in the vocalisations of interest. One-second segments, also enable us to say whether or not the bird was there that second or not. When dividing the audio into one-second segments in testing/production, the data segments were divided so each data-point had 50% overlap with the next data point. Why this was done, will be explained in the post-processing Section 3.6.

The third step for the pre-processing component as shown in Figure 3.10 is feature extraction. This part transforms data from standardised audio samples into new domains, extracting valuable information. How exactly this is done will be presented in greater detail in Section 3.4.2. The last step is called feature scaling, a technique introduced in Section 2.6 that scales all the data. This is done as ML algorithms has a tendency to perform worse when the data operates with different scales [53]. Standardisation scaling was applied on the data, which uses the mean and standard deviation of the training set to scale the data, by subtracting the mean value and then dividing it at by standard deviation [53].

Most of the steps in Figure 3.10 were done in the same way both during training, testing and when in production. However, since the training data came from many different files with corresponding labels, the data segmentation had to be done a bit differently. How this segmentation process works for the training data will be presented step by step in Section 3.4.1.

When it comes to the first step, we know that we do not need it, since all the training data will have a sampling rate of 16 kHz. However, it is done automatically when loading the data, which is why this step will remain. Between the segmentation of the data and feature extraction there is also added an additional step to look into how to scale the dataset, this we will come back to in Section 3.4.3.

3.4.1 Segmentation and extraction of the training data

The process for labelling the training data was presented earlier in Section 3.3. The next step is to extract data and segment the audio-clips correctly, so that it can be used to train a machine learning algorithm. Figure 3.11 shows an overview of how the audio and labelling files are used to extract the training data.

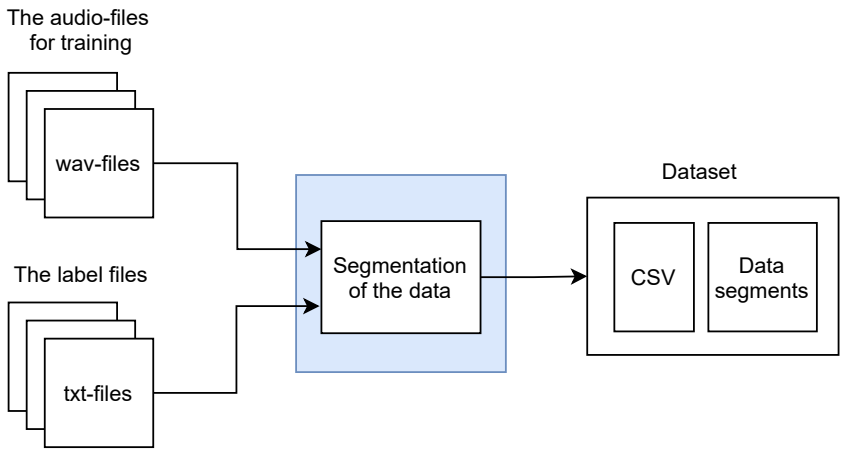


Figure 3.11: The segmentation block to extract training data from the audio files.

A simplified description of the process is that the segmentation block takes inn all the labelled wav files and their corresponding txt-files, iterates over them and outputs segmented data and a CSV-file. The CSV-file functions as a record, storing the file path for each segment and its label. Let us have a closer look on how this is done.

As seen in Section 2.1 and Section 3.2.4, the vocalisations has varying lengths. The timestamps of each label were given in seconds with a six decimal accuracy, as mentioned in Section 3.3. These timestamps were rounded off to one decimal, as it was assumed to be accurate enough and simplifies the segmentation process.
However, since these timestamps often does not add up to whole seconds, we must find a way to process the data so that it fits with our selected segmentation size of one second. This can be done by adding background noise to the segments, or by removing parts of the vocalisation.

For the bird vocalisations of interest, the length was rounded up by increasing the size of the data, to make sure that no information of interest was lost. This was done by extracting the sound with some of the background noise from right before or after the vocalisation in the audio clip. Making sure enough background noise is included so that the vocalisation plus the background noise adds up to whole seconds. Figure 3.12 illustrates how this was done.

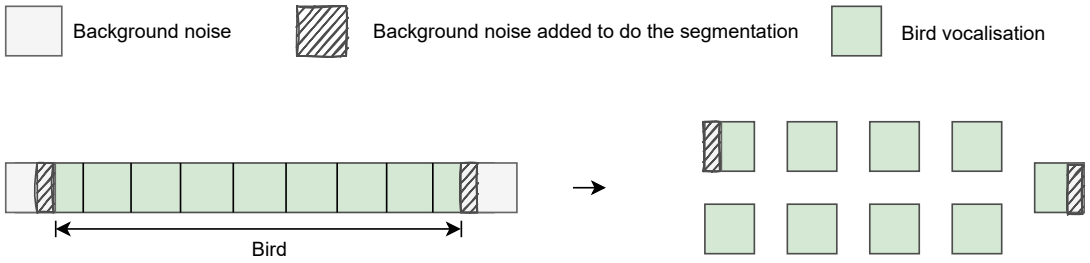


Figure 3.12: A demonstration of how the bird vocalisations of interest would be divided into one second segments.

In a scenario where the vocalisation did not add up to one second, X number of 0.1 second of noise would have to be included. If it was an even number, it would be divided equally on each side of the label. If there was an uneven number needed, the remainder would always be added to the start of the vocalisation.

However, the segmentation process illustrated in Figure 3.12 can not be used in all scenarios. What happens if the added background noise isn't noise at all, but is instead the end of another vocalisation? Or what if two individual birds vocalise simultaneously? These are cases that also had to be considered. It was decided to make sure there would be no overlap between two data-points in the training data in this project. From Section 3.3, we know that if two birds from the same species vocalise simultaneously they would share a label. However, if they are two separate species, we have to decide how to segment the data and extract the vocalisations correctly. Figure 3.13 shows an illustration of how two simultaneous vocalisations were segmented.

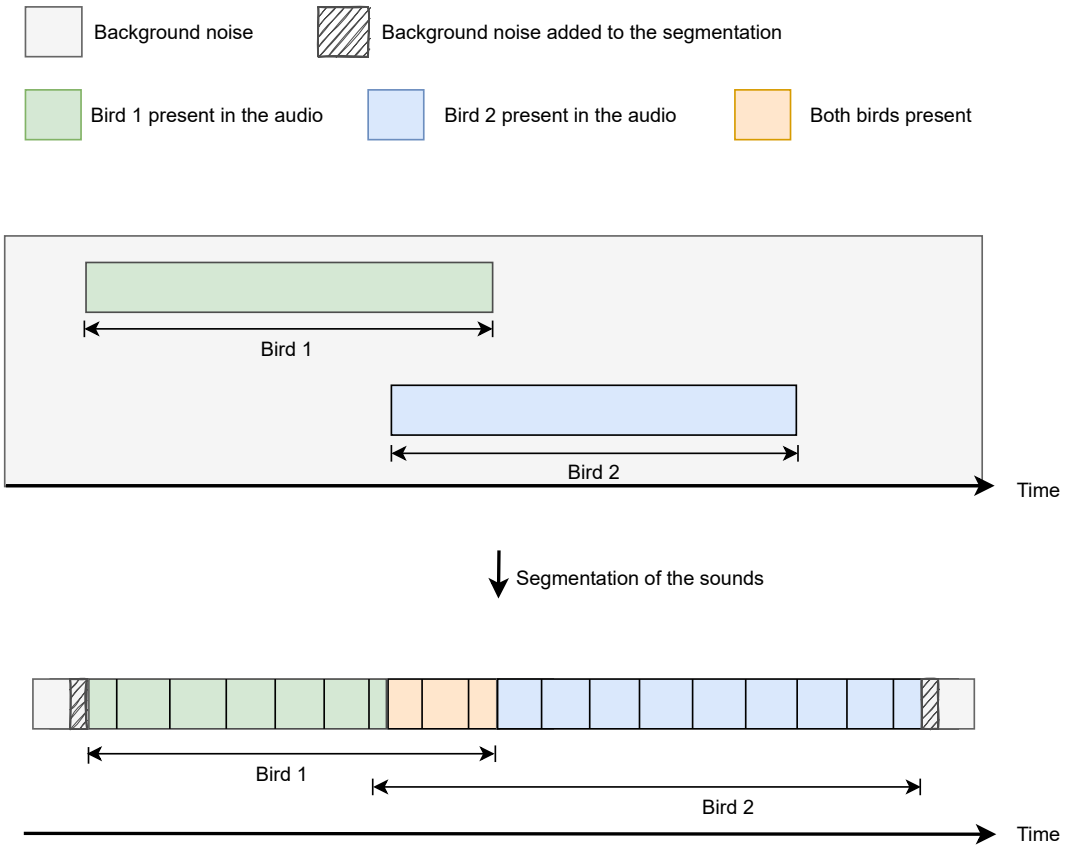


Figure 3.13: An illustration of how two different birds of interest vocalising at the same time were handled.

As the figure shows, the simultaneous vocalisations were not looked at separately but as a continuous audio clip. One would start counting when the first vocalisation started, and if that vocalisation ends with an overlapping vocalisation, the end timestamp is set to when the last vocalisation is finished. Then, if the length did not add up to whole seconds, extra sound from the original wav-file would be added to segment the data correctly. At the part where both birds were present, a check of how many percentages each bird was present in the one second segments would determine how to label each segment, which we will go into details about later in this section.

Then we have the scenario where two vocalisations were so close in time, that including background "noise" when segmenting a vocalisation leads to an overlap between the two vocalisations. This is particularly the case for BS and SR, as both of them were found to have many cases where their vocalisations are less than half a second apart, but still labelled separately. To avoid getting too many cases where sound segments overlap, all vocalisations that were 0.5 seconds or less apart were joined before starting the segmentation. This reduced the number of overlapping cases, but there would still be a few cases where adding additional data lead vocalisations to overlap. Figure 3.14 illustrates how these remaining cases were solved.

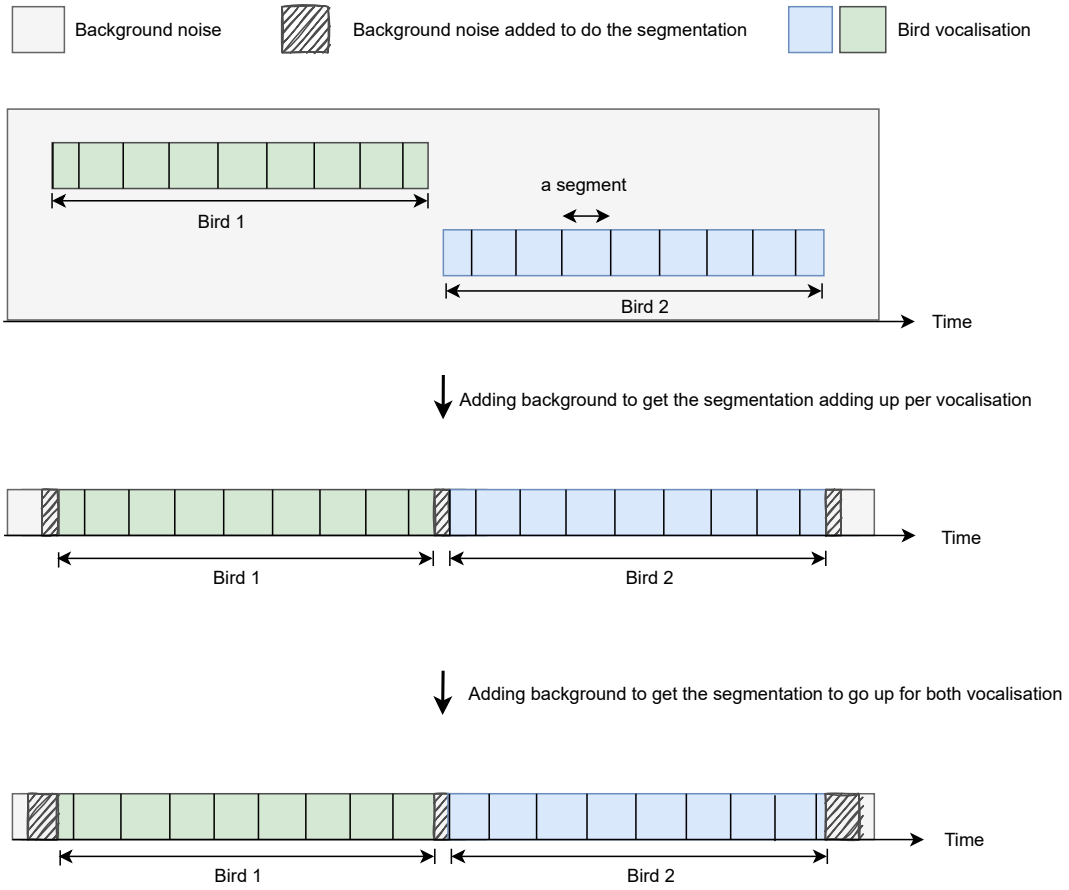


Figure 3.14: An illustration of how two close audio clips to be extracted could overlap because of the audio extraction method used and how it was handled.

The figure first shows two vocalisations that needs additional background noise to be divisible into one second segments. Then after doing this, the sounds to be segmented starts to overlap and are "joined" / viewed as one vocalisation sequence to be segmented. If they had been joined and did not overlap, no additional actions were needed as they would add up to whole seconds. However, in the cases where they do overlap, noise was added, as the figure illustrates. The last step is repeated until all the data fits into one-second segments. As one can see, and probably imagine, this can lead to some of the segments not having that much vocalisation left. In these cases, the same thing will be done as for two bird that vocalises simultaneously. To handle all cases like these, a check will be done at the end when the actual extraction of each data point is done and their labels are written to the CSV file. The check checks how much of the data was labelled with a birds vocalisation. If 0.25 seconds or more of a one-second segment were labelled with a bird before the label were rounded up to one decimal, the segment/data point would be labelled with that bird.

Now, let us look at what happens with the audio marked with the label 'other', illustrated in Figure 3.15.

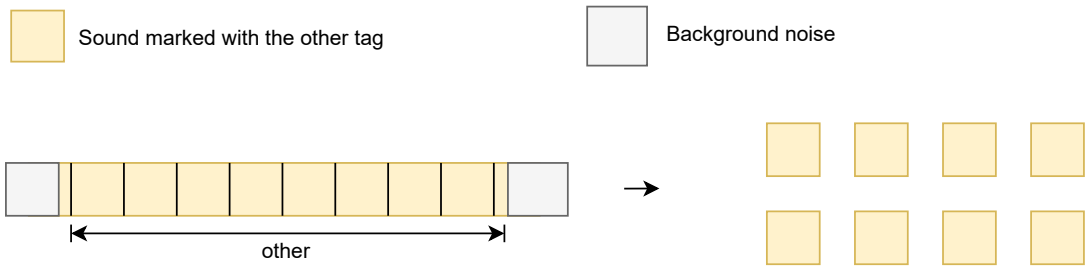


Figure 3.15: How the data labelled with *other* was segmented.

After handling the birds labelled data as described up to now, one would check that none of the *other* segments overlapped with the birds. If they did, that part of the **other** label would be removed. These sound clips would be rounded down, instead of up, at each side to make them fit into one-second segments.

All the sounds were extracted so that all segments would be put in separate wav-files at the end. Storing the filename, path and the belonging label for these files in a csv file, as illustrated in Figure 3.16.

Filename	JS	SR	BS
...	0	0	0
...	1	0	0
...	1	0	0
...	1	0	1
...	...	...	...

Figure 3.16: How the CSV-files holding the multi-labelled labels and the path to data points looked like.

The '0' means that a bird is not present, while '1' means it is present. If all birds have the label 0, this means that the data segment belongs to the 'other' class. An example of the 'other' class can be seen in the first row in the figure above.

Having processed and segmented the data, the next step is to find and extract features of the data that enables us to differentiate between the birds, and the background noise.

3.4.2 Feature extraction

Looking back at the system overview in Figure 3.10, the next block is *feature extraction*. This section explains how this block works both during training and testing. Now that the data is processed and segmented, it is in a format that could be fed into a neural

network. However, we can also do some form of transformation of the data, to extract valuable features from the data. Feature extraction can take many forms, and can consists of several steps. Examples of well used features in audio are: MFCC, Mel-spectrograms or spectrograms, as we saw some examples of in Section 1.2. It is also possible to use techniques such as filtering (low-pass, band-pass, band-stop or high-pass), noise reduction, and attention models to say what portion of the data to concentrate on, either in addition or alone.

In the preliminary project, one second long spectrograms with the frequency range 0-2 kHz were plotted and saved as RGB PNG-images (Section 1.2.1). There were two reasons why spectrogram images were used in the preliminary project: It was thought out that if a human could recognise a vocalisation through visually inspecting the pattern in a spectrogram, perhaps a neural network could recognise it as well if it was fed spectrogram images. Especially, if the network was a pre-trained network trained to recognise images, such as ResNet [26]. The second reason, which also supported the first one, was that spectral features were often used in similar projects, as we also have seen in Section 1.2.

It is possible to use the same procedure as in the preliminary project, if one adapts the frequency range to fit for all three vocalisations. However, plotting and saving images for each data point is inconvenient, and time demanding. It was not mentioned in Section 1.2.1, but the preliminary project report showed that the pre-processing took much more time than any of the other parts of the system because of the plotting of the PNG-images [26]. The system used a total of 3 hours and 40 minutes per day, where 2 hours and 33 minutes of that was from the pre-processing step. A time that is not terrible, but has potential to be improved through this master thesis. Because of the slightly inconvenient and time consuming feature extraction method used, it was recommended in the report as potential future work to look into other potential solutions and compare them.

For this project, mainly three different data formats were tested: spectrograms in RGB images (as done in the preliminary project), pure spectrograms, and Mel-spectrograms. The reason for continuing plotting spectrogram and saving images, even though it is considered a bit inconvenient and time consuming, was that it was a good benchmark for comparing results. The two others were a hope of getting a faster and better feature extraction process, as well as continue using features that has shown to work in related projects. Spectrograms, and Mel-spectrograms are quite similar. The only thing separating them is that Mel-spectrograms are spectrograms where the Mel-scale is applied, which takes less space/has a smaller size. However, as we do not know what will give the best results both features were tested.

When testing with RGB images, several frequency ranges were tested, but mainly around 0.45 kHz to 6.0 kHz. It was also tested with different window sizes, which was mainly motivated by two things: The fact that the BS vocalisation could sometimes look a bit "blurry" when having a bigger window size than a smaller one, and that within speech recognition it is recommend with a window size of 20-40ms [74]. Speech and bird

recognition are not the same, but there can be drawn parallels between the two problems, such as detecting rapid changing sounds made by an animal's vocal tract.
When it comes to Mel-spectrogram, both a frequency range of 0.45 kHz and 0.5 kHz to 6 kHz were tested, as well as the default frequency range 0.0-8.0 kHz. When it comes to the "pure" spectrograms, a Butterworth bandpass filter of the eight order was tested for this purpose using the cut off frequencies 0.4 kHz and 6.5 kHz.

Using a process of trial and error resulted in two of the methods mentioned to be proposed as possible final solutions for feature extraction:

- The Mel-spectrogram with a window-length of 34.5 ms, using 116 Mel-frequency bins, FFT of 552, a hop-length of 138, a minimum frequency of 0.5 kHz and a max frequency of 6.0 kHz, which all gives a shape of 116×116.
- The Spectrum without filtering, with window length of 27.9 ms, using a FFT of 446 and a hop length of 73, giving the shape of 224×220 after transforming the one second audio.

3.4.3 Balancing the training set

From early on in the project it was suspected that there would be an imbalance between the classes in the training data. Something that was confirmed when the training data was extracted for the first time and the number of data points per class was checked. The exact numbers of each class will be specified in Chapter 4, as the dataset is viewed as a result of this project, while this part will focus on what was done to even out the imbalance in the data.

From the four classes there were most examples of the 'other' class, the 'JS' had the second most, the 'SR' the third most, and the 'BS' the lowest number of examples. Both the 'JS' and 'SR' classes had considerably more data than 'BS'. As mentioned in Section 2.10 an imbalanced dataset is not necessarily a problem, but it can be. Therefore, it was decided to test different techniques that would make the dataset more balanced, to check whether it would give better results or not. As mentioned in Section 2.10, there are several measures that can help with this. One of them is to add more data of the minority class, which was done by going through two additional 18.6 hour long sound clips, as mentioned in Section 3.2.5. Even though it evened out the number of examples somewhat, it was far from enough. It was therefore decided to try another technique, also presented in Section 2.10, called re-sampling. This can both entail up- or down sampling, as both would be able to balance the data more. In this project, both up-sampling and down-sampling was tested.

When down-sampling, it was decided to removed existing samples from the majority classes 'other', 'JS' and 'SR' at random. It was tested by:

- Removing enough to have the same amount of samples per class (25% of the whole

dataset per class).

- Having more datapoints in the three majority classes, than in the BS class, but balancing them and having less data in all of them.

When up-sampling, one could have just duplicated the 'BS' samples and some of the 'SR' samples, but this makes the model more prone to overfitting [66]. Instead, it was decided to look into oversampling using data augmentation, which would increase the number of data points by duplicating the data and modifying them in a realistic way, creating realistic versions of the data [53]. When applying data augmentation to balance the dataset it is often referred to *Synthetic Minority Over-sampling Technique* (SMOTE) [64]. However, data augmentation can also be used to increase the number of samples in a dataset. Example of different data augmentation techniques that can be applied on audio is noise injection, time shifting, time stretching, pitch shifting, mix sounds, volume turning, and filtering(highpass, lowpass, bandpass or band-stop) [75, 76, 77]. Even though multiple of these could have worked, it was decided to test the three listed below:

- *Adding noise*
- *Time shifting*
- *Mixing two audio segments*

All three were chosen as they seemed to be well used and fitting for the problem. They were done after the audio and prior to feature extraction mentioned in Section 3.4.2, as mentioned in the introduction of this section. The adding of noise was done by adding a uniform distribution of noise between [-0.001, 0.001] with zero as the mean. The time-shifting was done by shifting the audio with a random length between 1% and 99% of the audio's length to the right. The last method, called mixing, was done by taking two sound clips from two different classes and adding them together in the matter shown in Equation 3.1, where a is a number between 0 and 1.

$$mixed\_audio = a * audio1 + (1 - a) * audio2 \tag{3.1}$$

To set the labels for these sound clips the sound clip called "audio1" would get a multiplied with it's label, and "audio2" get *(1-a)* multiplied with its label. This means that the bird would get either a or *(1-a)*.
The mixing was done with all the combinations of the three birds, and by mixing either 'BS' or 'SR' with other. When mixing two birds a would be set to 0.5 in Equation 3.1, so half of both sounds were added, and both labels would be set to 0.5. For mixing 'BS' or 'SR' with other sound clip, both a=0.5 or a=0.4 were tested, where the bird would be *audio1* in Equation 3.1, giving the data point the label a.

3.5 Defining NN structure

In this section, we will look at the second block of the system, presented in Figure 3.10, and what it entails in testing or production. Section 3.8 will supplement by looking at what was done to train potential models.

There are a lot of different neural network structures out there. Some of the categories are feed forward neural networks (FFNN), Convolutional neural networks (CNN), Recurrent neural networks (RNN) and Long short time memory (LSTM), which again have many combination of network structures and ways they can be initialised. In the preliminary project, the pre-trained model called *ResNet18* was used, in combination with RGB-images as described in Section 1.2.1. This model was as mentioned in Section 3.4.2, chosen as it was thought to work well on images of spectrograms. The original model structure was as mentioned in Section 2.7.4 built by Kaiming He for the ILSVRC ImageNet challenge, which he won using ResNet in 2015 [11].

In this project it was determined to focus on pre-defined CNN structures, which one could adapt using transfer learning as explained in Section 2.8. This was done for several reasons: It have shown to give good results in other projects, including the preliminary project, both for images and other features such as spectrograms and mel-spectrogam [11, 26, 33], which was tested and used in this project. The other main reason is that there is already so many other thing one can tweak and change in a project, such as in the pre-processing step of the data, and different training parameters and techniques. Selecting a pre-trained network utilises work already done by often highly skilled people, specifically designed to extract a type of information, potentially saving time and giving a better overall result.

Before we talk about the specific models that were looked at, we will talk a little bit about the programming language of choice and machine learning framework that was used to build the system, as it is relevant for the following part. The software used will be explained in greater detail in Section 3.7. To implement the system, the programming language *Python* was used.From Python, there are two big and well supported machine learning libraries call Tensorflow [78] and PyTorch[79]. PyTorch was chosen in the preliminary project, and this project continues to use the same due to its popularity in research.

Two types of structures were mainly focused on and tested in this project: *ResNet* and *EfficientNet* [11, 80]. They were both tested with different sizes, with and without pre-trained weights. The ResNet structure comes in several sizes (152 , 101, 50, 34 and 18 layer). As a part of the PyTorch project, the library *torchvision* enabled one to download certain models, among others; ResNet with 18,34,50, 101 or 150 layers. The other model, *EfficientNet*, was developed by Mingxing Tan and Quoc V. Le, who submitted a paper on it in 2019 [80]. They had developed a model that demands much less parameters to achieve the same top-5 accuracy on the ImageNet dataset than many other networks, including the ResNet models (152 , 101, 50, 34 and 18 layer) **??** [11]. Therefore it seemed

fitting to test EfficientNet as well. However, EfficientNet is not, at least yet, a part of PyTorch's model library and was therefore downloaded through the much used python library; *efficientnet_pytorch*, which we will come back to in Section 3.7.

After adapting both structures using transfer learning (see Section 2.8), testing with different sizes of the structures, using both pre-trained and random initialised weights with both features; Mel-spectrogram and spectrogram, the best results were achieved by using Pytorch's ResNet34 model with pre-trained weights. Figure 3.17 shows an overview of the model's structure used for both features, which we will go through step by step in the rest of this section.

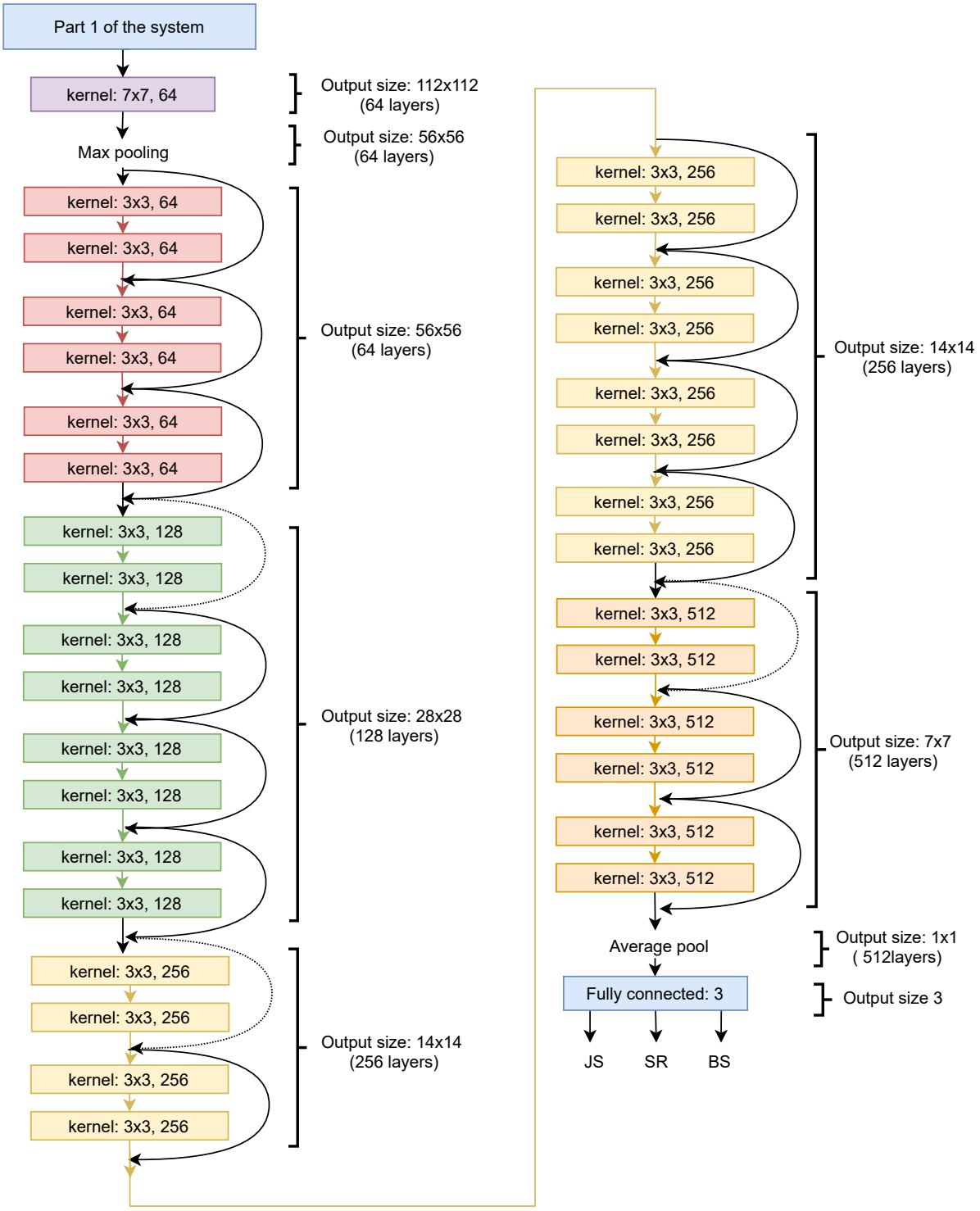


Figure 3.17: A block diagram of the model structure used in the system.

The first layer of the the original model was a convolutional layer that took in images with 3×224×224 (channel×height× width) input. It used a 7×7 kernel with two in stride and three in padding to get the output size of 112×112×64, where the latter number is the number of feature maps 64. For both the Mel-spectrogram and spectrogram this layer was adapted so it would take in the input shape defined in Section 3.4, and give out a 112×112. For the Mel-spectrogram which had the shape 1×116×116, the first layer used the same kernel size, but was adjusted to only take in one channel and changing both the padding and stride to one. For the spectrogram with a shape of 1×224×220 the first layer only takes in one channel and only pads with one at the longer side and three on the shorter. Otherwise it also used the 7×7 kernel, with two in stride to get an output 112×112×64. After this the output of the first layer was taken into a max pooling layer with a 2×2 kernel using a stride of two, giving out an output of 56×56×64.

The successive 32 layers are also convolutional layers. However, unlike the first layer, they are all residual blocks (see Section 2.7.4) with two layers per block, or a skip connection and a kernel size of 3×3. As Figure 3.17 shows, they are all divided into colours after which output size they have, meaning width, height, and the number of feature maps. To maintain the same output size, both the stride and padding size is set to one. When downsizing the width and height and increasing the number of features maps the stride is set to two, and the number of kernels is doubled. The skip-connections or residual connections that have dotted lines in Figure 3.17 symbolises a change in size, which is achieved by changing the stride from one to two.

The first set of three residual blocks in pink in Figure 3.17 both takes in the input size and give out the output size of 56×56×64. Followed by four residual blocks in green with the output size of 28×28×128. Then, a total of six residual block in yellow give out an output of 14×14×256 before the last set of six residual blocks in orange in Figure 3.17 give out is a output of 7×7×512. After this, an average pooling layer is applied to get the output 1×1×512, which is then flattened. The last layer is a fully connected layer. It originally took in 512 and gave out 1000 outputs, but is adapted to give out 3 classes: one for each bird as Figure 3.17 shows. Since this is set up to be a multi-label classification problem, the outputs are independent of each other. Meaning each node is basically a binary classification, where we want it to say '1' for yes and '0' for no for whether that class is present. However, the models output will not automatically be '0' or '1', it must first be scaled between zero and one, and than processed in the post-processing step, which we will come back to. To scale the fully connected layer's output between 0 and 1 the activation function Sigmoid which was presented in Section 2.6 is used. However, Sigmoid is not the only activation function used in the model, there is also ReLU, presented in Section 2.6, which is used between the convolutional layers. In addition, what is called a batch normalisation was used before each ReLU. Batch Normalisation proposed by Sergey Ioffe and Christian Szegedy in a paper in 2015 [81]. It normalises the data in mini-batches through standardisation, which act as a regulator addresses the problem of vanishing/expedient gradient [53]. This was not the only regulation technique, which we will come back to in Section 3.8 when talking about training. First, we will look at how the model's output data was processed in production/testing and what software and

hardware was used to built the system.

3.6 Post-processing of model output

This section will look at the third and last block of the system in Figure 3.1. As mentioned in Section 3.1, this part only exists in testing or production, and is responsible for taking the models output, processing it and returning the CSV-file listing of potential vocalisations found by the model used with start and end timestamp.

In the preliminary project, the model had two outputs; one for the 'JS' and one for the 'other' class. Since only one of the outputs could be correct per datapoint, the class with the highest value was set to belong to the input [26]. After this, all the vocalisations the system believed to have found were checked for whether they lasted for three seconds or longer by having three data points after each other, as mentioned in Section 1.2.1. If so, they would be "accepted" as vocalisations and be added to a txt-file with timestamps. In this project, the model and problem is set up as a multi-label classification problem, meaning that each output node belongs to a class/bird and is independent of the other classes as all or none of them can be active at the same time, as mentioned in Section 3.5. This suggests that the first method used to process the model's output in the preliminary project would not fit this problem, as the output of each class is independent. Instead, an "acceptance" threshold was set for each class to say whether one of the birds were present or not.

When it comes to the next step of the system, the preliminary project check for a three-second duration, which worked great as most of Jack Snipes vocalisation are a bit longer and it removed a lot of false positives. However, the BS and SR are often a bit shorter, as mentioned in Section 3.2.4, which is why a three-second threshold would not necessarily work that great. It should also be mentioned that the number of false negatives in the preliminary project was almost exclusively lower than this, which is why they became false negative. Therefore it was chosen to think slightly differently but still not drop the concept used in the preliminary project. The system will still check for a number of X data points in a row that belongs to the same bird class, but instead of the data points having no overlap they will have 50% overlap with the data segments next to them, as mentioned in Section 3.4. Doing this enables the system to check for multiple positive outputs in a row per class, but for smaller time intervals than the method used in the preliminary project allowed. Nevertheless, even though such a check eliminates some false positives, one will risk losing some true positives. The same goes for where the threshold of acceptance is set per class. It will often be a trade off between true positives and false positives for both cases, resulting in several tests with multiple thresholds and durations.

3.7 Software and hardware

In this section we will we look at the software and hardware used to build the system. A few months into the project a Nvidia GPU (GeForce RTX3090) came at the projects disposal to speed up the systems processing time by using CUDA (version 10.1.243-3). As mentioned in 3.5, *Python* version 3.7.10 was used, together with the library management anaconda (version 4.9.2). The code was written in the editor Jupiter notebook (version 6.3.0). All the libraries downloaded for this project are listed together with version number below, as long as it is not one of pythons standard libraries.

- Librosa, version 0.8.0
- Matplotlib, version 3.3.4
- Torch, version 1.8.0
- Torchvision, version 0.8.0
- Efficientnet_pytorch, version 0.7.0
- Numpy, version 1.19.2
- Pydub, version 0.25.1
- Sklearn, version 0.24.1
- Scipy, version 1.6.2
- Pandas, version 1.2.3
- Os (part of pythons standard library)
- Random (part of pythons standard library)
- Copy (part of pythons standard library)
- Time (part of pythons standard library)

The main rule when downloading them was to use anaconda if possible, if not available the python install manager (pip) would be used. The version of each library can be important both for running the system code in the future, but also if one wants to reproduce the result at any time, as PyTorch do not guaranty reputability across versions and using different hardware [82]. However, to reproduce the results, the datasets and code would also be needed.

Librosa was used to both read the wav-files, and plot the Mel-spectrogram and the ordinary spectrogram. It was also used together with *Matplotlib* to plot all the spectrograms in this report, which was also used to make the PNG images. Pytorch's libraries *Torch* and *Torchvision* is used for most of what have to do with making and training the models, in addition to some of the pre-processing and CUDA. However, as mentioned in Section 3.5, *Efficientnet_pytorch* was needed to download and test Efficientnet. The library *Numpy* was used for a lot of the smaller mathematical operations, *Padas* was used to deal with the labels and some more, and *Os* was used to access the data files. Then we have *Pydub* that was used to transform the mp3 files in the test data to wav files, while *Scipy* was

used to test Butterworth filter (see Section 3.4). *Random* was used to set one of several random seeds, which we will come back to Section 3.8. It will also explain why we needed *copy* to make a deep copy of the model while training. The last library *Time* was used to time the computation time, both during training and testing.

3.8 Training the model

This section will go through what was done to train the neural network structures into models described in Section 3.5. When doing so, some of the concepts explained in the background chapter will not be explained in great detail.

For this project, a GPU was used to speed up the training and testing process of the system, as mentioned in Section 3.7. However, what we need to know is that it enables us to use CUDA and the cuDNN library, significantly reducing the time needed to train and test a model. First of all, to ensure that any model could be reproduced, all the random seeds and flags the code depend on were set, following the instruction for reproducibility provided by PyTorch [82]. This is done so the "random" numbers produced will always be in the same sequence.

Before starting to train a model, the training data was extracted with labels as explained in Section 3.4.1. The data was then divided into a training and validation dataset by randomising and splitting the data into a 70%-30% split(training-validation data). It is often common to split the training data in such a way that one can use the training data to train the model and the validation set to validate the results per epoch. An important thing to emphasise here is that the validation data is not used to train the model, but merely used as an indicator to validate if the model is actually improving. By having a smaller dataset the model has not trained on to test on, one will get a better sense of how the model works on new data, which can thereby help choose the best model. It can also help adjust the model's parameters and look for signs of over or underfitting.
If the dataset was down-sampled, it was done before the data was split into two sets, while if it was upsampled, it would be done after, so the validation set would not contain any of the data points used for data augmentation.

To upload the data a dataset class was defined, which would pre-process the data in a correct manner before it was uploaded into a dataloader. The dataloader determines, among other things, the batch size and number of workers used. Even though it was tested with both a batch size of 8 and 32, most of the training was done with a batch size of 16. These sizes were chosen based on the paper "Revisiting small batch training for deep neural networks" written by Dominic Masters and Carlo Luschi [83], which showed how they got the best performance by using a batch size between 2 and 32.

Now, let us focus on how the neural networks were set up. As mentioned in Section 3.5, both the ResNet and EfficientNet structures were tested with and without pre-trained

weights and different sizes/number of layers. This was done by loading the structures with a given size, either with or without weights. If we recall from Section 2.8, some of a pre-trained model's layers may be frozen. In this project, the models were "unfrozen" while training, meaning that all the weights in the model will continue to be adapted through the training, instead of training only some layers (see Section 2.8). After the models were loaded, the model's first layer would be redefined if needed, depending on the input shape of the pre-processing step. On the other hand, the last layer would always be redefined to having three nodes, one for each bird class. When the last layer was redefined, the popular regulation function called dropout was also added, which is meant to decrease the chance for overfitting and increase the accuracy in deep neural nets[58, 84]. The "dropout" technique is only used during training and will "drop" or ignore a set of random nodes per batch in the layer it is applied in, which helps the model not become too dependent on some nodes [53]. How many nodes are "blocked" will be determined by a parameter called the dropout rate, p, which is typically set between 10% and 50% [53]. Another measure done in the hope of improving the model was to initialise the last layer with what is often referred to as the *Xavier initialisation*, recommended when using the Sigmoid function to prevent its output from saturating and speed up training [53, 61].

When training the model, it was first decided to use the loss function *Binary Cross Entropy*, called BCEloss in PyTorch [85]. However, as PyTorch recommends using *BCEWithLogitsLoss*, which combines Sigmoid and BCELoss in a more numerically stable way than using them separately, this was used instead. When it comes to the optimiser used, both Adam and SDG were tested, with several learning rates ranging from 0.01 to 0.0001 and various weight decays between 0.1 and 0.0001. In addition, SGD was tested with a momentum of 0.9. The learning rate scheduler *StepLR* was used to adjust the learning rate during training, which decreases the learning rate with a number called gamma, per X epoch. The scheduler was tested with different gammas and different numbers of epochs, X. The gamma was mostly set between 0.1 and 0.5, and the number of epochs 3-6.

The training was done over a variable number of epochs ranging from 15 to 50. Each epoch started by iterating over the batches of training data. For each batch of data, the loss would be calculated, and the optimiser would be used to improve the model through backpropagation. While iterating over the data, both the loss and accuracy would be calculated for the training set. After all batches had been processed, the validation data would be iterated over to calculate the loss and accuracy for the epoch. Sklearn's function accuracy_score [86] was used to calculate the accuracy. The accuracy score does not consider the case where only some of the labels per data point are correct, but whether all are correct or not. The reason for using accuracy score as a metric is so so both the loss and accuracy of each epoch during training can be used to plot how the model's training progresses. These graphs will be looked into for the final model in Chapter 4.

The validation loss was used to determine the best model. To get the best model, a deep copy of the model's parameters was taken before the training started, and updated at the end of an epoch if the validation loss got a new minimum for the whole epoch. Then,

after all the epochs, the deep copy of the best model parameters would be uploaded to the model. This technique, where you either automatically or manually stop the model when the validation data has the lowest loss, is refereed to as *early stopping*. This is typically used to prevent the model from overfitting [53]. This method for training is a well known and used way of training a model. The way of training described up to now was considered the first way to train the model. However, a second method was also used, where one would use everything up to now but also update the model with the best parameters at the end of every X epoch. This would be when the learning rate scheduler decreased the learning rate, instead of just saving these parameters for later use. After training the model, the model's parameters would be saved, by using PyTorch's save function that stores the model parameters in a zip file [87]. By doing this it enables us to use the model later by loading the model's parameters on to a model again, so it is ready for use.

3.9 Testing

This section will look at how the different versions of the system were tested. A total of three tests were set up for the different versions of the system: Two of the tests were based on the test data from NINA and one on the publicly available data found. However, only the two tests using NINA's data were used to determine the best models (one using Mel-spectrograms and one using spectrograms). The last test that only uses publicly available data was used to check how well the system would generalise new data from other sources and soundscapes. As mentioned in Section 1.1, this project does not focus on how accurate the system is to predict the start and the end of vocalisations, but rather whether the final system can find the vocalisations at all. Therefore, the number of true positives (TP) and false negatives (FN) is not based on the number of data points predicted correctly, but on how many vocalisations the system is able to find. The number of true negatives (TN) and False Positives(FP) will, on the other hand, be calculated from the number of data points. Nevertheless, since NINA specifically expressed that having a low number of false positives was vital, we will also calculate how many FP vocalisations each test predicts.
To evaluate the system, the true positive rate (TPR), precision and the number of FP vocalisations are used. TPR is calculated by dividing the number of true vocalisation found (TP) on the total number of vocalisation in the data, as Equation 3.2 shows [53]. Precision is, on the other hand, a measure of how many of the vocalisations found were correct, given by Equation 3.3[53].

$$True\,positive\,rate\,(TPR) = \frac{TP}{P} \tag{3.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.3}$$

Testing with NINA's data
As mentioned, the two first test will use NINA's data. Both tests will feed the 72 hours of

test data, as described in Section 3.2.5, through the system. The 72 hours of data is split into six files, where three of them contain 18.6 hours of continuous audio, and three files contains 5.4 hours of data. The system will first be tested by feeding these files into the system, to see if it can handle processing such large files. If the system can not handle it, they will be divided into smaller once. By doing this, we get to test if the system meets the minimum requirement from Section 1.1 of being able to compute at least two hours of data at a time. By feeding continuous files like this we also get to test the system as it is intended to be used by NINA. The results from the tests will then be calculated by comparing the output of the system to the txt-files created through the labelling process described in Section 3.3. So, what differentiates the two tests?
Up to now, we have talked about detecting vocalisations, but what is considered a vocalisation? Does each sound a bird makes count as a new vocalisation? Or are all sound expressions made within a specific time window considered as one? As these types of questions not necessarily has a straightforward answer, the two tests will use different definitions of what a vocalisation is. The definitions are as follows:

1. A vocalisation is one or several sound expressions a bird makes, with no more than 0.5 seconds apart.
2. A vocalisation is one or several sound expressions a bird makes, with no more than 10 seconds apart.

Definition 2 is what the ornithologist at NINA would use, and how the system would be applied there. As this definition looks more at sound sequences, which is more relevant than definition one in bird monitoring. However, according to the ornithologist John Atle Kålås, there is no consensus among bird researchers on how this is defined and could therefore have been done differently by someone else.

When running the first and second tests, all false positive would be visually inspected. When running the first and second test, the false positives would often be checked. This was to check if it was actually a false positive, or if the system might have found a vocalisation that was not found in the labelling process. Any such vocalisations found by the model or author after the first round of labelling would be marked, so one can compare what a human had found vs the model. When going through the FP or any other of the test data, any cases that were too uncertain would be removed, just as done when labelling the rest of the data, as mentioned in Section 3.3. This was done so these would neither gain nor punish the system by selecting a class. The results from these two tests will reveal if the system meets the minimum requirements of being better than random, mentioned in Section 1.1.

Testing with public data
Unlike the two other tests, this only uses data gathered from publicly available sound archives. It focused more on checking whether the systems works on data from all types of sources using different recording equipment and having different soundscapes. However, as mentioned in Section 3.2.1, this data had varying quality and durations, which has to be considered when looking at the results. As much of the data from the publicly available dataset was mp3 files, they first had to be transformed into wav-files to be sent into the

system. This was done using the python library *Pydub*, as mentioned in Section 3.7. The test done were more or less exactly the same as for the two others, except for the mp3 conversion. The fact that the data from the publicly available data had different sampling rates also allowed us to test that the system could handle that. The only difference between the tests is that the this test data was not visually inspected afterwards in the same matter as for the two tests on NINA's data. The reason for this was that it was quite time demanding, and the main focus was at getting the system to work for NINA.

4 Results

In this chapter, we will start by looking at the datasets, in Section 4.1. Section 4.2 continues by looking at the two final models' loss and accuracy curves during training. Then at last, in Section 4.3, we look into how the two versions of the system preformed on the three tests presented in Section 3.9.
Something to note is that not all the parameters used in the two final systems were listed in Chapter 3, but all the parameters used to train and build the two versions of system are listed in Appendix A1.

4.1 The datasets

Up until now, we have talked about the process of collecting data and making the data set. In this section, we will look at how much data was actually collected for each class, and how much of the training data was duplicated and augmented. We will start by looking at the training data. Table 4.1 shows an overview of how many labels were used per class, their average, minimum and maximum duration, and number of data points that was extracted from these labels.

Table 4.1: An overview of the data collected for training.

The class	Number of labels	Average duration	Minimum duration	Maximum duration	Number of data points
JS	815	8.10 sec	0.93 sec	24.62 sec	6900
SR	1669	3.35 sec	0.31 sec	21.61 sec	5500
BS	947	2.28 sec	0.28 sec	17.92 sec	2553
other	2626	5.67 sec	0.26 sec	134.4 sec	13730

If we start by looking at the number of labels listed in Table 4.1, we can see that the JS had the fewest number of labels, following comes the BS class, while SR has twice the amount of JS. The *other* class has over three times as many as the JS, with 2626 cases. Part of the reason why the SR has so many is probably because the two types of vocalisations presented in Section 2.1, sometimes were a little bit apart and therefore labelled separately. Meaning that the average duration of SR probably is a bit longer than the duration listed in Table 4.1. When it comes to the average duration of JS, it is still more than twice as long as SR, and more than three times longer than for BS. When it comes to the minimum duration, we can see that all three bird classes can have short vocalisations, especially the SR and the BS. However, we can also see that they can have a longer vocalisations as well by looking at the maximum duration. Nevertheless, the average indicates that the maximum duration is far from average for all classes. When it comes to the minimum duration of the 'other' class on 0.26 sec, this specific case was discarded, due to the data extraction method described in Section 3.4.1. It should be noted that it was generally avoided to include such short vocalisations/labelled audio, even though some cases may have occurred.
When looking at the last column in Table 4.1 listing the number of data points after the

data extraction, we can see that the other class by far has the most data point with 13730 cases/seconds. After this, we have the JS with 6900, following SR with 5500 and BS with 2553 data points. As mentioned in Section 3.4.3, we can see that the BS has the fewest data points, by far. However, the class was even more unbalanced before the two extra 18.6 hours long wav-files were labelled to help balance the class, as described in section 2.10. By doing this and finding a couple extra vocalisations while double checking all the labels (see Section 3.3), the BS class got 362 additional data points.

To further deal with the imbalance in the data set, both up and down-sampling were tested as described in Section 3.4.3. The dataset that gave the best results for both version of the system on NINA's data was using all the types of augmentation described in Section 3.4.3. However, how big portion of the data set came as a result of duplication and augmentation was never discussed in that section as the amount of data found was not presented yet. Therefore, we will go through how the validation and training set was made as listed below, in the order they were done:

- 3730 data points from the other class were set aside.
- The remaining 24953 data points from all four classes were split into training data and validation data in a 70%-30% split. Setting aside and saving the validation data.
- Duplicating and random shifting all the BS samples and every fifth SR sample remaining in the training set.
- Duplicating and mixing two and two birds: For BS and SR, and BS and JS, this was done for every tenth sample until reaching the end of BS samples. For SR and JS, it was done for every tenth sample until one ran out of SR data points(since SR has less data than JS).
- Duplicating and mixing a bird class with the other class: This was done for all the original BS samples and every sixth SR, using the 3730 'other' set aside in the first step. The other class had a=0.6, in equation 3.1.
- Duplication and adding noise was done for all four classes in varying degrees: Every tenth BS data point, every 15th SR, every 18th JS, and every 20th other.
- Then all the data points were collected, including the other labels set aside and used for data augmentation, in a new CSV-file ready for training.

When duplicating only a part of a class, it should be mentioned that it was tried to switch which data points were duplicated. To prevent that only a portion of a class would have many augmented duplications, while other part had none.

When it comes to the 72 hours of test data from NINA all the data was used, except from 14 smaller sound segments that were too hard to determine and were therefore dismissed. Out of the 14 cases, two were suspected of being SR, while the remaining twelve suspected of being the BS. When counting the vocalisations as sound expressions with more than 0.5 seconds apart (test one), there were a total of 734 vocalisations: 88 belonging to the

JS, 290 belonging to the SR and 356 to the BS. Out of 734 vocalisations, 66 were found later on by either one of the models or by the author after the first iteration of labelling. Showing a human error rate of approximately 9.0% or a human TPR of 91.0% in this project. In most cases, these cases belonged to the BS or SR class. However, four of them were JS cases, all of which were found at location 2 and 4, meaning that the model in the preliminary project had poorer performance than initially thought and listed in Section 1.2.1. Three of these had a duration of 1-2 seconds, and the fourth with a duration of four seconds.

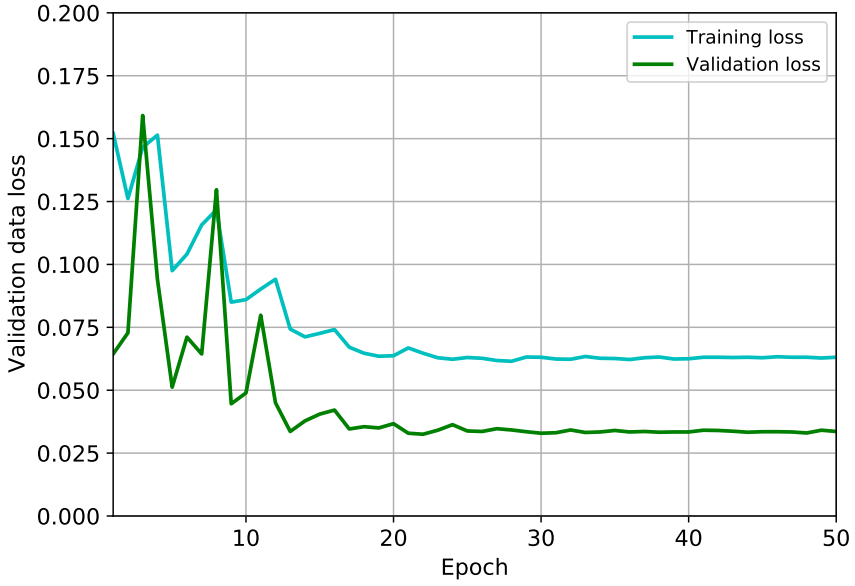
For the second test, where a vocalisation is counted as a sound expression or sequence with more than 10 seconds apart from another vocalisation of the same bird, the data had a total of 555 vocalisations. Of these, 84 belonged to the JS, 222 belonged to the SR and the remaining 249 belonged to the BS.

When it comes to the dataset collected from the publicly available data, there was a total of 81 audio files found: 29 belonged to the JS, 35 to the SR, and 17 to the BS. Appendix A6 will list a complete overview of what sound clips and any additional information about them that were requested by the database they came from. From the 81 audio files, a total of 212 separate vocalisations were found when defining a vocalisation as in test one. The number of separate vocalisations found for each bird was 49 for JS, 58 for SR, and 105 for BS.

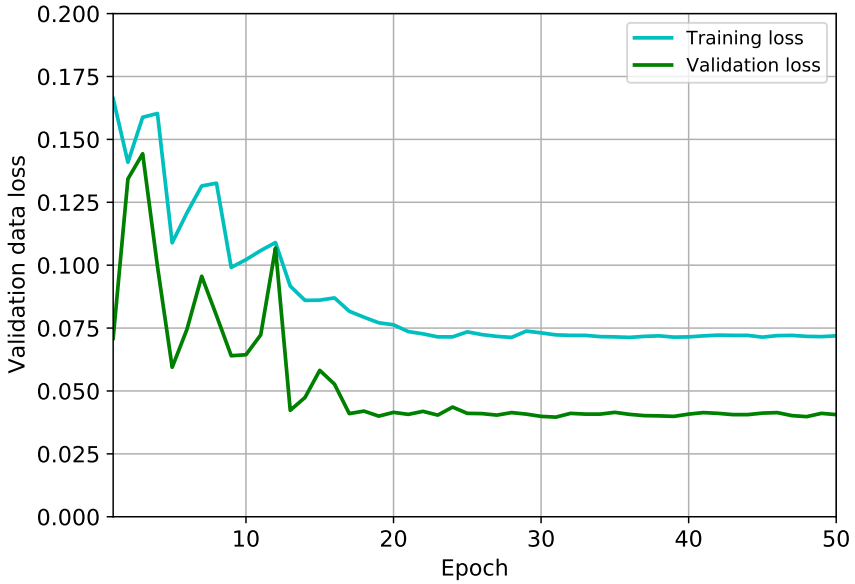
4.2 Training curves for the models

In this section, we will take a look at the loss and accuracy curves for the two final models during training. Both the models were trained for 50 epochs, using the second training method mentioned in Section 3.8 and the augmented data set described in Section 4.1. It took 78 minutes to train the model based on Mel-spectrograms over 50 epochs, while the spectrogram based model took 122 minutes.

Let us start by looking at the loss curves, as the loss was metric of choice to decide what model was the best. Figure 4.1a shows the loss for the Mel-spectrogram model, while Figure 4.1b shows the loss for the spectrogram model.



(a)



(b)

Figure 4.1: Plot of how the average loss progressed during training for both the training (blue) and validation (green) data. Figure 4.1a shows the curve for the model using the Mel-spectrogram, while Figure 4.1b shows the loss curvature for the model using spectrograms.

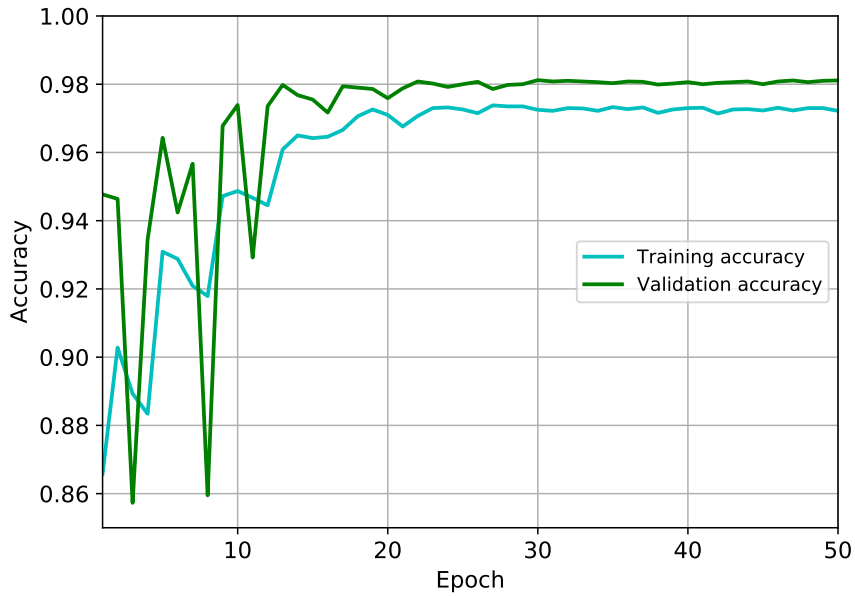
Both in Figure 4.1a and Figure 4.1b the blue line is the average training loss, while the green line is the average validation loss computed per epoch. The Mel-spectrogram model in Figure 4.1a, starts at a loss of 0.1521 for training data the first epoch, while the validation starts at a loss of 0.0644. They then converge to a loss of around 0.072 for the training data, and a loss of 0.033-0.034 for the validation data after about 20 epochs. The same happens for the spectrogram model in Figure 4.1b, which start with a loss of 0.1662 for the training data and a validation loss of 0.0709 in the first epoch.

Then after approximately 25 epochs the loss stabilises at approximately 0.0720 for the training loss, and 0.0400 for the validation loss. After about 18-25 epochs for both models, when the loss have "stabilised", both the training and validation loss only varies about +/- 0.0005. Meaning that there are no clear signs of overfitting, as one would expect the validation loss to increase and the training loss to continue decreasing if this happened [53]. However, we will come back to this topic in Chapter 5. The reason why all the loss graphs are stabilising is probably due to the learning rate scheduler, which in both cases decrease the learning rate every fourth epoch with half its value (gamma=0.5). Meaning that the models' weights will be changed in a smaller and smaller degree during training, as the learning rate is decreased.

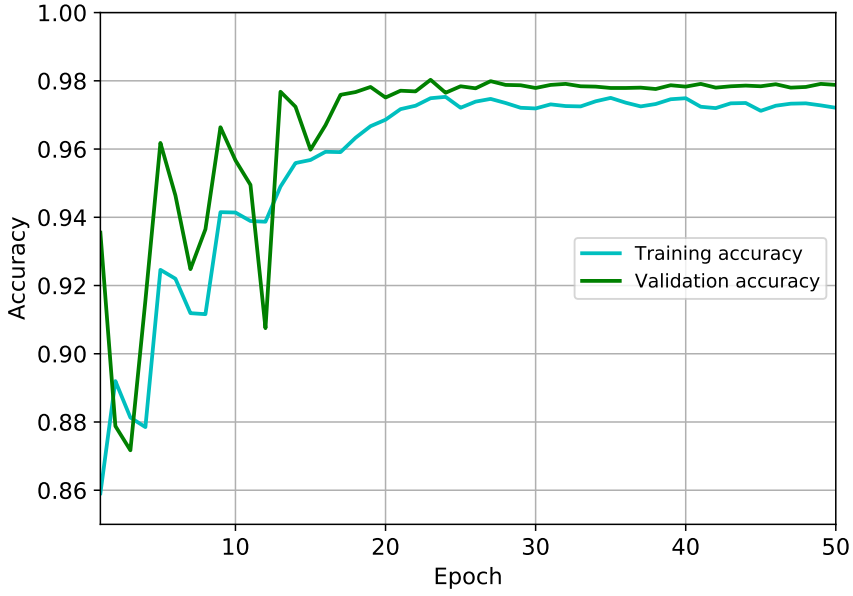
As mentioned in Section 3.8, the best model would be picked from the epoch with the lowest validation loss. For the Mel-spectrogram model this was in epoch 22, where it had a training loss of 0.0647 and a validation loss of 0.0325. While for the spectrogram model, this was in the 31st epoch where it had a training loss of 0.0723 and a validation loss of 0.0396.

In both figures, we can see a clear gap most of the time between the training and validation loss. This seems to be biggest during the first epoch, likely due to the training loss is calculated from the first to the last batch of data of training data. Meaning that the loss of the first time the model sees this type of data is apart of the average training loss of epoch one, while the validation loss is computed after the model has trained once at all the training data. After the first epoch, we see that the loss oscillate a little bit back and forward before stabilising. If we look a bit closer at how it oscillate, one can notice that the best loss comes typically after the learning rate scheduler has decreased the learning rate, which is every 4th epoch. This is probably due to the combination of the learning rate is decreased and the training technique used, which can enable the models to get closer to a local minima. If we then look back at the gap in loss between training and validation, we can see that it is still there after the loss is quite stable. This was speculated to be due three potential reasons: The first being that the validation data's loss is calculated without any regulation techniques such as drop out, while the training data's loss is, the second being that the training data has augmented data which would increase the loss. The last reason is that the model is chosen out from the best validation loss, which would have lead to a better validation loss. As the two first was relatively easy to check, they were confirmed through testing. This was done by first testing the training data on the final model using the evaluation mode, used for the validation and testing. Then, while using the evaluation mode, only the original training data, without data augmentation was tested on the models. This both confirmed that the dropout rate and data augmentation affected the training data's loss and showed that the data augmentation had the biggest impact on it.

Now, let us take a look at the accuracy curves for both models in Figure 4.2. Figure 4.2a shows the accuracy curve for the Mel-spectrogram, while Figure 4.2b shows the accuracy curve for the spectrogram model.



(a)



(b)

Figure 4.2: A graph showing how the average accuracy progressed during training for both the training (blue) and validation (green) data. Figure 4.2a shows the accuracy curvature for the model using the Mel-spectrogram, while Figure 4.1b shows for the model using spectrograms.

As for the loss curves in Figure 4.1, the training data's accuracy is plotted blue and the validation accuracy in green. Both the Mel-spectrogram and spectrogram model's training and validation accuracy starts at a lower value, and converges against a higher value in Figure 4.2, as the loss did in Figure 4.1. For the Mel-spectrogram model the best accuracy was found in epoch 29 where it had an validation accuracy of 0.9812, however the best model was picked by the best loss in epoch 22, which had an validation accuracy

of 0.9808. For the spectrogram, the best accuracy was achieved in epoch 22 with an validation accuracy of 0.9803, while the best loss the model achieved was at the 31 epoch had an validation accuracy of 0.9788. Since the loss and accuracy has a certain correlation, we will see some of the same things in the accuracy graphs in Figure 4.2 as for the loss curves in Figure 4.1. As an example, we can see that the peaks in both Figure 4.2a and Figure 4.2b happens after the learning rate has been decreased and the best model weights has been "updated" while training. We can also see that there is a gap of 0.5%-1.0% in both Figure 4.2a and Figure 4.2b, between the training and validation data, which also is affected by the data augmentation and regularisation used during training,.

4.3 Results from the tests

This section will start by looking at the test results from test one and two, followed by look at the third and last test. As mentioned in Section 3.9, the TPR, the precision and the number of FP are particular important measures for the system and is therefore listed per version of the system and test. The results per class, and for the whole current dataset will be listed as well.

4.3.1 Test 1 and Test 2

We will start by listing up the results of test one and two for both versions of the system, before we will look at some sound examples linked to the results.

Test1, Mel-spectrogram
We start by looking at Table 4.2, which shows the result from the first test using Mel-spectrograms.

Table 4.2: With Mel-spectrogram, test 1.

	TPR	Precision	Number of FP
JS	0.9090	0.8989	9
SR	0.9000	0.9491	14
BS	0.8340	0.9224	25
For all classes	0.8692	0.9300	48

Overall, we see that the TPR=0.8692 with a precision on 0.9300, meaning that version of the system found 86.92% of the vocalisations in the data, and that for every 40 vocalisation it predicted approximately 37 were correct. When looking specifically at each species, we can see that both the JS and SR have a higher TPR than the average, while the BS has a bit lower TPR. When it comes to number of FP cases, the JS has the lowest number with 9 cases, the SR have 14 cases and BS has 25 cases, giving a total of 48. However, it is important to remember that the BS has by far the most vocalisations in the first test as well. Meaning that one missed case (FN) or FP have much less impact on the results of the BS class than on JS in Table 4.2. If we divide the 48 FP cases on three days, we get

approximately 16 FP cases per day/24 hour. When it comes to the systems TNR, given by TN divided on negative cases [53], the system got a TNR=0.9997.

Test1, spectrogram
Now, let us look at the results from the first test using the spectrogram version of the system listed in Table 4.3.

Table 4.3: With spectrogram, test 1.

	TPR	Precision	Number of FP
JS	0.9545	0.9882	1
SR	0.9103	0.9706	8
BS	0.8387	0.9550	14
For all classes	0.8787	0.9656	23

When comparing Table 4.3 and Table 4.2, we can see that the overall TPR for the spectrogram solution is a bit higher than for the Mel-spectrogram solution. The difference in TPR for the BS is less than 1.00%, for SR is it approximately 1.00%, while for the JS it has increased with 4.55%, as the TPR=0.9545. When looking at the FP in Table 4.3 we can see that the JS has one, the SR has eight, and the BS has 14, giving a total of 23 FP. This is approximately half of the amount of FP compared to the Mel-spectrogram solution. The lower amount of FP combined with a higher TPR have lead to a increase of 3.69% in precision for the spectrogram solution. For the JS it has increased the most, by approximately 10%, while the SR has increase 2.15%. The 23 FP, mean that there were on average approximately 8 FP per day. It has also increased the TNR compared to the Mel-spec solution, as it got a TNR of 0.9998.

Test2, Mel-spectrogram
For the second test, the Mel-spectrogram solution got the results listed in Table 4.4.

Table 4.4: With Mel-spectrogram, test 2.

	TPR	Precision	Number of FP
JS	0.9286	0.8966	9
SR	0.9099	0.9352	14
BS	0.8635	0.8958	25
For all classes	0.8919	0.9116	48

The total amount of TPR has gone up for all three classes, when defining the vocalisation a bit differently, where the biggest change lies at BS which went up 2.95%. Nevertheless, the change in vocalisation definition has also gotten the overall precision for the three test days to go down. This is due to the number of vocalisations in the second test is somewhat lower than in the first test as we saw in Section 4.1. The FP number and TNR, on the other hand, stays the same for both test.

Test2, spectrogram
For the second test at the spectrogram solution got the test scores listen in Section 4.5.

Table 4.5: With spectrogram, test 2.

	TPR	Precision	Number of FP
JS	0.9640	0.9878	1
SR	0.9099	0.9619	8
BS	0.8916	0.9407	14
For all classes	0.9099	0.9564	23

When comparing the numbers in Table 4.5 with Table 4.3, we can see that the overall TPR has gone up 3.12%, where the highest increase can be seen for the BS at 5.29 %. However, the TPR for the SR class showed a decrease of -0.50%. Looking at Table 4.5, one can see that the precision decreased for all classes, just like the Mel-spectrogram solution. The number of FP stays the same, but the TNR on the other hand, is 0.9997 due to less number of negative data points in test two.

When it comes to computation time, it took the Mel-spectrogram approximately 5 minutes and 10 seconds on the three test days. In contrast it took the spectrogram solution approximately 12 minutes. Nevertheless, the time could vary some from time to time, especially if the computer was computing the results while running several other things at once. When it comes to the system's computation abilities, the system in both cases were able to handle the shorter and longest wav-files of 5.4 and 18.6 hours without any problems.

Now, let us look a bit closer at why there is a noticeable change in number of vocalisation for the BS class between test one and two, as mentioned in Section 4.1, and why especially the BS class's TPR went up as it did in test two. Figure 4.3 shows one of the better examples on this from one of the test days.

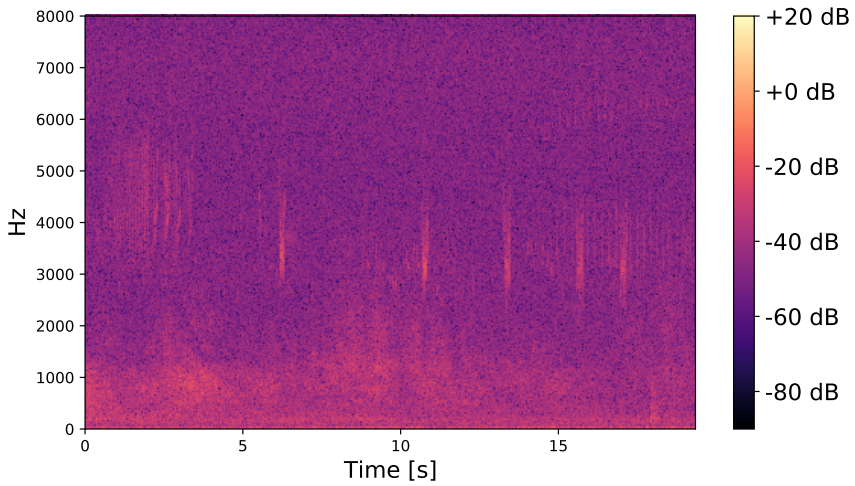


Figure 4.3: A sound clip with BS mating vocalisations. In the first test this sound clip contains six vocalisations, while in the second test it only counts as one.

In test one, there are a total of six sound expression made by the BS in this sound clip, at second: 1-3.5, 6-7, 11-12,13, 16, and 17.5. "Both systems failed to predict vocalisations for the five shorter ones, meaning they can give 5 FN cases. In contrast, the second test would only count all six as one vocalisation, meaning it would only be one TP or FN, from this sequence of sounds.

Now, let us take a closer look at some false negative cases for both tests. Figure 4.4, shows two examples of this.

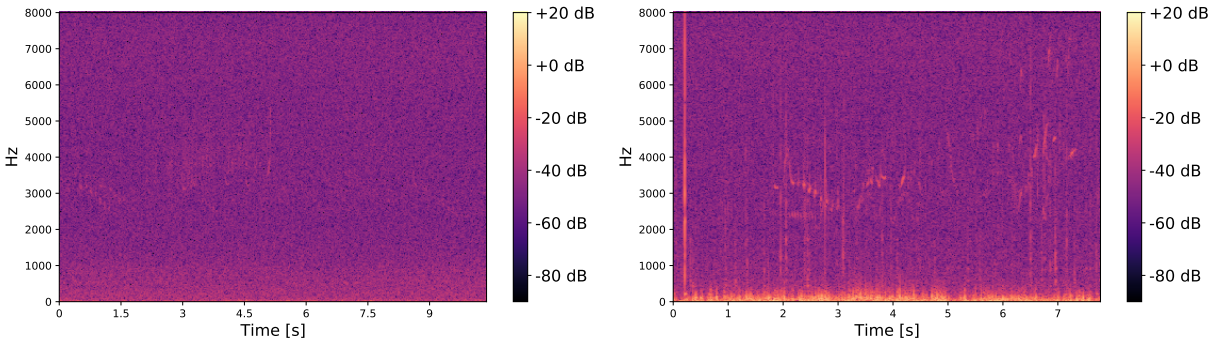
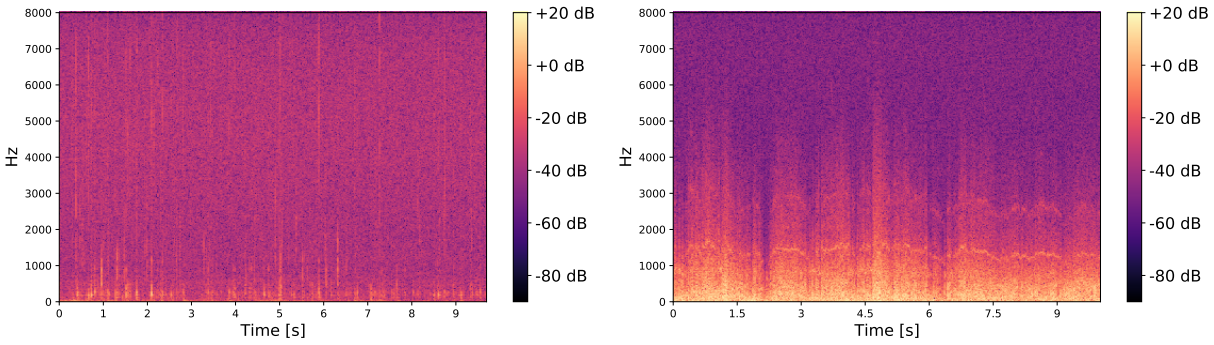


Figure 4.4: Two examples of false negatives.Figure 4.4a is of a BS, while Figure 4.4b shows an SR vocalisation.

Figure 4.4a shows a BS between 2.5-5 seconds in the frequency range 3.0-5.7 kHz, while Figure 4.4b is of an SR from 2 to 4.5 seconds in the frequency range of 2.2-3.2 kHz. As we can see both examples are quite weak, which is the case for most of the FN vocalisations. Several of them are also shorter than the two examples in Figure 4.4, such as the five shorter sound expressions made in Figure 4.3, which can make them even harder to detect.

Now, let us take look at the other group of false prediction; false positives. First, a closer look at the difference in FP cases for the JS for Mel-solution and the pure spectrogram solution. As seen in both tests, the Mel-spectrogram solution has nine FP cases for the Jack Snipe, while the spectrogram solution has one. When inspecting these cases closer, it was discovered that seven of the FP cases for the Mel-spectrogram solution were during heavier rain, which appeared several times during the 72 test hours. Figure 4.5a shows an example of one of these FP predicted by the Mel-spectrograms solution.



(a) **(b)**

Figure 4.5: Two examples of FP by the models. The first case in Figure 4.5a is one of several cases the Mel spectrogram model had of FP prediction of the JS during strong rain, while Figure 4.5b shows a type of wind that would trigger FP for both models.

Another FP example worth showing is in Figure 4.5b, which shows a type of wind both the versions of the system mistook for the SR a couple of times. This is probably due to the winds between 1 kHz, and 3 kHz, which create an activation that can resemble the SR mate attraction vocalisation. When it comes to the BS class, there were not that much repetition in the FP for any of the systems, except for a couple of different looking Bluethroat vocalisations.
When comparing the two versions FP there were not that much overlap, on when exactly the FP occurred for any of the classes. One thing that should be mentioned for the birds FP is that none of their vocalisations were mistaken for another bird, and all their FP cases always belonged to the 'other' class. When it comes to the FP cases duration, most of them had a duration of 3 seconds or longer.

Even though many more cases could have been interesting to look at, there is one type of SR vocalisation in particular that is perhaps more interesting than many of the others. The reason for this is that the vocalisation's frequency pattern diverges from the others seen up to now and found at the different training sites. Figure 4.6 shows an example of this vocalisation in question.

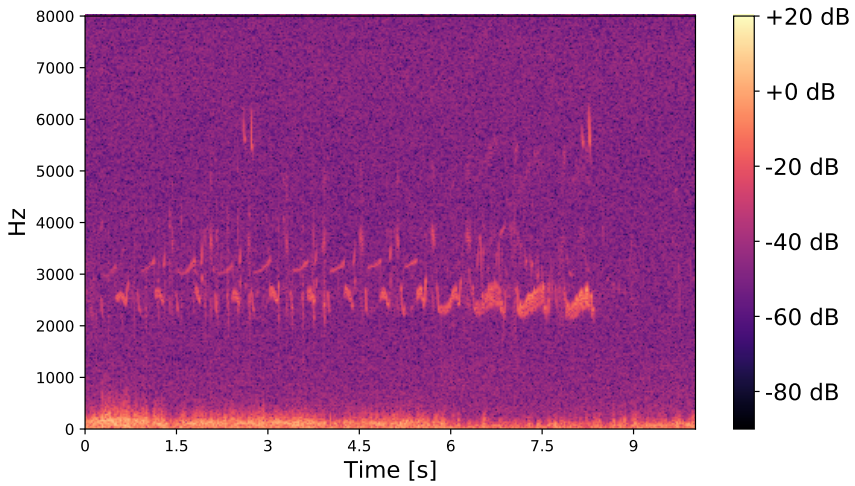


Figure 4.6: A SR mate attraction vocalisation found at location 7.

The vocalisation has a frequency range of approximately 2.2 kHz to 3.5 kHz, and was confirmed to be a SR mate attraction vocalisations by John Atle Kålås. When comparing the vocalisation to the other SR vocalisations found in the test data, such a Figure 2.3, we can see that the start of the vocalisation is somewhat different by looking at its pattern in the spectrogram. Nevertheless, after approximately six seconds it seems more like the vocalisations seen at the training locations. Why these vocalisations are a bit different is hard to say, but it is probably another individual than the ones at the other locations. When it comes to the two versions of the system's performance on these types of vocalisation, it were in most cases not a problem to find them. However, there were a couple of cases where both system missed, even though they were not among the weakest vocalisations. The specific case shown in Figure 4.6 were found by both versions of the system.

4.3.2 Test on public available data

For the third test using the publicly available data and test one's definition of a vocalisation, both versions of the system were tested with two different acceptance thresholds. The first threshold was the same as used during testing on NINA's data(test one and two), and the second was set to 0.5 for all three classes, as the original threshold was relatively high (see Appendix A5).
We will start with the first threshold, where the Mel-spectrogram results are listed in Table 4.6 and the spectrogram solution's result is listed in Table 4.7.

Table 4.6: With Mel-spectrogram, test 3 with the same threshold of acceptance as on NINA's data.

	TPR	Precision	Number of FP
JS	0.5306	1.0000	0
SR	0.6552	0.8261	8
BS	0.0476	1.0000	0
For all classes	0.3255	0.8961	8

Table 4.7: Spectrogram solution results on test 3, using with the same threshold of acceptance as on NINA's data.

	TPR	Precision	Number of FP
JS	0.2449	1.0000	0
SR	0.6552	0.7917	10
BS	0.0190	1.0000	0
For all classes	0.2453	0.8387	10

As we can see, the TPR is much lower for both systems, than in test one and two, especially for the BS class where only 4.76% and 1.90% were found. When comparing the two versions of the system, the Mel-spectrogram has over twice as high TPR for the JS as the spectrogram solution, while they have the same TPR for SR. However, the Mel-spectrogram has, in this case, less FP than the spectrogram solution, making its precision on the SR a bit higher. When it comes to the other classes, both versions of the system had a precision of one due to no FP cases. When it comes to the TNR, the Mel-spectrogram got 0.9929, while the spectrogram solution got 0.9862.

However, when changing the threshold of acceptance in the post-processing step, both systems number of FP change as we can see in Table 4.8 and 4.8.

Table 4.8: Results from Mel-spectrogram solution on test 3, with a threshold of acceptance = 0.5.

	TPR	Precision	Number of FP
JS	0.7143	0.9459	2
SR	0.6552	0.8261	8
BS	0.2857	1.0000	0
For all classes	0.4858	0.9115	10

Table 4.9: Results from spectrogram solution on test 3, with a threshold of acceptance = 0.5.

	TPR	Precision	Number of FP
JS	0.6735	1.0000	0
SR	0.7586	0.6567	23
BS	0.2667	1.0000	0
For all classes	0.4953	0.8203	23

The Mel-spectrogram solution got two additional Jack Snipe FP cases, while the number of FP for the SR increases to 23 for the spectrogram solution. Thus, when it comes to the TPR, it is much more even between the two systems, than for Table 4.6 and Table 4.7. The precision for the Mel-spectrogram solution is lower due to the spectrogram solution's higher number of FP cases. In the case of the Mel-spectrogram, the prediction given for the JS is better than for the spectrogram solution, while the SR is worse. For the last class: BS, have the Mel-spectrogram 1.9% higher TPR. When it comes to the TNR, the Mel-spectrogram solution got 0.9904 while the spectrogram solution got a TNR=0.9723, which is a bit lower but expected to the higher number of FP. A last thing worth mentioning is that in all cases both versions of the system was able to process all the files from the public available dataset, meaning the system was able to handle different sampling rates.

5 Discussion

This chapter is divided into seven sections. The first two, Section 5.1 and Section 5.2, will discuss around the three tests and their results. Then will Section 5.3 talk around the data used and some limitations it has. Followed by Section 5.4, which will talk about the system and its processing capacity and ability to generalise. Then will Section 5.5 and Section 5.6 compare the system against others and talk about usage of it, before Section 5.7 will round off the chapter talking about potential future work.

5.1 Test 1 and Test 2

This section will look closer at the two first tests, which were used to determine the best version of the system, as stated in Section 3.9. The two versions of the system presented with results in Chapter 4 were picked by looking at the TPR and precision, while tweaking each bird's acceptance threshold and minimum duration used in post-processing, as explained in Chapter 3. When choosing which threshold and minimum duration to use in the post-processing step, a trade-off had to be made between FP and FN, which affects the TPR and precision. A lower threshold and a minimum duration would have resulted in more predictions on the bird, which at least in this case would have made the system able to find more of the vocalisations in the data and thereby give a higher TPR. However, this would also have given a terrible precision score. The same goes the other way around. If the threshold and duration were set so only some cases were found per class and no FP were predicted, the precision would then be one, while the TPR would be terrible. Therefore a balance had to be found. It is also why both versions of the system could have had a higher TPR than the results listed in Section 4.3. However, it would have been at the cost of the precision and higher number of FP cases. Although the results from both tests suggest that the spectrogram version worked the best, both will be commented on in through out this chapter. It should also be said that the spectrogram versions were not working best at all times. It could also be that the Mel-spectrogram version of the system could have worked better with some other parameters than the once listed in Appendix A5.

In Section 1.1, we saw that one of the minimum requirements is that the system should predict better than random. As it is a multi-label problem, we will look at each class separately and see if it predicts better than 50% correct per class("random"). By looking at the results in Section 4.3.1, which shows that both version of the system has a TNR of over 99% and a TPR of over 83% per class, we see that the system is much better than random. This means that the system meets this minimum requirement. Nevertheless, even though both versions of the system did way better than random, it is also safe to say that the system is not perfect either and has its limitations. This is why we will look closer at the FP and FN in the following part.

False positive cases are important as they will appear as any other prediction for the end-user of the system, which can not differentiate them from the TP cases without further investigation. As we saw in Section 4.3.1, both versions of the system have some

FP cases for the three test days. The spectrogram solution has a total of 24, while the Mel-spectrogram solution has 48 cases. The number of FP perhaps the most important difference between the two systems test scores, as the Mel-spectrogram solution had over twice the number of FP cases. The most significant difference was for the JS class, where the Mel-spectrogram solution had nine FP cases, while the spectrogram solution had only one FP case. Of the nine cases, seven of them seemed to be caused by rain, which we saw an example of in Figure 4.5a. Rain also happens to be a reoccurring sound in the landscape, meaning that it is likely that the Mel-spectrogram will have reoccurring FP cases over time when there is rain in the wav-files. Nevertheless, it should also be mentioned that several hours of the data did contain rain, and the FP cases listed would be much less than one percent of this time.
Another example mentioned in Section 4.3.1 was the false positive SR cases that seem to come from a specific type of wind. If they were due to a type of wind, they also have a good chance of reoccurring. However, as there were much fewer cases than for the JS for the Mel-spectrogram solution, even though there was a lot of wind, it migth not happen that often. For all the FP cases, a common denominator is that the sound source was in the same frequency range and typically resembles the vocalisation to some degree, which makes sense, as the system thinks it is one of the birds.

When it comes to false negatives, the number itself refers to how many cases/vocalisations were not found during testing and can therefore be an important thing to have in mind. In Section 4.3.1, we could see that the number of FN cases depended on whether it was the first or second test. Showing that how one defines a vocalisation can have a lot to say. Even though both versions of the system's FN cases were often weak or short in both tests, the ones that only were present in test one were close to other sound expressions made by the same bird. This was especially the case for some of the BS cases, as it had both a big shift in the number of vocalisations and TPR between the two tests, as seen in Section 4.1 and Section 4.3.1. However, the BS class did still hold the highest number of vocalisations in both tests on NINA's data, meaning it also has the biggest impact on the test set's overall precision and TPR. However, there was almost the same amount of SR and BS in the second test, while the JS class had considerably fewer vocalisations. This is important to have in mind when looking at the datasets scores and not the individual classes. How many vocalisations there are per species will also affect how much one single vocalisation affects the different "scores" of the class. We will touch more upon this in Section 5.3. Nevertheless, there is also an uncertainty connected to whether one has found all the FN cases and if each vocalisation is labelled correctly. As we saw in Section 4.1, four additional JS cases were found from the two test days used in the preliminary project[26]. However, these were also mostly shorter cases the JS detector could catch do to its minimum duration of three seconds. Still, there was also a human error both in that and this project, as we also saw in Section 4.1, where the human error was calculated to be approximate 9% leaving a TPR of 91%.
When we compare it to the system's scores, we can see that the spectrogram version of the system scored more or less the same in test two for the overall dataset with a TPR of 90.99%. Then when looking at the other test results, we can see that this is not far from the same score neither. Even though both the test and the error rate may not be 100% representative for all NINA's data, it does at least indicate that the system can help to

find approximately the same amount of data as if one use one human being to label it manually. Nevertheless, one should still go through the cases as the precision score is not equal to one, but who knows, this might not be the case for a human either. It is also important to remember that the system is not critical in the way a medical system may be or a system that is scanning for terror threats.

5.2 The third test

Now, let us see closer at the third test, meant as an additional test to look at the system's ability to use other data than NINA's. As we saw in Section 4.3.2, both versions of the systems was tested using different thresholds of acceptance in the post-processing step; one using the thresholds used in NINA's data and one using a 50% threshold on every class. This was done as the thresholds listed in Appendix A5 were chosen based on NINA's data, which the systems also trained on, while the public available data has more noise and other sound in it. Making it interesting to test with another threshold not determined from NINA's data, and see what diffrance it has.

When using the same threshold as in test one and two, the Spotted Redshank had the highest TPR of the classes with 65.52%, for both versions of the system, as shown in Section 4.3.2. However, it is also the only class with FP cases for both versions of the system. When it comes to the JS class, Section 4.3.2 showed that the Mel-spectrogram had the highest TPR for both thresholds but that the difference was the biggest for the first threshold as it went from 28.57% difference to 4.08%. The BS class had, on the other hand, a poor TPR for the first test of 1.90% and 4.76%, but a better one at the second test, yet still was not ideal. This test set is also the BS, the one holding the majority of vocalisations, meaning the BS has a bigger effect on the overall TPR and precision on the test. However, if we look at the classes separately, we also see that both the SR and JS class have not the badest scores when one considers that the data is completely different and that there are bigger uncertainty connected to the data as well. Uncertainties both due to the quality of the data and the test sets' labels as they were not gone through after they were first labelled, as for NINA's test data. Nevertheless, to expect the same results from the third test as for the two priors would be somewhat unrealistic as the data differs entirely from what the model has been trained to recognise. Even good results could have been obtained. It is also important to remember that this project has not focused on making the system perform best on this test or type of data, but for it to perform well at NINA's data, which the system is intended for.

5.3 The data and its limitations

In this report, data has been a topic several times, both when discussing topics such as what data was available and how it was used, and the soundscape in NINA's data. As no dataset was available, one of the main tasks in this thesis was choosing which data to use, labelling it and making datasets out of it. This presented several challenges and decisions that had to be made. One of these was what data to use. This is why a plan on how to

utilise the available data as good as possible was made in Section 3.2.5. Nevertheless, it will always be hard to say want would have worked the best for the project, using the available time.

One of the biggest challenges related to the data was how to label it. Getting assistance from the bird researcher/ ornithologist, John Atle Kålås, did increase the quality and probably the credibility of the dataset. However, the fact that the author made the final decision on what to label or not, probably is one of the most significant uncertainties of the project, both for thee classifiers built and the results presented in Chapter 4.
This uncertainty is especially the case for the BS class, which both the author and the ornithologist had problems with at times, as mentioned in Section 3.3. It was particularly hard because other birds could both resemble its vocalisations or mask the bird. It also had a more complex vocalisation than the two others, especially the JS. The difficulty around labelling the BS class, in particular, is also reflected by the cases removed from the three test days. As 12 out of 14 cases were connected to the BS. Nevertheless, there were also cases used in the test data that could not be confirmed with 100% certainty. However, if one were to remove all cases that could not be said with certainty, the test would be far from the same. The two tests using NINA's data would also not test on the real use case if to many cases were removed, as the real recordings will have different strengths and duration in the vocalisations .Therefore, it is just something that has to be taken into account when reviewing both the results and the way of detecting vocalisations. Nevertheless, there are bright sides to the fact that the author made the dataset. It allowed one to get to know the data and the decisions made along the way. Including the datasets' strengths and weaknesses.

When it comes to the three test days used, it allowed one to test for a real use case and also test for a relatively large amount of data. Nevertheless, the number of vocalisations for each bird and the sound made by other sources can vastly diverse from one day to another. Thereby, the results given in Chapter 4 could be somewhat different if one had use three completely different days. Both in the number of vocalisation, how easy they could be for different versions of the system to detect, or how many false-positive cases the system producers. This was also reflected in the three test days used as all the SR cases come from one day, meaning two of the three days only tested whether it gets FP or not. That the number of vocalisation can change from day to day could also be seen if one compared the training and test data in Section 4.1. Where we can see that the BS holds the majority of vocalisation in the test data while not in the training data. However, this could be due to the absence of SR in two of the test days and the fact that the training data is the average of 18-19 days of data taken from 2016 and 2017 (Appendix A4). It also does that the BS class has the most to say for the overall results, while if one of the other class had the majority of cases, its class would have the most significant impacts.

Another thing worth commenting on regarding the training data is the estimated human error at 9%, reflecting some of the difficulties in catching all the cases the first time one looks and by a human. Nevertheless, the error rate is estimated based on the author's ability to label and not necessary the researchers at NINA. Although, in the preliminary

report, the author's labelling was compared to the ornithologist, where the author found the most cases. This was speculated to be due to thorough labelling from the author's side. It should also be commented on that 9% is relatively high, and almost double as high as the estimated human labelling error at 5% mentioned in Section 3.3. This is thought to be because of the challenges in labelling the data used, and that the 5% comes from another type of data

When it comes to the public data, there are several uncertainties both to its quality and how well it is labelled, as mentioned in Section 3.9. As mentioned in Chapter 3, most of the publicly available data had to be transformed from mp3 to wav file format to be run through the system. We do not know what effect this has, as the data is already compromised when transformed into wav-files. This was not tested further in this project but could have been tested by transforming the test data to mp3 and back again to wav-files, then run it through the system and check the initial results.

5.4 The systems

5.4.1 Processing time and capacity

This part will take a brief look at both systems' computation speed and capacity. As mentioned in Section 3.4.2, one of the recommendations from the preliminary project was to look into other methods for transforming the data than using PNG images, as it was pretty time-consuming. Even after getting the GPU at the project's disposal, as mentioned in Section 3.7, it would still be quite time-consuming, since the most time demanding part of the system, making the images would have to run at the computer's CPU instead of GPU. A process that could take around 2 hours 33 minutes per day it was computing. In contrast, it takes less than two minutes for the Mel-spectrogram solution and four minutes per day for the spectrogram solution, when the computer is not computing anything else. As we can see, this is a tremendous improvement in speed. When comparing the two versions of the system, the one using Mel-spectrogram vs the one using the spectrogram, we can see that computation time using the spectrogram is more than twice as long time. This is due to the Mel-spectrogram almost having almost one-fourth of the data and a smaller input put layer as well.
However, when the system is computing a day as fast as it is for both versions of the system, we should also ask ourselves if we need the fastest solution if it will cost in performance? Here it is important to point that the system is not a real-time system, and therefore is not the Mel-spectrograms solution's speed needed. With smaller changes to the code of the system, it could iterate over many different files at once without human supervision. Meaning the system should be able to do all the computations over one night if it is using the same GPU everything goes well, after being set up correctly. Thereby, even if the Mel-spectrogram is faster, the tests suggest that the spectrogram solution is the better choice for NINA, as it seems to be more than fast enough and give the better results on their data.

When it comes to computational capacity, Section 1.1 stated that one of the minimum requirement was to compute at least two hours at once. This requirement was met as the system was able to compute the longer test files of approximately 18.6 hours with 16 kHz in sampling rate, as mentioned in Section 4.3.1. In addition to this, the system was also able to handle different sampling rates, as was set as a goal in Chapter 3.

5.4.2 Comments to how the system was built

When using machine learning, one has a sea of different possible methods and structures to try. Therefore, it is most likely some other type of model's structures and techniques out there that would have worked just as good, if not even better. However, we will not know without setting up and testing them. Many different decisions were made for this project, and we will not look at all of them. However, we will look at some selected ones.

First, we will look at how the data was segmented. In this project, it was determined to use one-second segments, as explained in Section 3.4.2. This was because it seems to work good in the preliminary project and that it would give a prediction per second feed-in. However, perhaps one should have tested with longer second segments as well on either 1.5, 2 or 3 seconds, for example. Maybe, longer segments would have caught more of the patterns at times and helped the model detect more such it could distinguish better between false and positive cases. However, longer segments would still have "edges" where a vocalisation could just have ended or started, or contain some really short vocalisations at times, meaning only a small part of the data set in would contain a vocalisation. Nevertheless, it was also thought that the way all the data samples overlapped 50% in testing/production combined with the minimum duration in the post-processing step would help filter away cases where the model had mistaken a negative data point for a positive data point. Another thing to think about is that there was limited time for the project, and one had to choose what to test.

When it comes to the training set, several small decisions had to be made in an attempt to extract the multi-label data in a good matter. However, perhaps some of the decisions made, explained in Section 3.4.1, should have been done differently. Maybe the labels should have been dealt with higher accuracy and not rounded off to a second given with one decimal. Maybe the threshold saying that at least 25% of the segments had to contain a bird label for the segment to be label the same was either too high or low, or it should have been done a different way.

Then we have the handling of the unbalanced dataset. In this project were two main techniques tested and used: adding data and resampling of the dataset. By going through some extra data, it increased the minority class, BS by 362 data points, which lead to an increase of 16.5% datapoints for that class. Then when testing, we saw that upsampling the dataset, especially focusing on balancing the data using all three using data augmentation seems to work best. However, as we also saw in Section 3.4.3 were there many other augmentations techniques that might have worked. It should also be mentioned that it

should probably have been tested with other types of noise than just the uniform one. This could also have helped on the system's overall performance in the third test and on other types of data in general as noise is supposed to help the model becoming more robust.

When it comes to the model, it was as stated in Section 3.5, decided to only use predefined structures that had to seem to work on similar project and ratter adapt than. This allowed one to focus more on other aspects of the project, such as how to train the model, collect more data and test different varieties of the techniques used. Several of these techniques used was to prevent overfitting as we will look at in the next section.

5.4.3 Is the system able to generalise?

As mentioned in the last section, several techniques were used during training to prevent overfitting of the models, such as drop out, the learning scheduler, and early drop-out. When it comes to the training graphs presented in Section 4.2, there were as mentioned no signs of overfitting, as both the training and validation data saturated around a number. However, the early stopping technique should have stopped it by taking the model with the lowest validation loss anyway. Even though the graphs do not show any clear signs of overfitting, we can still not know for sure just by looking at the graphs.

When looking at the results from the first and second tests, both models seem to also generalise well on NINA's data. The fact that it is able to pick up on the somewhat "special" SR mate attraction vocalisation also seems promising, as it was not included in the test data. However, when it comes to generalising on other data sources than NINA's, we saw in test three that the model did not perform as well. Suggesting that the system might be somewhat overfitting on NINA's data, especially for the BS. Nevertheless, it is some questions around the qualities around the test, and it does not mean that the system will not work for NINA. The system might even work more or less the same for data collected with similar equipment chains, however we will not know without further testing.

5.5 Comparison to other systems

The previous work section (Section 1.2) showed that there were only three related projects that had looked into any of the vocalisations of interest, as far as the author of this thesis could find: BirdNET, the preliminary project, and Maja Sofie Stava's master thesis. In the preliminary project, and as one can also see by studying the description in Section 1.2.1, the preliminary project performed better than Maja S. Stava's JS detector. This is why only the preliminary project and BridNET will be discussed and compared against the spectrogram version of the system in this section.

BirdNET:
BirdNET is the only publicly available system and also the only system prior to this thesis

that could detect any of the vocalisations of interest to the best of the author's knowledge. However, the system did also detects more than just the vocalisation of interest to the birds. Nevertheless, how accurate it makes the predictions is another question. To look into this in the preliminary project, a little test was made setting the preliminary project and the BirdNET demo[88] at the projects page up against the preliminary system. The test showed that the BirdNET found the stronger cases, but had problems with the weaker ones, that the JS detector was able to find in most cases. Nevertheless, both systems are built somewhat different. We also do not know how the system would perform on the SR or BS, as this were to look into in this thesis.

The preliminary project:
When it comes to the preliminary project's "Jack Snipe detector", there are both similarities and differences, which mostly has been compared throughout Chapter 3. The biggest difference is, of course, that the current system detects two more species mate attraction vocalisations. Nevertheless, there are also some other big changes if we only look at how the system is built and the results in the JS class. One of them is that the models in the current system were trained on much more data, and not at least diverse data from several locations and not only one as the JS detector was. Another huge difference is that the PNG images are no longer used, and the computation speed is much better, as we saw in Section 5.4.1. It also allows us not to have to plot, save, compute, and delete the images, making the current solution less "clumsy". When it comes to testing, the JS detector was not tested in the same matter as the system built in this project. It had a third test day from the same location as the training data and had no firm definition of a vocalisation. Also, as we saw in Section 4.1, four additional vocalisations were found in the test days used in the preliminary project, meaning that its real TPR is 93.85%. Nevertheless, the JS detector did not get any FP and thereby had a precision=1.0. In contrast, the spectrogram version of the system had one FP case and precision of 98.82% or 98.78%, depending on how one defines the vocalisations. However, the current system did not have such big of a problem with the shorter JS vocalisations as the preliminary project's. This project avoided this by having the data points overlapping 50% and a minimum duration of two data points instead of no overlap and a minimum duration of three data points. Nevertheless, the latter part was never tested for the preliminary project, and could have lead to higher TPR. However, it would also have lead to a more time-consuming computation as it would almost double the number of data points.

5.6 Usage of the systems

This section will mainly focus on the potential usage of this system for NINA, as their research was the main reason for doing this project. The system built in this project has certain limitation. However, the tests also indicate that the system is approximately as good as a human being. Nevertheless, as discussed in Section 5.3, we can not know this is the case for all NINA's data without further testing. However, regardless of this, it would be recommended for them to check all positive prediction the system makes, especially if they were to change their equipment chain any further or change the recording

locations. This could either be done by tweaking the system to write these cases to separate wav-files or using another program such as Audacity, which was, as mentioned in Section 3.7, used for labelling the data. Except for ensuring that the vocalisations are true, it could also be a good idea to go through if one wants to know the number of vocalisation made, as the system only detects whether one of the vocalisations of interest is present or not, not whether multiple birds from the same species are vocalising at once, which sometimes was the case. However, even though someone should go through the predictions to validate them, the system can still save NINA a lot of time compared to the time it would demand to go through it manually. It is also a possibility only to use one or two of the classes predictions as especially the JS vocalisation had relatively high TPR and precision. Another possibility is to have one person go through parts of the data in addition to the system computing the results.

For other use cases than on NINA's data, it is strongly recommended to test the system further on the intended data/use case due to the indications that the system works much poorer on data from other data sources. Nevertheless, there is also a possibility that some changes to the training of the model, which could potentially improve the results scores for the third test. This will be elaborated more on in the next section.

5.7 Future work

When it comes to future work, there can be the type that focuses on on improving the system performance, user-friendliness or the kind that goes more out of the scope of the problem described in Section 1.1 and looks into expanding what the system can detect. We will start by talking about the first type of future work. First of all, we can not know what will work the best, but we can speculate and look at what seems more likely to help. Collecting more data to train on is among the more likely things to help, especially making a more extensive training set on the Broad-billed Sandpiper. This can be by using more of NINA's data or by recording and collecting new data. The latter is also recommended doing using different types of equipment chains and other locations, perhaps even in other countries if possible, if one wants the system to work across even more soundscapes and equipment. This could also open up for making better tests for the system's generalisation on the vocalisations of interest. Nevertheless, it would also be recommended to look into and test more with data augmentation, especially with noise injections, as this is thought to make the system more robust. It may even boost the performance somewhat in test three. However, it is also recommended to test the system further, with more days to evaluate it. In addition, it could also be a good idea to take a deeper dive into the dataset made by publicly available data, as this was not done after making the dataset, as stated in Section 3.9. Including looking into if the mp3 format has anything to say or not.

It could also be possible to split the BS mate attraction up to two different classes, one class for each type of sound, especially if more data on the BS mate attraction vocalisations were collected. The same could be done for the SR's two types of vocalisation presented in Section 2.1. This way, each class would only contain one main pattern it would have to recognise. The results of these classes could also be added together before writing the

cases to the CSV file.

It could also be possible to go through all the labels and removed the weakest cases or at least test without them, as they have a higher uncertainty around them since they often are harder to verify. This was done for the preliminary project. However, the effect it had was never tested and therefore not prioritised in the thesis. Another thing that could perhaps improve the dataset and its balance is to look closer at the 'other' class. Since the 'other' data class is "randomly" collected over time and the largest class, some of the samples could probably be removed. By using clustering, an unsupervised ML technique, to look at the data's resemblance, one could have done selected downsampling. This could then have helped removed some examples there may be an abundance of. This could also open up for adding samples from other locations and perhaps sounds recorded with other equipment without affecting the datasets balance too much.
Another thing clustering or something called Principal component analysis (PCA) could be used for is to check the convolutional layer's ability to due feature extraction. This would be done by removing the fully connected layer and running data through them before using PCA or clustering to group the data. This can be used to consider if one should test with two fully connected layers at the end of the model instead of one. This technique was used in early testing to look at the feature extraction abilities of the model, but not on the later versions.

In addition to the things mentioned above, are there still many other options as testing with more with different parameters and the techniques already used, test with different lengths on the data segments, other unbalanced dataset techniques than collecting more data or re-sampling, testing with attention models (mentioned in Section 3.4.2), or noise reduction methods. One could also change the type of model completely to test other approaches, such as recurrent model or look into hidden Markov models, both of which are used within speech recognition.

Nevertheless, the system has a relatively good TPR and precision, and further improvement may not be needed. Therefore, potential future work could also be to add more vocalisations and try to train the model further. This could be vocalisations of other birds or other vocalisations from the three birds of interest than those in this project. Another way to angle future work could be adding smaller features such as the ability to run through a folder of wav-files as mentioned in Section 5.4.1, or give the user the ability to add together vocalisations within X seconds of each other. It could also be to make a graphical user interface for the less technical researchers at NINA, enabling them to use the system easier.

6 Conclusion

This thesis aimed to make a system to help NINA with detecting specific vocalisations from the JS, SR and BS in wav-files, as a tool for their research. By getting familiar with the vocalisations of interest and the data they wanted to process, it made it possible to use some of NINA's data to make multi-label datasets. This could then be used to both make the system, using a neural network, and test it. To evaluate the final versions of the system a total of three tests were made.

Two of the tests used 72 hours of NINA's data to test the systems ability to detect the vocalisations. The first test looked at the single sound expressions, while the second looked more at sound squeezes made by each bird and is the most relevant for NINA's research. The test showed the best results coming from a model using spectrograms at the input. It got a TPR of 87.87% and precision at 96.56% on the first test and a TPR of 90.99% and precision of 95.64% on the second test on the overall dataset. Using the same test data a human TPR was calculated to be approximately 91%, indicating that the final system should match the work of a human during labelling. When it comes to computation capacity, the system was able to process the longest wav-files of 18.6 hours with 16 kHz in sampling rate. For computation speed, on the other hand, the system was able to compute 24 hours of data in 4 minutes using a RTX3090.

The third test used the limited amount of publicly available data on the vocalisations of interest to test the system on other sources. Even though no indication of overfitting was found on NINA's data, the third test indicated that the system might not be able to generalise as well on a random data source, especially the BS vocalisations. However, there are also questions related to the data quality. Therefore, it is recommended to test the system further if more publicly available data is made available or data using other equipment chains than NINA did as a part of future work. It is also recommended that the system be further tested as three days will not necessarily give an accurate image of all of NINA's data. Particularly for the SR class, as only one test day contained its vocalisation. To improve the system's performance any further, it is recommended to gather more data, especially for the BS class, and look further into data augmentation and other data balancing techniques. Other potential future work could be to make a GUI for the less technical researchers at NINA.

The thesis itself has contributed by making a potential tool for NINA to process all their data, which can be used to select the data a researcher should go through, and potentially save them a lot of time. It has also contributed with several spectrograms for each species, going into its spectral features, and relatively large multi-label training and test sets on the three species.

References

[1] Edvard Kaurin Barth and Jørn H. Hurum (Universitetet i Oslo) Jan Ove Gjershaug (Norsk institutt for naturforskning). The norwegian lecicon (store norske leksikon) on the page for birds. URL https://snl.no/fugler. (Accessed: 10.11.2020).

[2] *Bird study.* British Trust for Ornithology, Thetford, Norfolk, UK, 1954.

[3] Elliot Coues. *Birds of the Northwest : A hand-book of the ornithology of the region drained by the Missouri river and its tributaries.* Washington, 1874.

[4] Thomas Bewick. *A history of British birds*, volume 2. Edw. Walker, 1826.

[5] IUCN. International union for conservation of nature (iucn) redlist webpage, . URL https://www.iucnredlist.org/. (Accessed: 1.05.2021).

[6] George F Barrowclough, Joel Cracraft, John Klicka, and Robert M Zink. How many kinds of birds are there and why does it matter? *PLoS One*, 11(11):e0166307, 2016.

[7] R. Dennis Cook and Jerald O. Jacobson. A design for estimating visibility bias in aerial surveys. *Biometrics*, 35(4):735–742, 1979. ISSN 0006-341X.

[8] John C Montgomery and Craig A Radford. Marine bioacoustics. *Current Biology*, 27(11):R502–R507, 2017.

[9] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[12] Md Zahangir Alom, Tarek M Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S Awwal, and Vijayan K Asari. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*, 2018.

[13] Tom M Mitchell. *Machine learning.* McGraw-Hill series in computer science, Artificial intelligence. McGraw-Hill, New York, 1997. ISBN 0070428077.

[14] Bernard Marr. *Artificial intelligence in practice: how 50 successful companies used AI and machine learning to solve problems.* John Wiley & Sons, 2019.

[15] Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th annual meeting of the Association for Computational Linguistics*, pages 26–33, 2001.

[16] Wouter Verbeke, Karel Dejaeger, David Martens, Joon Hur, and Bart Baesens. New

insights into churn prediction in the telecommunication sector: A profit driven data mining approach. *European journal of operational research*, 218(1):211–229, 2012.

[17] Francesco Clavorà Braulin and Tommaso Valletti. Selling customer information to competing firms. *Economics Letters*, 149:10–14, 2016.

[18] Christopher J.C. Burges Yann LeCun, Corinna Cortes. The mnist database. URL http://yann.lecun.com/exdb/mnist/. (Accessed: 5.12.2020).

[19] The Sound Understanding group in the Machine Perception Research organization at Google. Google's audioset. URL https://research.google.com/audioset/. (Accessed: 16.05.2021).

[20] The freesound database. URL https://freesound.org/. (Accessed: 4.12.2020).

[21] Cornell Lab of Ornithology of the Cornell University. Macaulay library: World's largest archive of animal sounds. URL https://www.macaulaylibrary.org/.

[22] Xeno canto foundation. Xeno-canto:bird song database from netherlands. URL https://www.xeno-canto.org/. (Accessed: 26.11.2020).

[23] Ileana Betancourt and Colleen M McLinn. Teaching with the macaulay library: an online archive of animal behavior recordings. *Journal of Microbiology & Biology Education: JMBE*, 13(1):86, 2012.

[24] K.-B. Strann, R. Rae, and Frivoll. Overvåking av hekkende vadefugler med østlig utbredelse i palsmyr i goahteluoppal, kautokeino 2009-2012 : Sluttrapport. 968, 2013.

[25] *Norsk fugleatlas : hekkefuglenes utbredelse og bestandsstatus i Norge*. Norsk ornitologisk forening ; [Trondheim] : I samarbeidmed Norsk institutt for naturforskning, Klæbu, 1994. ISBN 8299086825.

[26] Ingrid Grimstad. Detection of jack snipe vocalisation using neural networks. unpublished, 2020.

[27] The warblr team. Warblr's offical page, . URL https://www.warblr.co.uk/. (accessed: 4.12.2020).

[28] The warblr team. Warblr's technology information page, . URL https://www.warblr.co.uk/our-technology. (Accessed: 4.12.2020).

[29] Ablemagic. Whatbird's offical page. URL https://whatbird.no/. (accessed: 4.12.2020).

[30] Kristian Selboe. Implementation and adaptation of a system for automatic classification of birdsong. Master's thesis, NTNU, 2015.

[31] Cornell Lab of Ornithology and Chemnitz University of Technology. Birdnet offical page. URL https://birdnet.cornell.edu/. (accessed: 4.12.2020).

[32] Stefan Kahl, Shyam Madhusudhana, and Holger Klinck. Github account where one can download birdnet's code. URL https://github.com/kahst/BirdNET. (Accessed: 4.12.2020).

[33] Cornell Lab of Ornithology. Cornell birdcall identification. URL https://www.kaggle.com/c/birdsong-recognition/. (Accessed: 29.04.2021).

[34] Stanley (chief editor) Cramp, K E L Simmons, Duncan J Brooks, N J. Collar, E Dunn, R Gillmor, P A D Hollom, R Hudson, E M Nicholson, M A Ogilvie, et al. *Handbook of the birds of Europe, the Middle East and North Africa : the birds of the Western Palearctic : 3 : Waders to gulls*, volume 3. Oxford University Press, Oxford, 1983. ISBN 0198575068.

[35] Jan Eivind Østnes. The norwegian lexicon website on scolopacidae, . URL https://snl.no/snipefamilien. (Accessed: 15.03.2021).

[36] Michael Allaby. *A Dictionary of Zoology*. Oxford University Press, 2014. ISBN 9780191764882. doi: 10.1093/acref/9780199684274.001.0001. URL https://www.oxfordreference.com/view/10.1093/acref/9780199684274.001.0001/acref-9780199684274.

[37] IUCN. International union for conservation of nature (iucn) redlist page for the jack snipe, . URL https://www.iucnredlist.org/species/22693133/86640472. (Accessed: 30.10.2020).

[38] IUCN. International union for conservation of nature(iucn) redlist page for the spotted redshank, . URL https://www.iucnredlist.org/species/22693207/86682083. (Accessed: 30.02.2021).

[39] IUCN. International union for conservation of nature(iucn) redlist page for the broad-billed sandpiper, . URL https://www.iucnredlist.org/species/22693464/155481741. (Accessed: 30.02.2021).

[40] *Nature's music : the science of birdsong*. Elsevier Academic, Amsterdam ;, 2004. ISBN 1-280-96690-4.

[41] Jan Eivind Østnes. The norwigan lexicon on jack snipe, . URL https://snl.no/kvartbekkasin. (Accessed: 30.10.2020).

[42] France OMPO/CICB, Paris, editor. *The Jack Snipe Lymnocryptes minimus, written by OLIVER, G-N*. 2007. ISBN 978-2-9527375-1-7. Translated from Franc to English, by Jeanne-Amélie Oliver.

[43] C. K Catchpole and P. J. B Slater. *Bird Song: Biological Themes and Variations*. Cambridge University Press, Cambridge, 2008. ISBN 9780521872423.

[44] Almo Farina. *Soundscape Ecology: Principles, Patterns, Methods and Applications*. Springer Netherlands, Dordrecht, 2013. ISBN 9400773730.

[45] Oxford Dictionaries. *ambient noise*. Oxford University Press, 2002. ISBN 9780199891580. doi: 10.1093/acref/9780199891580.013.0358. URL https://www.oxfordreference.com/view/10.1093/acref/9780199891580.001.0001/acref-9780199891580-e-358.

[46] Marcel Escudier and Tony Atkins. *signal-to-noise ratio*. Oxford University Press, 2019. ISBN 9780191870866. doi: 10.1093/acref/9780198832102.013.5662.

URL https://www.oxfordreference.com/view/10.1093/acref/9780198832102.001.0001/acref-9780198832102-e-5662.

[47] ISO 9613-2:1996(E). Acoustics — Attenuation of sound during propagation outdoors — Part 2: General method of calculation. Standard, International Organization for Standardization, March 1996.

[48] Jack W. Bradbury and Sandra L. Vehrencamp. *Principles of animal communication*. Sinauer Associates, Sunderland, Mass, 2nd ed. edition, 2011. ISBN 9780878930456.

[49] *Animal Communication and Noise*, volume 2 of *Animal Signals and Communication*. Springer Berlin Heidelberg : Imprint: Springer, Berlin, Heidelberg, 1st ed. 2013. edition, 2013. ISBN 3-642-41494-X.

[50] J Tate Mason, Christopher JW McClure, and Jesse R Barber. Anthropogenic noise impairs owl hunting behavior. *Biological Conservation*, 199:29–32, 2016.

[51] Xiao-Jing Yang and Hans Slabbekoorn. Timing vocal behavior: lack of temporal overlap avoidance to fluctuating noise levels in singing eurasian wrens. *Behavioural processes*, 108:131–137, 2014.

[52] Marvin Minsky and Seymour A Papert. *Perceptrons: An introduction to computational geometry*. MIT press, 2017.

[53] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Incorporated, Sebastopol, 2019. ISBN 9781492032649.

[54] Andrew M. Colman. data set, 2015. URL https://www.oxfordreference.com/view/10.1093/acref/9780199657681.001.0001/acref-9780199657681-e-2080.

[55] New oxford american dictionary., 2010.

[56] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. *CoRR*, abs/1707.02968, 2017. URL http://arxiv.org/abs/1707.02968.

[57] Hao Dong, Zihan Ding, and Shanghang Zhang. *Deep Reinforcement Learning: Fundamentals, Research and Applications*. Springer Singapore Pte. Limited, Singapore, 2020. ISBN 9811540942.

[58] *Artificial Neural Network Modelling*, volume 628 of *Studies in Computational Intelligence*. Springer International Publishing : Imprint: Springer, Cham, 1st ed. 2016. edition, 2016. ISBN 3-319-28495-9.

[59] Igor Livshin. *Artificial Neural Networks with Java: Tools for Building Neural Network Applications*. Apress L. P, Berkeley, CA, 1 edition, 2019. ISBN 9781484244203.

[60] Pramod Singh and Avinash Manure. *Learn TensorFlow 2. 0: Implement Machine Learning and Deep Learning Models with Python*. Apress L. P, Berkeley, CA, 2020. ISBN 9781484255605.

[61] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[62] *Support vector machine in chemistry*. World Scientific, Singapore ;, 2004. ISBN 1-281-93460-7.

[63] Yusuke Sugomori. *Deep learning : practical neural networks with Java : Build and run intelligent applications by leveraging key Java machine learning libraries*. Packt, Birmingham, England, 1st edition. edition, 2017. ISBN 1-78847-171-7.

[64] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[65] Alberto Fernández. Learning from imbalanced data sets, 2018.

[66] Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas, et al. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30(1):25–36, 2006.

[67] The museum für naturkunde, berlin. URL https://www.tierstimmenarchiv.de/webinterface/contents/index.php. (Accessed: 30.11.2020).

[68] Daniel Chandler and Rod Munday. mp3, 2020. URL https://www.oxfordreference.com/view/10.1093/acref/9780198841838.001.0001/acref-9780198841838-e-1796.

[69] Angus Stevenson. Wav file, 2010. URL https://www.oxfordreference.com/view/10.1093/acref/9780199571123.001.0001/m_en_gb0992230.

[70] Artsdatabanken. Artsdatabanken sin side. URL https://artsdatabanken.no/. (Accessed: 4.04.2021).

[71] Tom Schandy. The norwegian lexicon on the bluethroat (blåstrupen). URL https://snl.no/bl%C3%A5strupe. (Accessed: 23.05.2021).

[72] Audacity. The official site to by audacity. URL https://www.audacityteam.org/. (Accessed: 17.10.2020).

[73] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[74] Kuldip Paliwal, James Lyons, and Kamil Wojcicki. Preference for 20-40 ms window duration in speech analysis. pages 1 – 4, 01 2011. doi: 10.1109/ICSPCS.2010.5709770.

[75] Loris Nanni, Gianluca Maguolo, and Michelangelo Paci. Data augmentation approaches for improving animal audio classification. *Ecological informatics*, 57: 101084, 2020. ISSN 1574-9541.

[76] Gianluca Maguolo, Michelangelo Paci, Loris Nanni, and Ludovico Bonan. Audiogmenter: a matlab toolbox for audio data augmentation. 2019.

[77] Justin Salamon and Juan Pablo Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE signal processing letters*, 24(3):279–283, 2017. ISSN 1070-9908.

[78] Tensorflow's Team. Tensorflow official web page, . URL https://www.tensorflow.org/. (accessed: 16.05.2021).

[79] PyTorch's Team. Pytorch official web page, . URL https://pytorch.org/. (accessed: 16.05.2021).

[80] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[81] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[82] PyTorch. Reproducibility, using pytorch, 2019. URL https://pytorch.org/docs/stable/notes/randomness.html. (Accessed: 17.10.2020).

[83] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

[84] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *Journal: arXiv preprint arXiv:1207.0580*, 2012.

[85] Pytorch Team. The documentation page on bceloss, . URL https://pytorch.org/docs/stable/generated/torch.nn.BCELoss.html#torch.nn.BCELoss. (accessed: 16.05.2021).

[86] https://scikit learn.org. Scikit-learn's page on the accurcay function:accuracy_score. URL https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html. (accessed: 13.06.2021).

[87] Torch Contributors. Torch save function documentation page, 2019. URL https://pytorch.org/docs/stable/generated/torch.save.html. (Accessed: 15.12.2020).

[88] Stefan Kahl. Birdnet demo, 2019. URL https://birdnet.cornell.edu/api/. (Accessed: 10.10.2020).

[89] Wildlife acoustics. Equipment supplier for the gear to nina, . URL https://www.wildlifeacoustics.com/. (Accessed: 4.11.2020).

[90] Wildlife acoustics. *SMX-NFC Night Flight Call Microphone*, . URL https://www.wildlifeacoustics.com/uploads/user-guides/SMX-NFC-Night-Flight-Call-Microphone-Specification.pdf. (Accessed: 4.11.2020).

[91] Wildlife acoustics. *Song Meter User Manual*, . URL https://www.wildlifeacoustics.com/uploads/user-guides/Song-Meter-Users-Manual.pdf. (Accessed: 4.11.2020).

[92] Wildlife acoustics. Frontex: Acoustic damping, . URL http://www.audio-constructor.com/index.php?main_page=product_info&cPath=67&products_id=369. (Accessed: 4.10.2020).

[93] Gyldendal Norsk Forlag AS. Ordnett's webpage. URL https://www.ordnett.no/. "Ordnett is a digital language and dictionary service" (Accessed: 17.06.2021).

[94] Pytorch Team. The documentation page on cuda semantics, . URL https://pytorch.org/docs/stable/notes/cuda.html. (accessed: 16.05.2021).

[95] Creative Commons. Attribution-noncommercial-sharealike 3.0 germany (cc by-nc-sa 3.0 de), . URL https://creativecommons.org/licenses/by-nc-sa/3.0/de/deed.en. (Accessed: 22.04.2021).

[96] Creative Commons. Attribution-sharealike 3.0 germany (cc by-sa 3.0 de), . URL https://creativecommons.org/licenses/by-sa/3.0/de/deed.en. (Accessed: 22.04.2021).

[97] Creative Commons. Attribution-noncommercial-noderivs 3.0, . URL https://creativecommons.org/licenses/by-nc-nd/3.0/. (Accessed: 22.04.2021).

[98] Creative Commons. Attribution-noncommercial-sharealike 3.0, . URL https://creativecommons.org/licenses/by-nc-sa/3.0/. (Accessed: 22.04.2021).

[99] Creative Commons. Attribution-sharealike 3.0, . URL https://creativecommons.org/licenses/by-sa/3.0/. (Accessed: 22.04.2021).

[100] Creative Commons. Attribution-noncommercial-sharealike 4.0 international (cc by-nc-sa 4.0), . URL https://creativecommons.org/licenses/by-nc-sa/4.0/. (Accessed: 22.04.2021).

[101] Creative Commons. Attribution-noncommercial-noderivatives 4.0 international (cc by-nc-nd 4.0), . URL https://creativecommons.org/licenses/by-nc-nd/4.0/. (Accessed: 22.04.2021).

Appendix

A1 Location and time of the recordings

This appendix aims to give information on when and where NINA made the recordings in 2016, 2017 and 2018. As the introduction chapter informed, are all three species mainly found in the area surrounding and at Finnmarksvidda, in North of Norway. Therefore was an area in Kautokeino, the western parts of Finnmarksvidda, chosen by NINA to do the recordings. Figure A1.1 shows a map with six of the eight locations are marked, while the last two are indicated on the map. In Table A1.1 can you see some details regarding each location; the coordinates of the recording site given in dd.ddddd°, which year(s) the recording was carried out there, and potential comments from NINA if there are any. In Table A1.1 one can also see that location six really were five separate locations but moved along a road as Figure A1.1 points out. Each site was chosen either to test or because it was a known or a potential breeding site for one or multiple of the three birds. Table A1.2 shows an overview of whether the different sites was believed to be a; breeding site (BS), a potential breeding site (PS), or a test site (TS) for each bird.

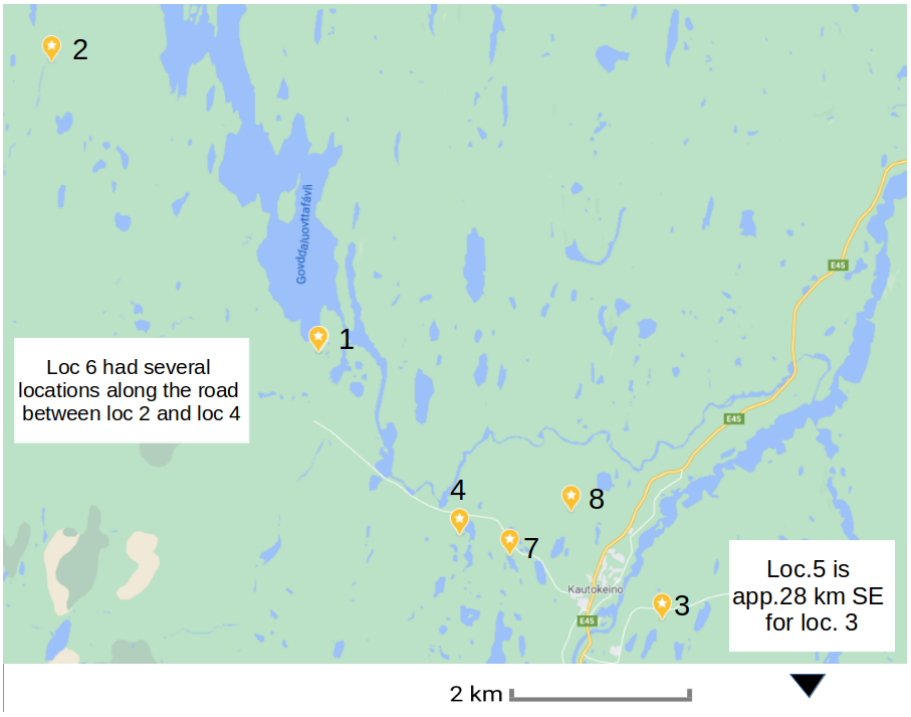


Figure A1.1: Map over the area where the recordings were carried out, with marks and numeration of the recording sits. The map was derived from google maps (at the 4'th of november).

Table A1.1: Overview of the recording locations with coordinates, year of recordings, and additional comments from NINA.

Location	Latitude °N	Longitude °E	Years with recordings	Comment
Loc1	69.08064	22.84273	2016, 2017, 2018	
Loc1ud	69.08081	22.84227	2018	20m from Location 1, without noise cover. Meant to test the difference in equipment.
Loc1b	69.08096	22.83392	2018	Supplement to Loc1, for covering the hole swamp area.
Loc2	69.15720	22.64518	2016, 2017, 2018	
Loc3	69.01016	23.09574	2016, 2017	
Loc4	69.03275	22.94640	2016, 2017, 2018	
Loc5	68.76405	23.29447	2016	Jack Snipe breeding site. Much traffic noise.
Loc6a	69.05347	22.88347	2016	Short recording period.
Loc6b	69.02665	22.98874	2016	Short recording period.
Loc6c	69.08185	22.79189	2016	Short recording period
Loc6d	69.10470	22.75046	2016	Short recording period.
Loc6e	69.09732	22.73927	2016	Short recording period.
Loc7	69.02711	22.98377	2017, 2018	'Hidden bog'
Loc8	69.03868	23.02890	2017	Potential Jack Snipe site.

Table A1.2: An overveiw over whether the different recording sites were breeding sites(BS), potential breeding sites (PS), or test sites (TS)

Location	Jack Snipe	Broad-billed Sandpiper	Spotted Redshank
Loc1	BS	BS	BS
Loc1ud	BS	BS	BS
Loc1b	BS	BS	BS
Loc2	BS	PS	TS
Loc3	PS	TS	TS
Loc4	PS	PS	BS
Loc5	BS	BS	PS
Loc6a	TS	TS	TS
Loc6b	TS	TS	TS
Loc6c	TS	TS	TS
Loc6d	TS	TS	TS
Loc6e	TS	TS	TS
Loc7	PS	BS	PS
Loc8	PS	TS	TS

Table A1.3 , Table A1.4, and Table A1.5 shows when the recordings started and ended at each site, where each year is represented by one of the tables. The equipment had a maximum duration it could record at a time, which was around 18 hours and 38 minutes. Therefore the recordings were not made 100 % continuously, but divided into several files when longer recordings where done. For recordings of whole days, the first 18 hours and 38 minutes were normally written to one file, and the remaining time of the day were written to another. For information about the equipment used on the sites, see Appendix A2.

Table A1.3: The different recording-times on the different locations for 2016.

2016	
Location:	Date and time (CET)
Loc1	03.06.2016 19:07 - 23:59
Loc1	04.06.2016 18:39 - 19:07
Loc1	05.06.2016 18:39 - 23:59
Loc1	06.06.2016 18:39 - 23:59
Loc1	11.06.2016 18:39 - 23:59
Loc1	12.06.2016 18:39 - 23:59
Loc1	13.06.2016 21:54 - 23:59
Loc1	14.06.2016 18:39 - 23:59
Loc1	15.06.2016 21:54 - 23:59
Loc2	03.06.2016 20:17 - 04.06.2016 18:39
Loc2	07.06.2016 00:00 - 18:39
Loc2	09.06.2016 18:39 - 23:59
Loc3	06.06.2016 18:39 - 23:59
Loc3	07.06.2016 18:39 - 23:59
Loc3	08.06.2016 18:39 - 23:59
Loc3	09.06.2016 20:21 - 23:59
Loc3	10.06.2016 18:39 - 23:59
Loc3	11.06.2016 18:39 - 23:59
Loc3	12.06.2016 18:39 - 23:59
Loc3	14.06.2016 16:31 - 23:59
Loc3	16.06.2016 18:39 - 23:59
Loc4	05.06.2016 18:39 - 23:59
Loc4	06.06.2016 18:39 - 23:59
Loc4	07.06.2016 18:39 - 23:59
Loc4	09.06.2016 18:39 - 23:59
Loc4	13.06.2016 18:39 - 23:59
Loc4	15.06.2016 00:00 - 17.06.2016 11:01
Loc5	04.06.2016 17:39 - 07.06.2016 20:45
Loc5	08.06.2016 00:00 - 12.06.2016 23:59
Loc5	13.06.2016 04:39 - 14.06.2016 01:18
Loc5	14.06.2016 04:59 - 15.06.2016 01:37
Loc5	15.06.2016 11:59 - 23:59
Loc6a	04.06.2016 19:48 - 23:59
Loc6a	05.06.2016 18:39 - 08.06.2016 18:57
Loc6b	09.06.2016 19:55 - 11.06.2016 10:17
Loc6c	11.06.2016 10:52 - 13.06.2016 19:09
Loc6d	13.06.2016 19:34 - 15.06.2016 11:38
Loc6e	15.06.2016 11:40 - 17.06.2016 12:31

Table A1.4: The different recording-times on the different locations for 2017.

2017	
Location:	Date and time (CET)
Loc1	03.06.17 18:31 - 18.06.17 23:59
Loc2	03.06.17 20:46 - 20.06.17 19:09
Loc3	04.06.17 13:57 - 19.06.17 23:59
Loc4	04.06.17 12:37 - 20.06.17 23:59
Loc7	04.06.17 13:17 - 20.06.17 23:59
Loc8	05.06.17 12:32 - 20.06.17 23:59

Table A1.5: The different recording times for the different locations in 2018.

2018	
Location:	Date and time (CET)
Loc1D	15.06.18 21:29 - 20.06.18 17:35
Loc1UD	15.06.18 21:29 - 17.06.18 12:31
Loc1v	17.06.18 13:34 - 20.06.18 17:10
Loc2	16.06.18 18:03 - 20.06.18 17:15
Loc4	16.06.18 14:36 - 20.06.18 18:15
Loc7	16.06.18 14:33 - 20.06.18 17:17

A2 Equipment used for the recordings

The following list and information was provided by NINA, regarding what equipment they used for carrying out the recordings.

The equipment used on the different sites, for all three years were mostly the same with the exception of location 6, from 2016 and location 8 in 2017. Here was another microphone used. Otherwise, the only difference was that all the recording sites, except location 1ud in 2018 had an additional windscreen/noise-cover added to protect the microphone. All the equipment NINA used was delivered by *Wildelife Acoustics* [89], and should been listed below.

List of the equipment used:

- Plexiglas plate
- SMX-NFC microphone, was the microphone used on the most locations, and was the one in the images of Figure A2.2 and Figure A2.1. For more information look at [90].
- SMX-II microphone, was the microphone used on location 6 in 2016, and location 8 in 2017.
- The recorder was a SongMeter Model SM2, for more information look at [91].
- FRONTEX (Acoustic Damping), only used in 2018 at location 1, for more information look at [92]. + 3D printed plastic roods to put it on.
- 32GB SD card
- 4 alkaline batteries type C (LR4)
- Aluminum sticks, to attach the microphone

The first type of microphone listed, SMX-NFC, was attached to a Plexiglas plate, while the other was not. Then both the recorder and the microphone was attached to an aluminium poll, around 3-3.5 meter above ground. Figure A2.1 and Figure A2.2 shows two different pictures of the setup. Both were taken by John Atle Kaalaas, which have given permission to use them in this report. Figure A2.2 was from the second location in 2018, were you can see the windscreen on top. While Figure A2.1 was taken at the seventh location, in 2017.

Figure A2.1: Photo taken of the set up at location 7 in 2017. (Private picture from John Atle Kålås, NINA, used in this report with permission from him.)

Figure A2.2: Photo taken of the set up at location 2 in 2018.(Private picture from John Atle Kålås, NINA, used in this report with permission from him.)

A3 Birds in the area

To get a picture of which birds coexist in the same area as the Jack Snipe, Spotted Redshank, and Broad-billed Sandpiper (a big part of the the bioacoustic landscape) were a list of all the birds in the area gathered. To do so was the Norwegian site *artsdatabanken* used [70], which in English is called The Norwegian Biodiversity Information Centre[70]). One of the services at this site is a map that looks at species occurrences in Norway registered on their site. To narrow down the search, is it possible to apply different types of filters. Since we are interested in the area of recordings, around the same time that the recordings were taken was the following filters used; "Kautokeino, Troms and Finnmark"(location), "May-July"(moths) and "bird" (group of species).

This resulted in a list of 172 items, eight of which were a replica of another item or a bird family (not a species), leaving 164 bird species. Out of the 164 had all the birds, everything from 1 to 416 registered sightings. To eliminate the ones that have barely been observed and perhaps just have flown through were the 61 birds with less than five recordings disregard, leaving 101 birds. These were then sent to the ornithologist at NINA, John Atle Kaalaas, which looked through them and grouped them into five different groups. Each group were grouped based on there anticipated level how dominant their vocalisations the group of birds is assumed to be in NINA's recordings in June in Kautokeino of 2016-2018. Below can one see an explanation of each group and how many birds belong to the group.

- Group 1: Quite little anticipated activity in the recordings (51 of the birds on the list).
- Group 2: Little anticipated activity in the recordings (23 of the birds on the list).
- Group 3: They will be heard frequently, but not that often and therefore not have such a dominant vocalisation (18 of the birds on the list).
- Group 4: The species that dominates the soundscape (7 of the birds on the list).
- Group 5: The species that dominates the soundscape the most (5 of the birds on the list).

The complete list of the 106 birds can be found in Table A3.1, which list each bird's scientific name, common name [3], number of observations and the group it belongs to. When looking up the Jack Snipe and the Broad-billed Sandpiper we can see that they belong to group three, while the Spotted Redshank belongs to group four.

[3]The common name was found by using the Norwegian site ordnett.no[93](a dictionary site by one of the larger publishing houses in Norway), and mostly looking up the Norwegian common name or the scientific name. In some cases was there multiple name to choose between, so it might not be 100% correct.

Table A3.1: An overview of potential birds in the area, collected from [70], divided into groups depending on how dominant there vocalisations are assumed to be in the Kautokeino recordings done by NINA.

Scientific name	Common name	Number of observations	Group
Phylloscopus Trochilus	Willow Warbler	324	5
Tringa Glareola	Wood Sandpiper	271	5
Luscinia Svecica	Bluethroat	256	5
Anthus Pratensis	Meadow Pipit	197	5
Motacilla Flava	Yellow Wagtail	209	5
Fringilla Montifringilla	Brambling	205	4
Turdus Iliacus	Redwing	191	4
Acanthis Flammea	Redpoll	180	4
Tringa Erythropus	Spotted Redshank	172	4
Numenius Phaeopus	Whimbrel	148	4
Gallinago Gallinago	Common Snipe	153	4
Calcarius Lapponicus	Lapland Bunting	180	4
Pluvialis Apricaria	Golden Plover	173	3
Lagopus Lagopus	Willow Grouse	151	3
Cuculus Canorus	Cuckoo	98	3
Emberiza Schoeniclus	Reed Bunting	141	3
Cygnus Cygnus	Whooper Swan	261	3
Grus Grus	Crane	38	3
Acanthis Hornemanni	Arctic Redpoll	31	3
Larus Canus	Common Gull	161	3
Oenanthe Oenanthe	Wheatear	145	3
Corvus Cornix	Crow	133	3
Charadrius Hiaticula	Ringed Plover	124	3
Calidris Temminckii	Temminck's Stint	103	3
Turdus Pilaris	Fieldfare	102	3
Lymnocryptes Minimus	Jack Snipe	86	3
Phoenicurus Phoenicurus	Redstart	47	3
Calidris Falcinellus	Broad-billed Sandpiper	45	3
Gavia Arctica	Black-throated Diver	96	3
Turdus Philomelos	Song Thrush	15	3
Sterna Paradisaea	Arctic Tern	159	2
Corvus Corax	Raven	126	2
Actitis Hypoleucos	Common Sandpiper	121	2
Limosa Lapponica	Bar-tailed Godwit	77	2
Chloris Chloris	Greenfinch	37	2
Tringa Totanus	Redshank	36	2
Parus Major	Great Tit	28	2

Anthus Trivialis	Tree Pipit	27	2
Prunella Modularis	Accentor	15	2
Ficedula Hypoleuca	Pied Flycatcher	13	2
Tringa Nebularia	Greenshank	103	2
Calidris Alpina	Red-backed Sandpiper	91	2
Motacilla Alba	White wagtail	83	2
Acrocephalus Schoenobaenus	Sedge Warbler	69	2
Asio Flammeus	Short-eared Owl	55	2
Chroicocephalus Ridibundus	Black-headed Gull	48	2
Charadrius Morinellus	Dotterel	26	2
Cinclus Cinclus	Eurasian Dipper	16	2
Turdus Torquatus	Ring Ouzel	8	2
Phylloscopus Collybita	Chiffchaff	7	2
Hirundo Rustica	Barn Swallow	7	2
Sylvia Borin	Garden Warbler	7	2
Muscicapa Striata	Spotted Flycatcher	5	2
Bucephala Clangula	Goldeneye	239	1
Mareca Penelope	European Wigeon	269	1
Anas Crecca	European Teal	222	1
Clangula Hyemalis	Long-tailed Duck	165	1
Falco Columbarius	Merlin	160	1
Mergus Serrator	Red-breasted Merganser	129	1
Anas Platyrhynchos	Mallard	128	1
Anas Acuta	Pintail	95	1
Mergus Merganser	Goosander	85	1
Melanitta Nigra	Black Scoter	84	1
Riparia Riparia	Sand Martin,	65	1
Melanitta Fusca	Velvet Scoter,	65	1
Falco Tinnunculus	Kestrel	54	1
Aythya Marila	Scaup	41	1
Gavia Stellata	Black-throated Diver	32	1
Anthus Cervinus	Red-throated Pipit	28	1
Larus Argentatus	Herring Gull	26	1
Plectrophenax Nivalis	Snow Bunting	26	1
Mergellus Albellus	Smew	24	1
Lagopus Muta	Common Ptarmigan	13	1
Bombycilla Garrulus	Waxwing	13	1
Poecile Montanus	Willow Tit	9	1
Anser Anser	Graylag Goose	5	1
Aythya Fuligula	Tufted Duck	416	1
Aquila Chrysaetos	Golden Eagle	332	1

Phalaropus Lobatus	Red-necked Phalarope	194	1
Stercorarius Longicaudus	Long-tailed Skua	170	1
Calidris Pugnax	Ruff	164	1
Buteo Lagopus	Rough-legged Hawk	124	1
Pica Pica	Magpie	81	1
Passer Domesticus	House Sparrow	78	1
Falco Rusticolus	Gyrfalcon	58	1
Anser Fabalis	Bean Goose	57	1
Motacilla Flava Thunbergi	Western Yellow Wagtail	43	1
Delichon Urbicum	House Martin	39	1
Alauda Arvensis	Skylark	23	1
Poecile Cinctus	Siberian Tit	22	1
Lanius Excubitor	Great Grey Shrike	22	1
Phylloscopus Borealis	Arctic Warbler	18	1
Haliaeetus Albicilla	Sea Eagle	17	1
Surnia Ulula	Northern Hawk Owl	15	1
Saxicola Rubetra	Whinchat	10	1
Vanellus Vanellus	Lapwing	10	1
Sturnus Vulgaris	European Starling	10	1
Hydrocoloeus Minutus	Little Gull	8	1
Dryobates Minor	Lesser Spotted Woodpecker	7	1
Larus Marinus	Great Black-backed Gull	7	1
Circus Cyaneus	Hen Harrier	6	1
Passer Montanus	Tree Sparrow	5	1
Perisoreus Infaustus	Siberian Jay	5	1
Pinicola Enucleator	Pine Grosbeak	5	1

A4 Training data

This appendix lists the data from NINA that was reveiwed and labelled to make the training-set.

Location 1:

- 04.06.2017 00:00-23:59
- 05.06.2017 00:00-23:59
- 06.06.2017 00:00-23:59
- 07.06.2017 00:00-23:59
- 08.06.2017 00:00-23:59
- 09.06.2017 00:00-23:59
- 10.06.2017 00:00-23:59
- 11.06.2017 00:00-23:59
- 12.06.2017 00:00-23:59
- 13.06.2017 00:00-23:59
- 14.06.2017 00:00-23:59
- 15.06.2017 00:00-23:59

Location 3:

- 11.06.2016 18:39-23:59
- 14.06.2016 16:31-23:59
- 05.06.2017 18:39-23:59
- 09.06.2017 18:39-23:59

Location 5:

- 11.06.2016 18:39-23:59
- 11.06.2016 00:00-18:38 **
- 12.06.2016 16:31-23:59
- 11.06.2016 00:00-18:38 **
- 14.06.2016 19:37-23:59
- 15.06.2016 11:59-23:59

Location 6:

- (location 6a) 06.06.2016 00:00-09:00
- (location 6b) 10.06.2016 09:00-18:38

- (location 6d) 14.06.2016 18:39-23:59
- (location 6e) 16.06.2016 18:39-23:59

Location 8:

- 06.06.2017 00:00-18:38
- 07.06.2017 00:00-18:38

** = Added after seeing that BS had an minority of data points in the data-set, which is why the BS class was of focus when labelling these files.

A5 Code parameters for the system

This appendix is meant as supplementary information and overview, to what is already described in section 3, on how the different version of the system presented in section 4 were built. In not explicitly told other wise were the parameters used to build/test the system the same. In the following does # symbolise a comment.

For both the models were the audio divided into one second long segments, which was then transformed into either mel-spectrogram or pure spectrograms.

For the Mel-spectrograms: (Giving the shape 116×116)

```
Sampling rate= 16 kHz
fmin=500
fmax=6000
n_mels=116
hop_length=138
n_fft=552
```

For spectrograms: (Giving the shape 224×220)

```
Sampling rate= 16 kHz
n_fft=552
hop_length=138
```

After splitting the training data 30% validation 70% training, it was iterated over the original data (not augmented or down sampled) to extract the standard and mean of the data, which latter was used to feature scaling the data, see section 3.4.

The random seed set for reproducibility:

```
random_seed=0
```

For the data loader used: (pin_memory=True is recommended by PyTorch [94] when using cuda)

```
batch_size=16
num_workers=8,
shuffle=True # for training data
shuffle=False # validation
drop_last=False
pin_memory=True
```

For making both models were the pre-trained version of ResNet34 loaded in, with wights. To set up the model the both version of the system used:

```
Drop_out=0.45 # only in the last fully connected layer
Number_of_output_nodes=3
```

In addition to PyTorch's *xavier_ uniform_ ()* function to reinitialise the last layers weights recommended by Sigmoid. In production/testing was the Sigmoid function used at the end of the model, but in the training BCEWithLogitsLoss was used, and the Sigmoid function were just used at the data before calculating the accuracy. Nevertheless, due to that the Mel-spectrogram and spectrogram had different shapes, the first convolutional layer had to be redefine a bit differently for each.

For the Mel-spectrogram model:

```
num_channel=1
output_first_layer=64
kernal_size=(7,7)
padding=(1,1)
stride=(1,1)
```

For the spectrogram model:

```
num_channel=1
output_first_layer=64
kernal_size=(7,7)
padding=(3,1)
stride=(2,2)
```

For training the Mel-spectrogram model:

```
# Rounds with the training data
epochs=50

# For the optimizer (SDG)
learning_rate=0.005
momentum=0.9
weight_decay=0.012

# The scheduler
step_size=4
gamma=0.45
```

For training the spectrogram model:

```
# Rounds with the training data
epochs=50

# For the optimizer (SDG)
learning_rate=0.005
momentum=0.9
weight_decay=0.015

# The scheduler
step_size=4
gamma=0.45
```

When it comes to the post-processing step were a duration of 2 data points used as a minimum duration of a vocalisation i both version of the system (after a bit of testing). However, the threshold of acceptance, which was set per class, were a bit different for one system to the other. This threshold was set based on NINA's data, and what gave best results in test 1 and test 2(section 3.9).

For the Mel-spectrogram model:

```
Threshold=[0.9,0.5,0.94] # JS, SR, BS
```

For the spectrogram model:

```
Threshold =[0.96,0.8,0.95] # JS, SR, BS
```

However, when testing the system using data from publicly available dataset the same threshold was tested. Nevertheless, as we can see in section 4.3.2 were the system tested using a threshold of 0.5 in every class as well.

A6 Data collected from online sources

This appendix is divided into three sections, each belonging to separate databases that were used to extract recordings from, to be used for testing. Each section will list the recordings in the matter requested by the cite or the license the data is made available on. The databases used are the; *The Museum für Naturkunde Berlin* (tierstimmenarchiv) [67], *Macaulay Library*[21] and *Xeno-canto* [22].

A6.1 The Berlin museum:

Four sound clips were collected from tierstimmenarchiv/the Berlin museum [67]. Below you can find the recorders name, the filename, and the licence it is published under, together with a link to the license:

- Recorded by: Krey, Winfried
  Filename: Lymnocryptes_minimus_Kr0100_08
  Licence: CC BY-NC-SA, [95].
- Recorded by: Krey, Winfried
  Filename: Lymnocryptes_minimus_Kr0101_01
  Licence: CC BY-NC-SA, [95].
- Recorded by: Krey, Winfried
  Filename: Lymnocryptes_minimus_Kr0101_03
  Licence: CC BY-NC-SA [95].
- Recorded by: Krey, Winfried
  Filename: Limicola_falcinellus_Kr0058_09
  Licence: CC BY-SA [96].

A6.2 Macaulay Library

The following recordings were used from the Macaulay Library at the Cornell Lab of Ornithology[21]:

ML207373, ML121526901, ML207610, ML205668, ML205679, ML205669, ML205683, ML207611, ML304235, ML207389, ML207382, ML207612, ML304237, ML207615, ML205674, ML207390, ML207372, ML205676, ML207368, ML205684

A6.3 Xeno-canto

The main source of the "database-test" was Xeno-canto. Xeno-canto is a website for sharing bird sounds, across the world. All the sounds collected from this site is under one of three creative commons licenses; Attribution-NonCommercial-NoDerivs [97], Attribution-NonCommercial-ShareAlike [98], or Attribution-ShareAlike [99]. Which creative common licenses that each recording fall under, can be found by looking up the sound clip on xeno-canto's website [22].

Following Xeno-canto's wishes when using data from their site, one can find all recordings used listed below with the name of the recorder and the XC-number. By replacing the ... in "www.xeno-canto.org/.../download" with one of the XC-number, will the url take you to a download site of that recording.

(**For the Broad-billed Sandpiper:**)

Recorded by: Patrik Åberg, XC-number:58590 Licence: CC BY-NC-SA 3.0 [98]
Recorded by: Lars Edenius, XC-number:417176 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Lars Edenius, XC-number:564395 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Philippe J. DUBOIS, XC-number:492001 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Tomek Tumiel, XC-number:189033 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Jonas Buddemeier, XC-number:386224 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Antero Lindholm, XC-number:356974 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Stanislas Wroza, XC-number:385901 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Tero Linjama, XC-number:342360 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Patrik Åberg, XC-number:58592 Licence: CC BY-NC-SA 3.0 [98]
Recorded by: Lars Edenius, XC-number:562504 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Lars Edenius, XC-number:562289 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Lars Edenius, XC-number:614620 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Stein Ø. Nilsen, XC-number:298254 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Lars Edenius, XC-number:562569 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Jarmo Pirhonen, XC-number:318951 Licence: CC BY-NC-SA 4.0 [100]

(**For the Jack Snipe:**)

Recorded by: Albert Lastukhin, XC-number:431130 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Marcin Sołowiej, XC-number:211051 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Lars Edenius, XC-number:554829 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Marcin Sołowiej, XC-number:202908 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Teet Sirotkin, XC-number:554679 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Stein Ø. Nilsen, XC-number:492150 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Lars Edenius, XC-number:556118 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Stein Ø. Nilsen, XC-number:255123 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Gerard Kenter, XC-number:240586 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Uku Paal, XC-number:552331 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Patrik Åberg, XC-number:42189 Licence: CC BY-NC-SA 3.0 [98]
Recorded by: Lars Edenius, XC-number:416429 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Albert Lastukhin, XC-number:431008 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Ilkka Heiskanen, XC-number:216726 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Ulf Elman, XC-number:610671 Licence: CC BY-NC-ND 4.0 [101]
Recorded by: Patrik Åberg, XC-number:27087 Licence: CC BY-NC-SA 3.0 [98]
Recorded by: maciej sobieraj, XC-number:553674 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Patrik Åberg, XC-number:27086 Licence: CC BY-NC-SA 3.0 [98]
Recorded by: Patrik Åberg, XC-number:42188 Licence: CC BY-NC-SA 3.0 [98]
Recorded by: Patrik Åberg, XC-number:166406 Licence: CC BY-NC-SA 3.0 [98]
Recorded by: Tero Linjama, XC-number:342362 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Stein Ø. Nilsen, XC-number:265128 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Lars Lachmann, XC-number:133318 Licence: CC BY-NC-SA 3.0 [98]

(**For the Spotted redshank:**)
Recorded by: Lars Edenius, XC-number:370548 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Lars Edenius, XC-number:370549 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Tero Linjama, XC-number:342884 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Lars Edenius, XC-number:480959 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Peter Boesman, XC-number:281139 Licence: CC BY-NC-ND 4.0 [101]
Recorded by: Stein Ø. Nilsen, XC-number:181668 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Ulf Elman, XC-number:490175 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Ulf Elman, XC-number:490176 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Alan Dalton, XC-number:577272 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Lars Edenius, XC-number:562591 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Lars Edenius, XC-number:504669 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Peter Boesman, XC-number:281140 Licence: CC BY-NC-ND 4.0 [101]
Recorded by: Lars Edenius, XC-number:562287 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Lars Edenius, XC-number:504692 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Uku Paal, XC-number:476690 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Lars Edenius, XC-number:562505 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Albert Lastukhin, XC-number:484261 Licence: CC BY-NC-SA 4.0 [100]
Recorded by: Lars Edenius, XC-number:562506 Licence: CC BY-NC-SA 4.0 [100]

Ingrid Grimstad

Classification of Wader Birds' Vocalisations using Neural Networks

NTNU
Norwegian University of
Science and Technology