Norwegian University
of Life Sciences

Master’s Thesis 2021 30 ECTS
Faculty of Science and Technology

Determining the origin of impulsive noise events using wireless sound sensors

Fabian Nemazi
Master of Science in Data Science

Preface

This thesis was a use case for Politiets Nasjonale Beredskapsenter in collaberation with Soundsensing A/S. It was written at the Norwegian University of Life Sciences in the spring of 2021 and marked the end of my study time at NMBU.

Firstly, I would like to thank my supervisor from NMBU, Associate Professor Ulf Geir Indahl. Your guidance and help provided me significantly. Furthermore, I owe a special thanks to my supervisor from Soundsensing A/S, Jon Nordby. His willingness to respond to my messages at any given day or time was invaluable.

Moreover, I thank my great friend Meron Haile for taking the time to review my thesis and provide input to improve. I also owe a special thank you to Emil Skar. He must've had 22 conversations with me, sparring over solutions and helping me immensely.

Lastly, I thank my mother and brother for their support throughout my years at NMBU. I would like to emphasize the influence my brother had guiding me in the right direction in life. I firmly believe I would not be where I am without him. I dedicate this thesis to him.

Oslo, 01.06.2021

Fabian Nemazi

Abstract

Environmental noise is a serious and growing pollution problem, with over 100 million people affected in Europe just by road traffic. Measuring and monitoring noise using a wireless sensor network is therefore getting more common. However, many scenarios consist of multiple sources of noise, with each source varying over time. That presents a challenge in determining which noise source is the cause of high noise levels, which is necessary to identify to reduce the problem.

This work investigates how to identify the source of impulsive noise events using a pair of wireless noise sensors. One sensor is placed at a known noise source, and another sensor is placed at the noise receiver. Machine learning models receive data from the two sensors and estimate whether a given noise event originates from the known noise source or another source.

In order to avoid privacy issues, the approach uses on-edge preprocessing that converts the sound into privacy compatible spectrograms and sound level representations. These different representations of sound can be stored, combined, and processed at a central server.

The system was evaluated at a shooting range and explosives training facility, using data collected during noise emission testing. The combination of convolutional neural networks with cross-correlation achieved the best results (denoted bundled model). We created multiple bundled models with different stacking of delta-spectrograms. The best model detected 70.8% of the impulsive noise events and correctly predicted 90.3% of the noise events in the optimal trade-off between recall and precision score.

Sammendrag

Miljøstøy er et seriøst og økende forurensingsproblem med over 100 millioner mennesker påvirket i Europa av veitrafikk. Målinger og monitorering av støy ved hjelp av trådløse sensor nettverk er derfor stadig vanligere. Mange scenarioer består derimot av flere kilder med støy, hvor hver kilde varierer i tid. Dette presenterer en utfordring når det gjelder å avgjøre hvilken støykilde som er årsaken til de høye støynivåene, som er nødvendig å identifisere for å redusere problemet.

Dette arbeidet undersøker hvordan støykilden til impulsive støybegivenheter kan identifiseres ved bruk av et par trådløse støy sensorer. En sensor er plassert hos en kjent støykilde, og en annen er plassert hos støymottakeren. Maskinlærings modeller mottar data fra de to sensorene og estimerer hvorvidt en gitt støy begivenhet kommer fra den kjente støykilden eller noe annet.

For å unngå problemer med personvern bruker denne tilnærmingsmåten 'on-edge' preprosessering som konverterer lyd til personvernskompatible spektrogrammer og lydnivå representasjoner. Disse ulike representasjonene av lyd kan bli lagret, kombinert og prosessert hos en server.

Systemet ble evaluert på skytebaners og eksplosivers treningsfasilitet, ved bruk av data innsamlet fra testingen av støy forurensing. Kombinasjonen av konvolusjonalt nevrale nettverk sammen med kryss-korrelasjon oppnådde best resultater (betegnet samlet modell). Vi skapte flere samlede modeller med ulike stablinger av ulike delta-spektrogrammer. Den beste samlede modellen detekterte 70.8% av de impulsive støy hendelsene og predikerte korrekt på 90.3% av hendelsene identifiserte som støy i den optimale kompromisset mellom precision og recall score.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Noise

”Noise makes no good, good makes no noise” said Saint Vincent de Paul, a priest who dedicated his entire life helping the poor. Some would agree, and some would argue that noise is unavoidable. Regardless, noise has been an increasing issue over the years. With the steadily more urbanizing, recreational activities, and humans implementing new technology, noise is here to stay. The increasing noise caused by humans has led to the discovery of noise’s effect on health.

Around 50% of humans in several European countries are regularly exposed to noise levels of 55 decibels or higher during the day. In Oslo, a total of 508 400 people are exposed to day-evening-night average sound levels of 55 dB or higher from road traffic [1]. According to Worlds Health Organization (WHO), this has negative impacts on health [2]. European Economic Area (EEA) has estimated that long-term exposure to environmental noise causes around 12000 premature death each year across Europe [2].

Noise and sound are for many inseparably. Noise is perceived differently from person to person. Noise is, therefore, a very subjective term. It can be described as any unwanted sound. Humans detect these sounds like a less pleasant sensation and often try to screen it out [3].

The emission of noise from sources has been monitored and regulated by the EU for many years. In 2002, the introduction of Environmental Noise Directive (END) sought to monitor EU regulations’ effectiveness by assessing environmental noise at the Member State level (members of the EU). That meant that if measured noise at a given source preceded the given limit, actions had to be taken to reduce exposure. Since then, noise maps have been created every fifth year and cover all urban areas, railways, major roads, and airports over a specific size [4].

1.2 Monitoring noise with wireless sensor networks

An increasing number of cities have deployed networks of sound sensors. Generally, the sensors have been positioned in areas of interest. Correspondingly, this permits control of which districts having high emission of noise. There have been high costs related to sensors that deal with these issues. However, in recent years a low-cost sound sensor with dense coverage has emerged on the market, and it can be designed to operate wirelessly.

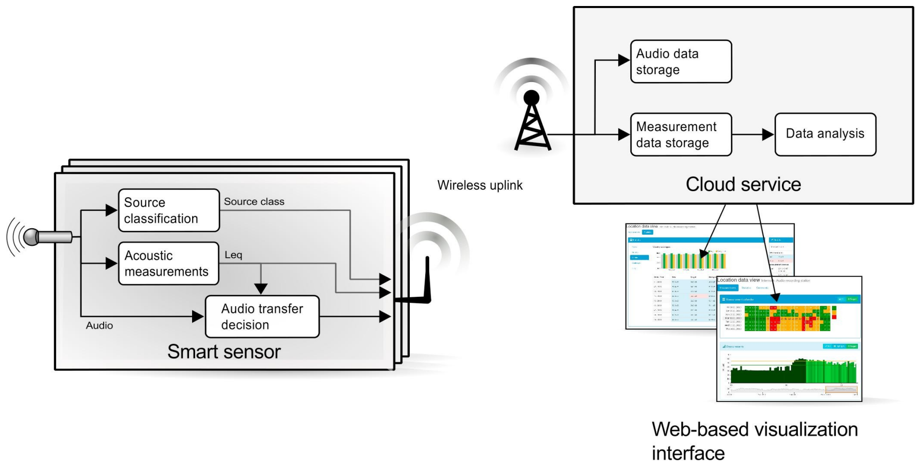


__Figure 1.1:__ Building block diagram of a noise monitoring system [5]. Illustration of how the sound registered is sent up to the cloud.

This typical noise monitoring system includes smart sensors that connect through a wireless uplink to a cloud service. The smart sensors have a measurement microphone and single-board computer, which could be described as a complete computer built on a single circuit board. This smart sensor sends out data via a wireless transition unit [5]. It is privacy issues regarding recording. However, this smart sensor processes the ability to prepossess and classify on the sensor. This ability could help to alleviate the privacy issue.

1.3 PNB

This thesis is regarding a feasibility study for Soundsensing A/S and Politiets Nasjonale Beredskapssenter (PNB). Soundsensing is a start-up company that specializes in environmental noise data. They have created a sensor device called the ”Soundsensing Noise Monitor”, a noise monitor. In combination with machine learning which analyzes collected data, its purpose is to differentiate between various noise sources.

PNB has recently moved into a new training base, where the police’s special force team undergoes their training. As this is the special force team, their training includes simulation of possible terror and high-threat attacks. Furthermore, they use different weapon types. These weapons vary from hand weapons with and without silencers to bombs with distinct amounts of explosives.

The issue surrounding this training facility is the settlement laying approximately 1 kilometer away. Accordingly, the municipality has expressed a desire to have a solution that tracks noise generated by PNB and its effect on the settlement.

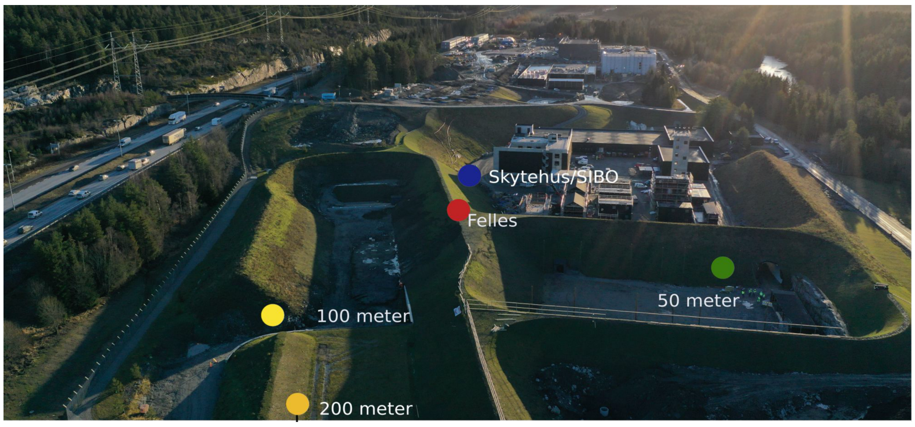


Figure 1.2: *Picture of the training facility [6]. Facility consists of three different shooting ranges (all with different length) and one houses for explosives training.*

Figure 1.2 displays the appearance of the training facility. PNB and the municipality have increased focus on limiting noise emissions. Therefore the facility is built in a specific manner where sound is contained. They have collaborated with acousticians to achieve the best noise reduction solutions possible. Figure 1.2 shows how each shooting range is built into the ground, with hills surrounding the entire field. This makes the sound travel in the shooting direction and minimizes the spread of sound.

Soundsensing, in collaboration with PNB, placed professional sound recorders around the facility and the settlement during the noise emission testing. During the data collection, the activity tested at PNB was intended to be 'worst-case' scenarios, where one can research whether this affects the settlement close by. These sound recorders will work as temporary stand-ins for sound sensors until a

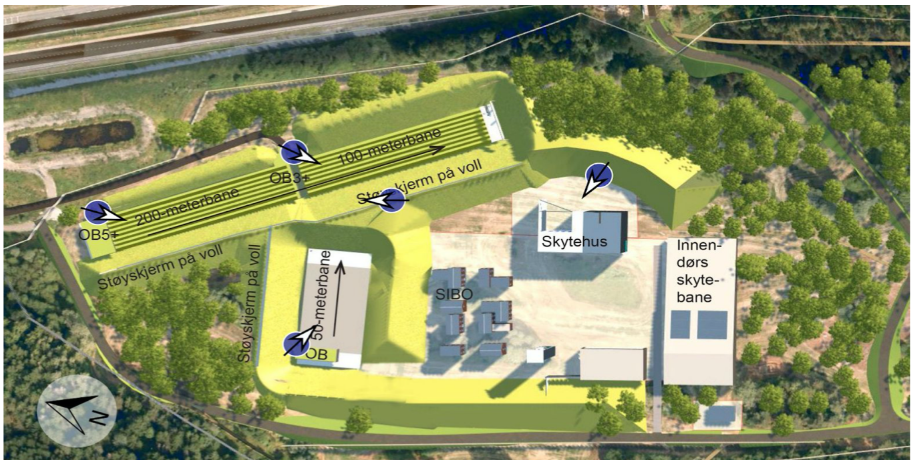


Figure 1.3: *Graphics showing the different sound recorders placement around the facility. [6]. In total five different sound recorders were placed at PNB.*

potential solution is achieved.

1.4 Problem statement

To summarize, PNB and the municipality have built the training facility such that noise should not negatively affect the nearby settlement. However, they are interested in monitoring their actual noise emissions and have collaborated with Soundsensing A/S. In a trial project, sound recorders were placed around the facility and recorded sound data under 'worst-case' scenarios. This allows analyzing how noise from PNB spreads to the settlement. Since a residential area also has many other noise sources, it is critical to differentiate between noise originating from PNB and noise originating from other sources.

That above distills the following research question to be addressed and explored in this thesis:
Is it possible to detect and track origin of impulsive noise events with the help of multiple wireless sound sensors?

Chapter 2

Theory

2.1 Physics of sound

2.1.1 Sound

Sound is a change in the air pressure, moving or swinging back and forth in a regular rhythm around the atmospheric pressure [7]. Humans are familiar with the concept of sound through *hearing*, but that is a perception of sound. More specifically, the sound is a wave that in many instances operates as a periodic wave [8]. A periodic wave is a wave with a continuous repeating pattern. This pattern describes the wavelength and frequency [9].

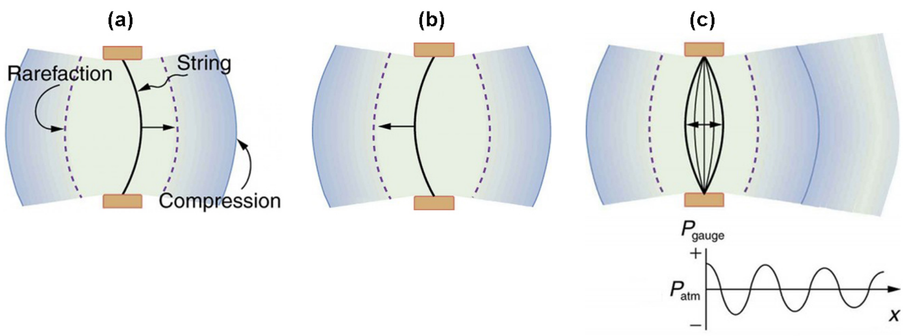


Figure 2.1: *A vibrating string producing a sound wave[8].*

Figure 2.1 illustrates how the string oscillates back and forth and transfers energy to the air. Some small part of the strings compresses and expands the surrounding air, which creates slightly higher and lower local pressures. It could be further explained in-depth:

a) The vibrating string moves to the right and expands the air behind it.

b) When the string moves to the left, it creates another compression and rarefaction as the one in a).

c) After a series of many vibrations, multiple compressions and rarefactions are moving away from the string as a sound wave. The graph shows gauge pressure (pressure relative to atmospheric pressure) versus distance from the source.

2.1.2 Amplitudes

The amplitude of a periodic wave could be expressed as the maximum difference of the wave's waves' extreme values (minimum or maximum value of a wave) over one period. That is usually the distance between the top and bottom of the curve. The sound wave could either have a large amplitude or a low amplitude. Large amplitude signifies high sound levels, and low amplitude would signify the opposite; low sound levels (quiet) [8]. No matter the size of the amplitude, it decreases with distance from the source.

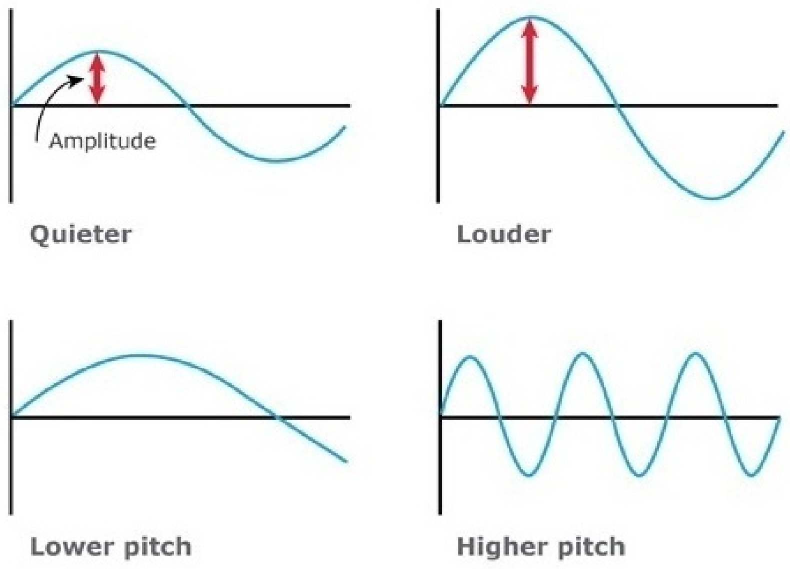


Figure 2.2: *Visualization of what could be described as amplitudes*

[3].

2.1.3 Frequency

The measuring method for a unit of frequency is Hertz (Hz). Hz describes the number of cycles repeated per second, where one cycle per second equals 1 Hz. The human ear can normally hear sound frequencies between 20 and 20 000 Hz [10]. Each sound wave has a length that depends on the frequency and the wave's speed in a specific direction. This frequency is defined more precisely by the period, measured by the time it takes to repeat a cycle. Typical high-frequency sound has short wavelengths and contains low energy, while low-frequency sounds have long wavelengths and contain high energy [11]. That could be observed in figure 2.2, where the lower pitch has a longer wavelength.

2.1.4 Sounds speed, propagation and absorption

In section 2.1.1 on page 6 we established that sound is a change in the air pressure, I.E., pressure waves traveling through the air. The frequency is the *how often*, and speed is the quantity that defines *how fast*. Speed is defined as the distance that a point on a wave travels per unit of time [12]. The speed could therefore be elaborated as $speed = \frac{distance}{time}$. There are conditions related to the speed of a sound wave. One is the property of the medium in which the wave is propagating [13]. For example, a sound wave will travel faster in a less dense material than a denser material [14]. The following will affect the speed (moving through the air): [15]:

- Coherence between density and pressure, could, due to the temperature, affect the speed of sound.
- The motion of the medium, meaning the wind. If the motion of the moment is in a tailwind, it will be further transported.
- The medium's resistance to deformation at a given rate, which affects the rate at which the sound is attenuated.
- The humidity, which is the result of water vapor being present in air.

Furthermore, sound waves propagating could be absorbed along the way. Sound absorption is a measure of energy reduction when a sound wave propagates through a material. When a sound wave spreads, it could strike the surface of a material. If this occurs, the wave is either reflected or penetrates the material [7]. Therefore, sound waves are generally absorbed by objects when spreading [8]. The speed of sound depends on temperature and humidity and is approximated by the following equation:

$$c = \sqrt{\gamma * R * T} \tag{2.1}$$

Here $\gamma = \frac{cp}{cv}$ is the ratio of specific heats, R is the specific gas and T is the temperature. Both γ and R depends on the composition of gas, which includes humidity in the air. Based on equation (2.1), one can derive the following table:

Temp (°C)	Speed of sound (m/s)								
	Relative Humidity (%)								
	10	20	30	40	50	60	70	80	90
-15	322.23	322.24	322.25	322.26	322.27	322.28	322.29	322.30	322.31
-5	328.43	328.45	328.47	328.49	328.51	328.53	328.55	328.57	328.59
5	334.52	334.56	334.61	334.65	334.70	334.74	334.79	334.83	334.88
15	340.52	340.61	340.7	340.79	340.88	340.97	341.06	341.15	341.24
25	346.44	346.62	346.79	346.96	347.13	347.30	347.47	347.65	347.82
30	349.38	349.62	349.85	350.08	350.31	350.55	350.78	351.01	351.24

Table 2.1: *The table shows the speed of sound as a function of temperature and humidity. The upper and lower temperatures are chosen based on the maximum and minimum temperatures in Oslo in 2020.*

From table 2.1, it can be concluded (calculated from the mean of the values) that the average speed of sound is about 337 m/s.

2.1.5 Measuring sound

2.1.2 on page 7, described how the amplitude was the height of waves from their middle position. This height determines loudness measured in decibels (dB), reflecting the intensity in pressure waves. Decibel is a dimensionless unit, which expresses the logarithmic ratio between two physical quantities and quantifies the relative loudness [3]. The formula used for decibel is

$$dB = 10log_{10}(\frac{S_1}{S_2}), \tag{2.2}$$

where S_1 and S_2 are the intensity of two sounds. S_1 is the measured quantity, and S_2 the reference quantity. The reference quantity is usually an intensity of sound around the hearing threshold. Due to decibel being measured on a logarithmic scale, the intensity would double on a 3 dB increase [16].

Noise source	**Decibel level**	**Decibel effect**
Jet take-off (at 25 meters)	150	Eardrum rupture.
Thunderclap, chain saw	120	Painful. 32 times as loud as 70 dB.
Boeing 737 or DC-9 aircraft at one nautical mile before landing	90	4 times as loud as 70 dB. Likely damage in 8 hour exposure.
Garbage disposal	80	2 times as loud as 70 dB. Possible damage in 8 hour exposure.
Vacuum cleaner	70	Arbitrary base of comparison. Upper 70s are annoyingly loud to some people.
Conversation in restaurant	60	Half as loud as 70 dB. Fairly quiet.
Quiet suburb, conversation at home	50	1/4 as loud as 70 dB.
Library	40	1/8 as loud as 70 dB.
Quiet rural area	30	1/16 as loud as 70 dB. Very quiet.
Breathing	10	Barely audible.

Table 2.2: *An overview of different measurements of sounds and human perception of it [17].*

2.1.6 Limits for noise

The European Noise Directive (END) operates with two leading indicators, L_{den} and L_{AFmax} [18]. They use the indicators to determine noise levels. L_{den} is the average noise level for a day and stands for noise throughout the day, evening, and night. The moments are 07-19 as day, 19-23 as evening, and 23-07 as night. L_{AFmax} is the maximum noise level measured during a time period. From the following information, the following boundaries exist for the shooting range:

Noise source	**Noise level in outdoor facilities and outside the windows of rooms used to noise increasing activities**	**Noise level outside bedrooms in timeframe night 23-07**	**Noise level in outdoor facilities and outside the windows of rooms used to noise increasing activities. In timeframe day and night, 07-23.**
Shooting ranges	L_{den} 35 dB	No activity should be taking place	L_{AFmax} 65 dB

Table 2.3: *Overview of acceptable noise level from noise source [6].*

Furthermore, noisy training activity at PNB is only allowed on weekdays between 07:00 and 19:00.

2.2 Audio classification

2.2.1 Digital audio

Digital audio is incorporated in many different technologies and serves properties as computers, phones, and speakers. A digital audio interface incorporates analog-to-digital and digital-to-analog converters [19]. These converters allow recording analog audio data from a microphone to a digital data stream. The sound's pressure waves can be measured on a physical medium for later reproduction by regenerating the pressure waves. When the data is converted to digital data streams, it becomes possible to manipulate with computers.

The sample rate is an essential aspect of converting pressure waves to digital audio. Samples rate is the number of samples captured per second to represent a sound wave [20]. These sample rates are measured in Hz or cycles per second. In converting audio pressure wave to digital audio, the wave is converted into data through a series of samples. The sample is taken at a particular time and records the amplitude. Finally, the recorded amplitude is converted into binary data.

Figure 2.3 on the following page illustrate how the sample rate affects the digital result. When the sample rate is high, an analog wave's complexity grows, and the reconstruction possibilities increase. The typical sampling frequency is 44100 Hz and captures most perceivable human information in acoustic sound [20].

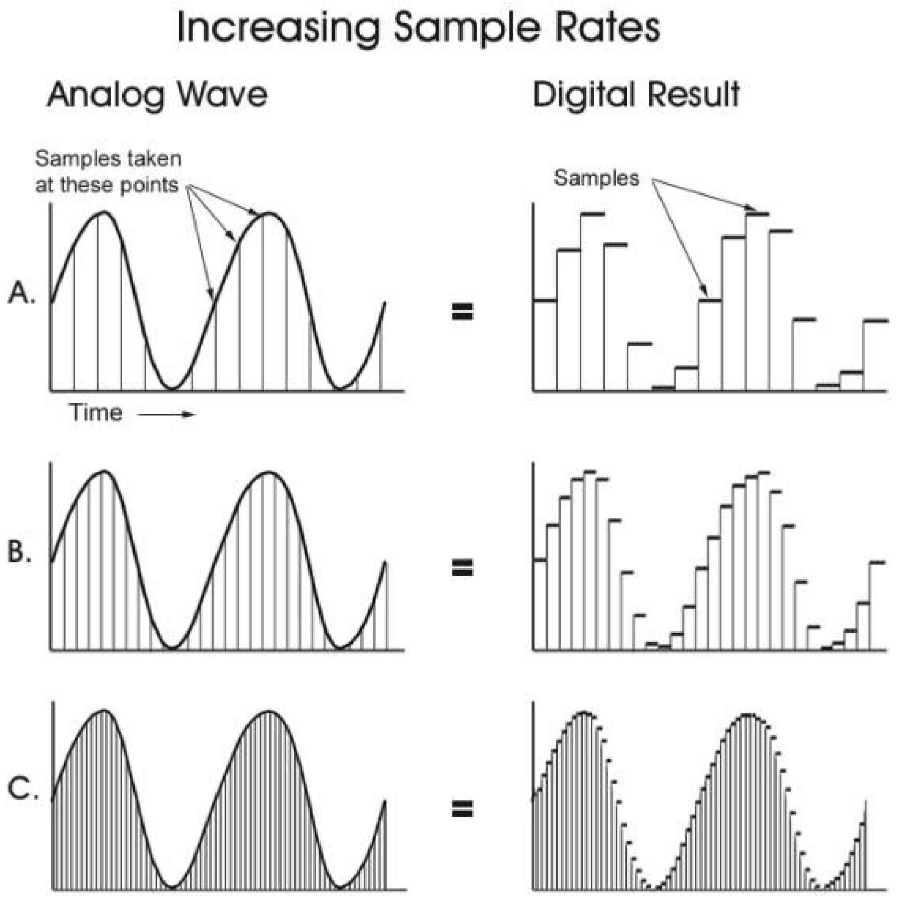


Figure 2.3: *Illustration of the digital result dependent on the samples rate [21].*

2.2.2 Spectrograms

A spectrogram is described as a detailed audio view, representing time, frequency, and amplitude all in one graph. Different sounds often have characteristics not just in time but also in frequency content [22]. Therefore, the spectrogram is commonly used when analyzing audio.

Spectrograms are two-dimensional graphs, with a third dimension represented by the color. The vertical axis represents frequency, the horizontal axis represents time, and the color represents a sound wave's amplitude [23].

One generates a spectrogram by dividing the time domain signal into shorter segments of equal length, then, lastly, applying Fast-Fourier-Transform (FFT). The spectrogram is, therefore, a plot of the spectrum in each segment. A ”frame count” parameter further defines the number of FFTs used to create a spectrogram. Consequently, it determines the amount of overall time signal split into independent FFTs [24].

2.2.3 Mel-spectrograms

The mel-spectrogram is a spectrogram with the mel-scale at its y-axis. This scale describes the perceptual distance between pitches of different frequencies [25]. The mel-scale is built in a manner where different 'sounds' sound alike when the distance is equal. Contrary to the Hz scale, where the difference in sound between 500 and 1000 Hz is significant, while between 7500 and 8000 is barely noticeable.

2.2.4 Delta-spectrograms

One obtains a delta spectrogram by taking the discrete cosine transform (DCT) of the frequency points where the distance is equal (mel-scale) [25]. The delta spectrogram has a delta feature, where delta order 1 is the first derivative, delta order 2 is the second derivative, etc.

Visually, the difference between the different spectrograms is illustrated in the figure below.

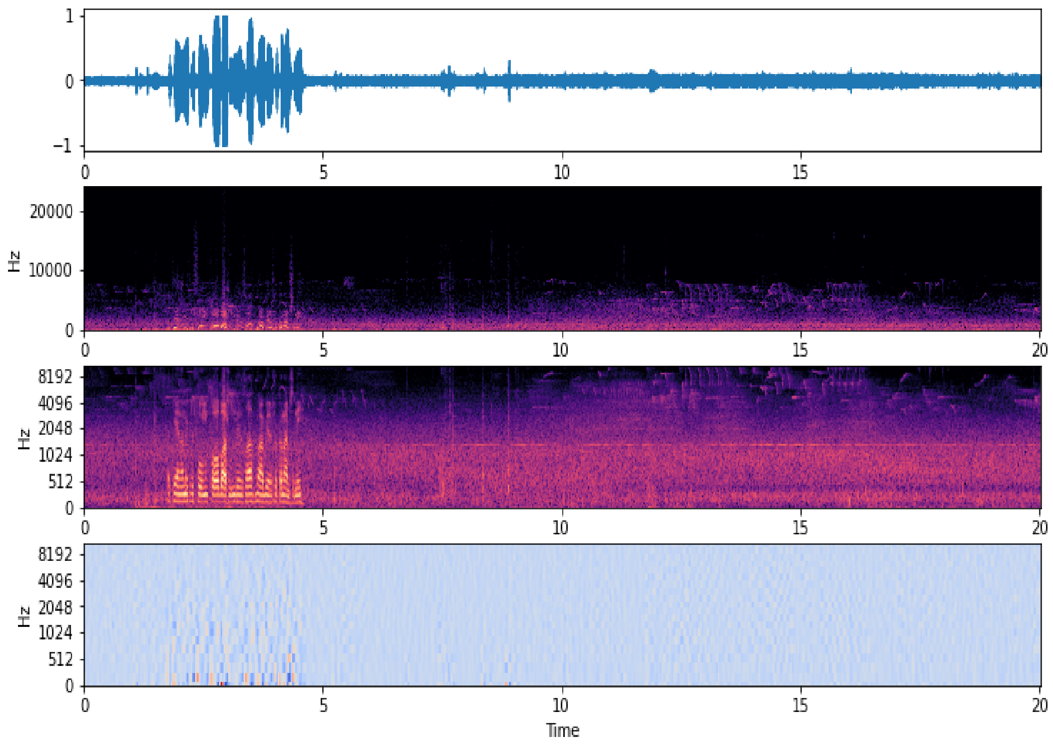


__Figure 2.4:__ The figure displays the conversion of audio pressure waves to different spectrograms. At the top are the audio pressure wave, linear spectrogram, normalized mel-spectrogram, and delta-spectrogram (order 1).

2.2.5 Analysis windows

When training and applying machine learning models with data, fixed size vectors are usually required. That differs from the recording of sound, as it forms a continuous stream of data. In order to account for this difference, the sound signals are often split up into analysis windows of fixed length [22]. Furthermore, the classification is done on the windows independently. There is no universal choice of window length; however, the window length is usually longer than the longest target sound.

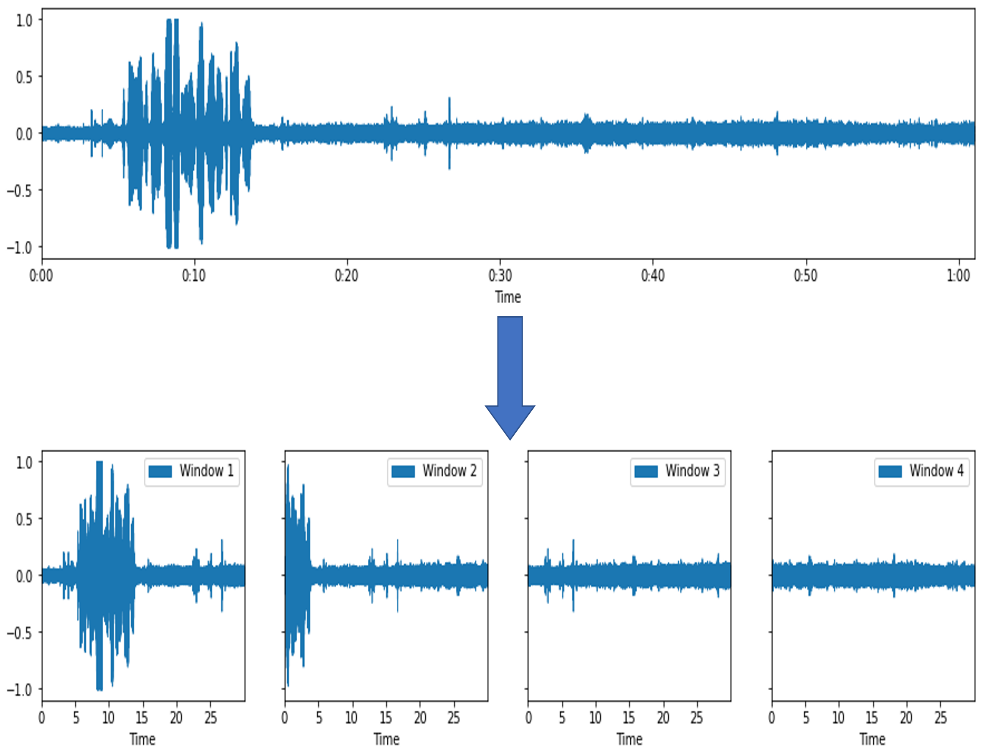


Figure 2.5: *Audio stream split into windows before subsequent analysis.*

In figure 2.5, an audio stream has been split up into four overlapping windows, covering 30 seconds each. Overlapping windows are a form of data augmentation. Data augmentation is described as a method where one can synthetically generate new labels from existing samples [22].

As the window lengths are chosen manually, they could be adjusted to increase data samples. Case in point, the overlapping windows could be set to 10 seconds with 5 seconds of overlap, creating eleven samples. Additionally, creating overlapping windows (data augmentation) has proven to increase classification accuracy [26]. With the overlapping windows, one can vary wherein the window an event occurs.

2.3 Combining signals

2.3.1 Time series

Data for time series are collected repeatedly at different times [27]. Due to the collection of data points at adjacent periods, there is potential for correlation between observations. Time series models can be used for predictions of future outcomes and understanding past recordings. Furthermore, a time series representation helps compare lagged signals reflecting shifts in time. There are numerous statistical and machine learning techniques handling the complexity of time series. The cross-correlation-based statistical methods and recurrent neural networks in deep learning are among the most popular choices.

2.3.2 Cross-correlation

Cross-correlation is a measure of similarity between two-time series or signals, where a potential dislocation of one signal relative to another is taken into consideration [28]. Conceptually, it creates or chooses a reference signal and determines to which degree the object under question resembles the reference signal [29]. Cross-correlation is a traditional concept in analyzing, signal processing, and image analysis. In signal processing, cross-correlation is used to measure the similarity between two time series as a function of time shift, where one series is relative to the other [30].

From this theory it is elaborated that the process has means $\mu_{x(t)}$, $\mu_{y(t)}$ and variances σ_x^2, σ_y^2 at time t, for every t. The definition of cross correlation between signals X and Y at times t_1 and t_2 is formulated as:

$$R_{XY}(t_1, t_2) = E[X_{t_1}, Y_{t_2}], \tag{2.3}$$

where E is the expected value. After calculating the cross-correlation between two signals, the maximum cross-correlation as a function of time delay would indicate the point in time where the signals are best-aligned [30]. Furthermore, the optimal time delay is calculated by the following equation:

$$\tau_{delay} = \underset{t \in R}{\operatorname{argmax}}(E[X_{t_1}, Y_{t_2}]), \tag{2.4}$$

where τ_{delay} is the optimal time delay between two signals at time t, determined by the $argmax$ function.

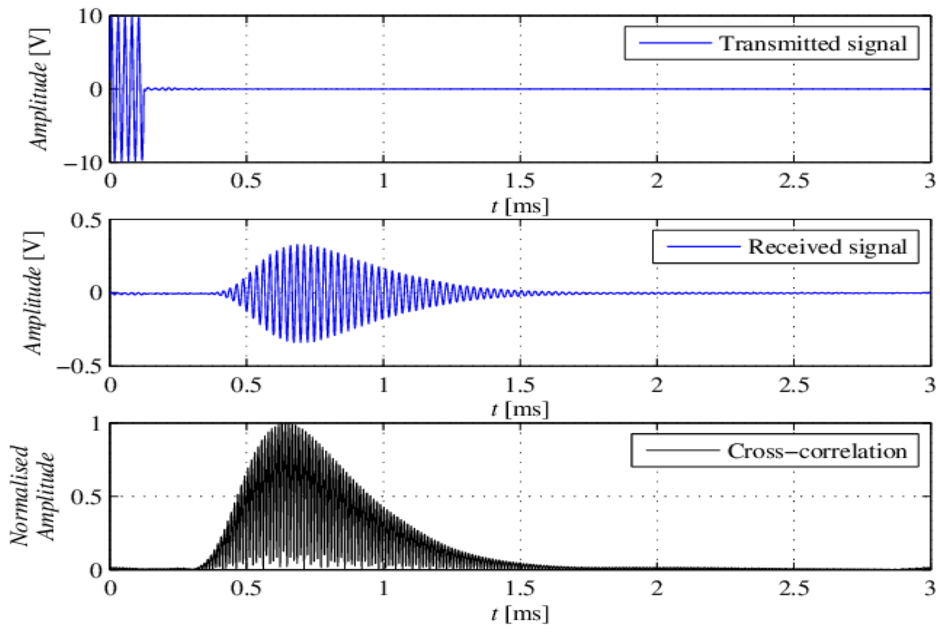


Figure 2.6: *In the figure above, the cross-correlation between two signals is depicted. The cross-correlation chooses a peak in correlation dependent on the time delay between the two-time series [31].*

2.3.3 Fourier transform of the cross-correlation

The Fourier transform is a mathematical tool that deconstructs signals or waves into its sinusoidal components [32]. Sinusoidal components have three characteristics: frequency, amplitude, and phase. Fourier transform gives the ability to decompose multiple signals into pure frequencies, making it a powerful tool within signal comparison. The following equation could describe Fourier transform:

$$\begin{aligned} \mathcal{F}_{\{g(t)\}} &= G(f) \\ G(f) &= \int_{\infty}^{-\infty} g(t)e^{-2\pi i f t}\, dt \end{aligned} \tag{2.5}$$

F is in this equation the frequency, $G(t)$ (the spectrum of $g(t)$) tells how much power $g(t)$ contains at the given frequency. One obtains g with the inverse of G:

$$\begin{aligned}\mathcal{F}^{-1}\{g(t)\} &= \int_{\infty}^{-\infty} G(f)e^{2\pi ift}\,df \\ \mathcal{F}^{-1}\{g(t)\} &= g(t)\end{aligned} \tag{2.6}$$

From 2.6 it is observed that one obtains the original function $g(t)$ from the function $G(f)$ via an inverse Fourier transform [32]. Fourier transform is something that is best explained visually due to its complex being. Below is a graphic that displays how the transform decomposes a sum of air pressure into pure frequencies:

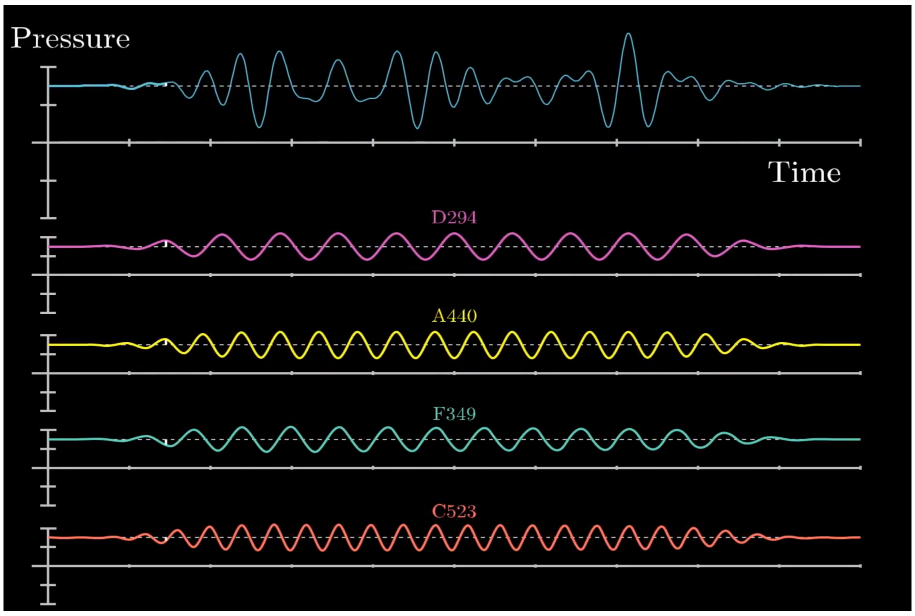


Figure 2.7: *The original air pressure is displayed at the top. Below is the four different frequencies that creates the sum. Fourier transform decomposes the sum down to the unique frequencies that makes it possible to "separate the ingredients from the soup" [33].*

The use of Fourier transform within cross-correlation is understood once the application and results of a transformation are obtained. The Fourier transforms decomposition makes it possible to filter out the parts created by 'noise'.

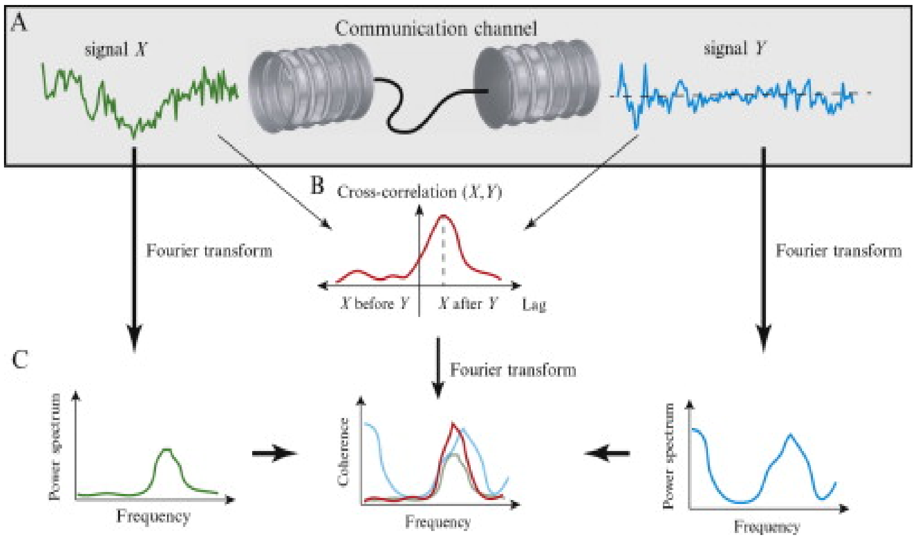


Figure 2.8: *Graphics which displays how Fourier transform changes the signals in question [34].*

Figure 2.8 displays two signals prior to Fourier transformation and calculation of the cross-correlation in the frequency domain. The signals are initially contaminated by background noise. After Fourier transformation and noise filtering in the frequency domain, the cross-correlation of the filtered signals can be calculated.

2.4 Machine learning

In a time where massive amounts of data are easily available, learning algorithms from the field of machine learning can turn data into knowledge. Machine learning has evolved as a subfield of artificial intelligence and involves self-learning algorithms that derive knowledge from unlimited data to make predictions. Within the field of machine learning, there are three distinct types: *supervised learning*, *unsupervised learning*, and *reinforcement learning*.

In supervised learning, a model learns from labeled training data as examples of the input-output relationship to be learned. The term ”supervised” refers to the output values, usually provided by manual annotations by humans inspecting the data. Subsequently, the correct output values are known prior to the training process in supervised learning. *Unsupervised learning* deals with unlabeled and often unstructured data. The purpose of unsupervised learning is to extract meaningful information without the guidance of a known outcome variable or reward function [35]. Lastly, *reinforcement learning* builds on training an agent, where one defines a measure of reward for particular actions carried out by the said agent. In this thesis, we will focus on supervised learning problems.

2.4.1 Random forest classification

Random forest is an ensemble based learning method built from multiple decision tree classifiers. The method is known for its good scaleability and ease of use. Conceptually, the trees are made by iteratively splitting its nodes to obtain the largest information gain. The root node of the three represents a subset of data, while on the contrary branches represent a smaller subset of data from the previous node [36]. Ultimately, the leaves on the tree are the value dedicated to the corresponding branch as illustrated below:

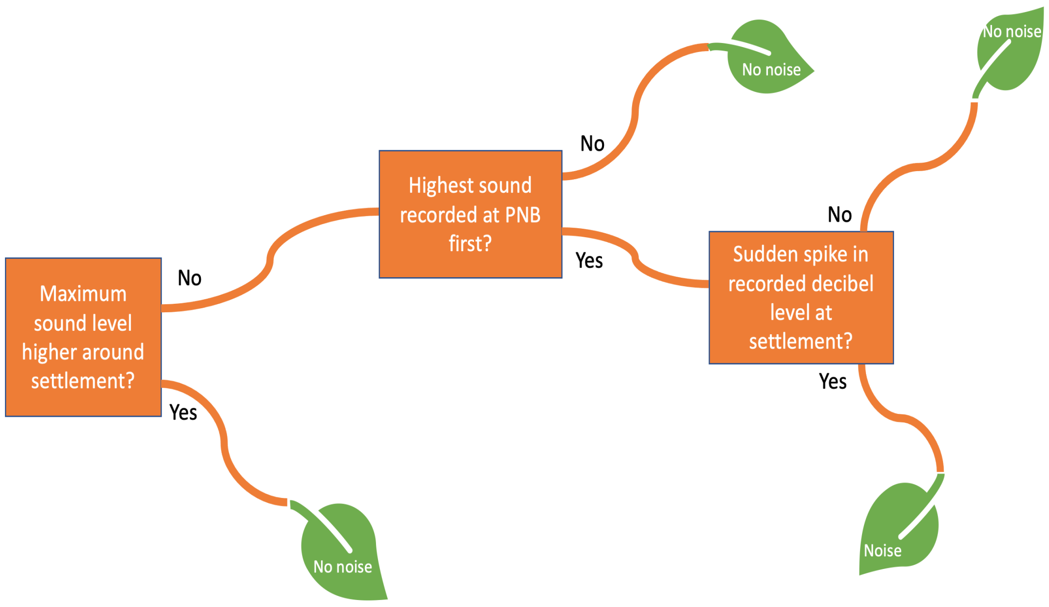

Maximum sound level higher around settlement?
No
Yes
No noise
Highest sound recorded at PNB first?
No
Yes
No noise
Sudden spike in recorded decibel level at settlement?
No
Yes
No noise
Noise

Figure 2.9: *The decisions and outputs of a decision tree. Based on the features in a dataset, a decision tree model learns a series of questions to infer the class labels of examples [37]. In the following figure cut-off values is defined and binary questions is asked. Prediction of noise in the leaf is therefore based upon the answer to the binary question. The decision tree is applied to data from audio recorders, where the explanatory value are sound levels. Furthermore, the response value is whether or not noise was caused by PNB. Inspired by [36].*

For the sake of reducing combinatorial search space the random forest implements binary decision trees. Each parent node are split into two child nodes (d_{left} and d_{right}) based upon the largest information gain (IG) as indicated by the following equation [38]:

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right}) \qquad (2.7)$$

Here, f is the feature to perform the split; D_p is the parent node dataset; I denotes the choice of *impurity* measure; N_p is the number of training examples at the parent node: N_{left} and N_{right} is the number of examples in respectively D_{left} and D_{right}.

There are three impurity measures or splitting criteria commonly used in binary decision trees: Gini impurity (I_G), Entropy (I_H), and the classification error (I_e).

For the sake of simplicity, only Gini impurity is elaborated (for the I-function in 2.7) in this section. The Gini impurity is used as a criterion for minimizing the probability of misclassification [38]:

$$I_G(t) = \sum_{i=1}^{c} p(i|t)(1 - p(i|t)) = 1 - \sum_{i=1}^{c} p(i|t)^2 \tag{2.8}$$

Here, $p(i|t)$ is the proportion of samples that belong to class i for some particular node (t). The impurity is 0 (minimum) if all examples at a node belong to the same class and 0.5 (maximum) when classes are uniformly distributed.

Additionally, a decision tree has a maximum depth that indicates the number of nodes made from the root node to the leaf node, to be tuned for a good bias-variance trade-off. The tree will generate leaves until it reaches maximum depth. However, one has to be careful, as the deeper a decision tree gets, the more complex the decision boundary and the risk of overfitting increases. Random forest is based upon taking the average of multiple decision trees, which may individually suffer from high variance, to achieve a more robust model that generalizes well. The training algorithm can be summarized in the following four steps [39]:

Random forest algorithm

1. Draw a random bootstrap sample of size n (randomly choose n examples from the training dataset with replacement)
2. Grow a decision tree from the bootstrap sample. At each node:
   (a) Randomly select d features without replacement
   (b) Split the nodes using the features that provided the best split according to the information gain.
3. Repeat the steps 1-2 k times
4. Aggregate the prediction by each three to assign class label by majority voting.

The majority voting in step 4 chooses the class label, which occurred most throughout the individual decision trees' predictions. It could be further understood in the visualization in figure 2.10.

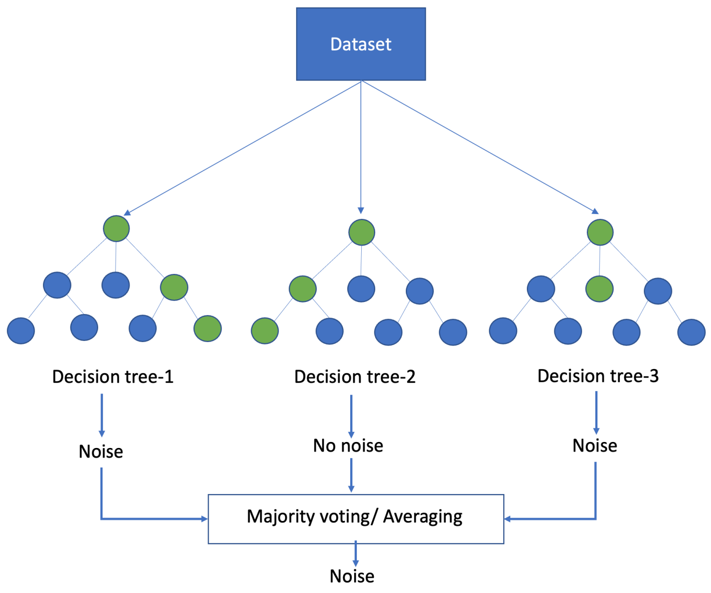


Figure 2.10: *Visualization that explains the last step in the random forest algorithm. Each decision tree predicts a class, and the most frequently occurring classification is chosen as the final prediction*

2.4.2 Classification problems and performance metrics

Classification problems are a subcategory of *supervised learning problems*. The goal is to build a classification model to predict the categorical class label based on some measured feature characteristics [35]. For instance, classifying whether an e-mail is spam or not, determine from an image which number is written, or deciding from sound recordings if noise is present or not.

The previous example of classifying whether an e-mail is spam represents a typical example of a binary (2-class) classification problem. Classification problems with more than two classes (such as the classification of digital images of handwritten digits between 0 and 9) are referred to as multi-class classification problems.

Accuracy

Multiple metrics can quantify the performance of a classification model. Most commonly is *accuracy* - the ratio of correct predictions to total predictions.

Recall

Other metrics may be useful when dealing with unbalanced data in binary classification problems, such as *recall* (sensitivity) - the number of correct positive predictions divided by the total number of positive samples [22]. The recall is acquired with the formula [40]:

$$Recall = \frac{TP}{TP + FN}, \tag{2.9}$$

where TP is the number of true positives (the samples correctly predicted as positives) and FN is the number of false negatives (the samples predicted as negatives by mistake).

Precision

Precision is the number of of true positives divided by the total number of positive predictions defined by the formula [41]:

$$Precision = \frac{TP}{TP + FP}. \tag{2.10}$$

Here, FP denotes the false positive (the samples predicted as positives by mistake).

F1

The F1 score is defined as the combination of the recall- and precision scores obtained by taking the harmonic mean of the scores [42]:

$$F1 = \frac{2RP}{R + P}, \tag{2.11}$$

where R denotes recall and P denotes precision as defined above.

Average precision

One uses thresholds to convert predictions into a class label. When the prediction is equal to or greater than the thresholds, the prediction is classified as one class. Otherwise, it is classified as the other class.

However, the average precision score separates itself from the different metrics in regards to having no threshold. One calculates the average precision by using a range of thresholds (from 0 to 1) and calculating the weighted sum of the scores at every threshold. The sum is divided by the number of thresholds used to acquire the average precision score.

2.4.3 The bias-variance trade-off

Models with large bias typically oversimplify the input-output relationship by putting too little emphasis on the training. Subsequently, the model underfits the training data. In comparison, variance is the variability in the model prediction for a given data point or a value that explains the data's spread. Models with high variance tend to overfit the training data and fail to generalize on unseen data [43]. Models achieving bias-variance trade-offs learn from the training data without overfitting nor underfitting, making it desirable to realize.

Mathematically, bias and variance can best be explained by expressing the expected prediction of a regression model. The expected prediction error of a model $\hat{f}(X)$ with given explanatory variables $X = x_0$ and actual values Y, where $Y = f(x) + \epsilon$ and $E(\epsilon) = 0$ has squared error losses as [36]:

$$Err(x_0) = IrreducibleError + Bias^2 + Variance \tag{2.12}$$

Where $Err(x_0)$ is the sum of $Bias^2$, $Variance$ and the $IrreducibleError$. The $Irreducible$ term is the variance of the response value around the true mean and can be explained as the noise in the data [43]. $Bias^2$ is the squared bias and the amount by which the average of the prediction differs from the actual mean. Lastly, $variance$ is the expected squared deviation of $\hat{f}(X)$ around its mean.

Based on this the bias-variance trade-off could be illustrated:

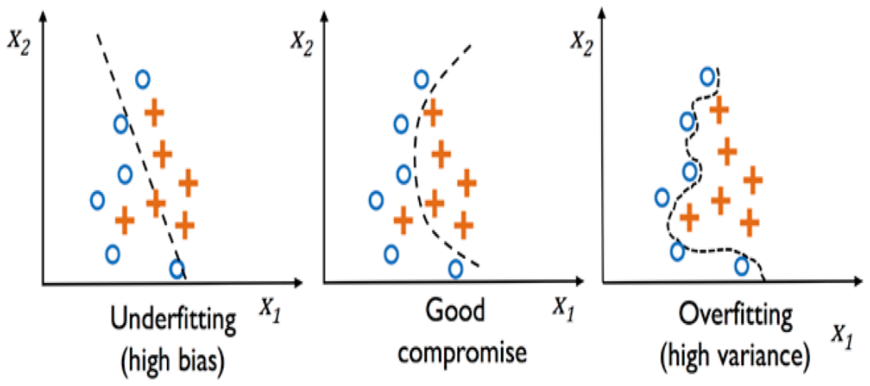


Figure 2.11: *Illustration of the bias-variance trade-off [44]. In this example the y-axis and the x-axis is the explanatory variables. First plot illustrates high bias which leads to underfitting. The plot in the middle illustrates a good trade-off which gives the best results. The last plot shows a model with high variance (memorises the data) which leads to overfitting.*

In conclusion, a model with high bias will have poor performance on the response value due to the lack of complexity in capturing relationships in the training data. Consequently, the model will underfit and have poor performance on both training and test data. A model that suffers from overfitting (high variance) will memorize the data and have good results on training data. However, when encountering test data (unseen data), it will fail to generalize and achieve poor results. Therefore, machine learning models' end goal is to achieve a bias-variance trade-off as illustrated in the middle of figure 2.11. The model will achieve good results on both the train and test data when achieving the bias-variance trade-off.

In practice, results from a machine learning model tend to have the issue of overfitting. Issues of this sort could be resolved by introducing regularization or reducing the model's complexity. Regularization works by adding penalties when a model's complexity increases. This regularization parameter penalizes all the parameters except the intercept, expecting the model to generalize the data and not overfit [45].

2.4.4 Grid search and model-validation

In machine learning, there are two types of parameters: those that are learned from the training data (model parameters) and the parameters of a learning algorithm that are optimized separately (hyperparameters) [37]. The weights of the random forest model are one type of model parameter. Contrarily, three depth is a hyperparameter.

Furthermore, the model parameters are properties learned by the training data, whereas hyperparameters are manually chosen [46]. Grid search, a popular hyperparameter optimization technique, can help to improve a model's performance by finding good combinations of hyperparameter values. Grid search is based on a "brute-force" approach where one specifies lists of values for the different hyperparameters. For all the hyperparameter value combinations, the learning algorithm trains and evaluates the separate resulting models to identify the optimal combination of hyperparameter settings [37].

Before training the models, splitting of the dataset to obtain training- and validation samples is required. Such splits should be done in a random manner, usually by including more samples in the training set than in the validation set. Each model is trained on training data and its predictive performance is evaluated on the validation data as a basis for the model selection.

K-fold cross validation

With grid search, one may obtain the optimal combinations of parameters for the training set. It was mentioned in 2.4.3 on page 26, machine learning models' end goal is to achieve a bias-variance trade-off. The cross-validation is complementary to the grid search and presents the model to unseen data, which gives a better evaluation of the model's performance.

With K-fold cross-validation, the training data is split into k equally sized folds. Out of these k folds, one fold is used as a validation set, and $k - 1$ folds are used in the training process. The folds that are used for validation is then cycled until it has been trained and validated k times - each time with a unique training and validation set.

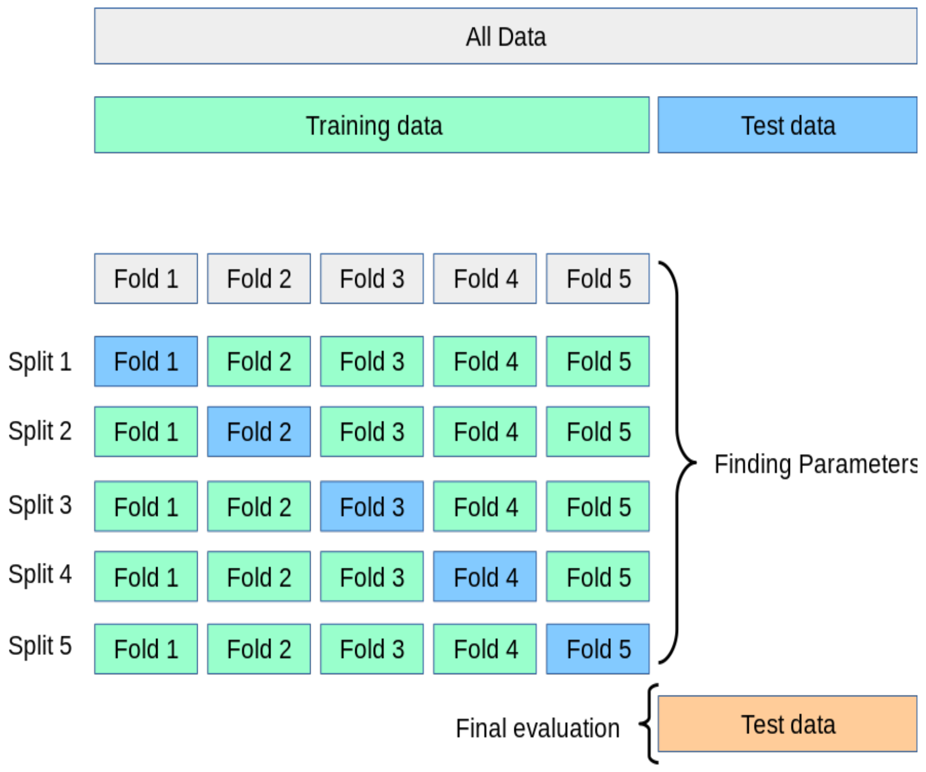


Figure 2.12: *Graphics that shows how the cross-validation splits the training data. For each split, there is a unique set of training and validation set [47].*

Figure 2.12 shows how the k-fold cross-validation splits the data into folds and then chooses a unique set of validation data. Furthermore, it fits the model on the training data and evaluates the validation data. The score from this evaluation is kept, the model is discarded, and a new process occurs. This continues until the model has been trained k-times. Lastly, the average of scores is calculated to give a good estimate of the model's performance. It is further explained in the graphic below:

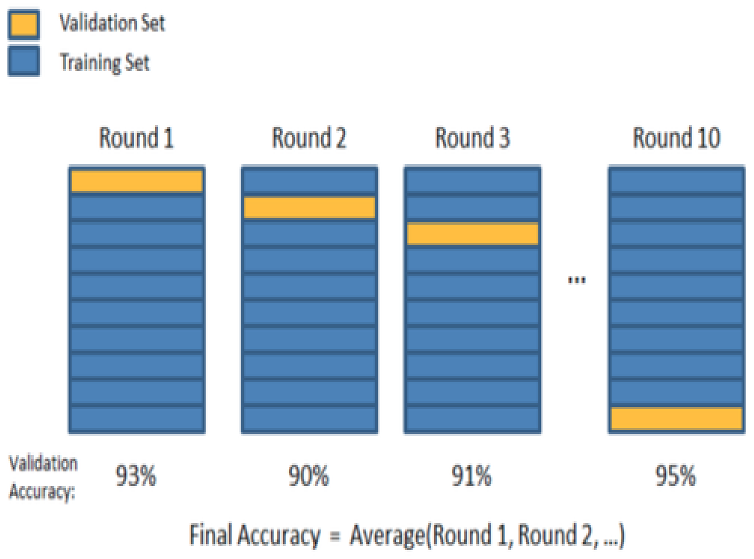


Figure 2.13: *From the figure, it is observed how the k-fold cross-validation calculates a validation accuracy in each split. When it has completed all the rounds, the average score is calculated as a final accuracy [47].*

2.5 Neural networks

Neural networks (NN) modelling represent a subfield of machine learning for flexible learning in *layers* of increasingly meaningful representations [48]. Neural Networks was first introduced in the 1950s by Rosenblatt's perceptron model. What started as a significant interest slowly faded, as no good solutions for training neural networks with multiple layers was found. However, thanks to breakthroughs in the previous decade, NN is more popular today than ever. This subsection will explain the neural network building blocks and the various components implemented in later model building.

2.5.1 Multi-layer neural network

To understand multi-layer neural networks (MLP), one must first understand the single ”neuron” model:

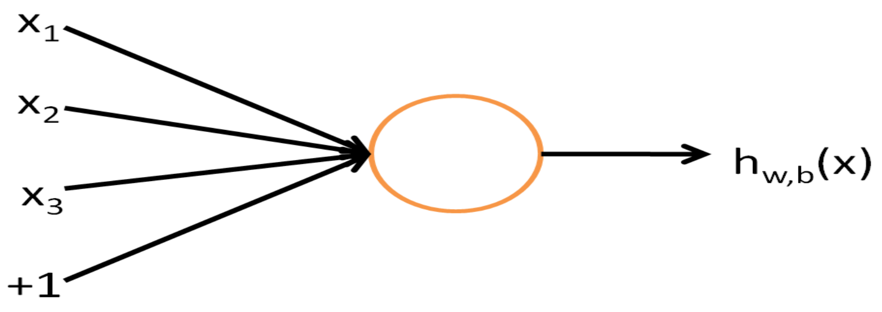


Figure 2.14: *Illustration of a single layer neural network [49].*

Consider a case in supervised learning where one has labeled training examples (x^i, y^i). This simple neural network is a computational unit that takes input X_1, X_2, X_3 (and an +1 intercept term) from the training examples and weights W_i. Aditionally, it outputs $H_{w,b}(X) = f(W^T x) = f(\sum_{i=1}^{3} W_i x_i + b)$, where $f : R \longrightarrow R$ is called the *activation function* [49].

Moreover, a neural network is put together by hooking together multiple single neurons as shown in 2.14. This could be observed in figure 2.15:

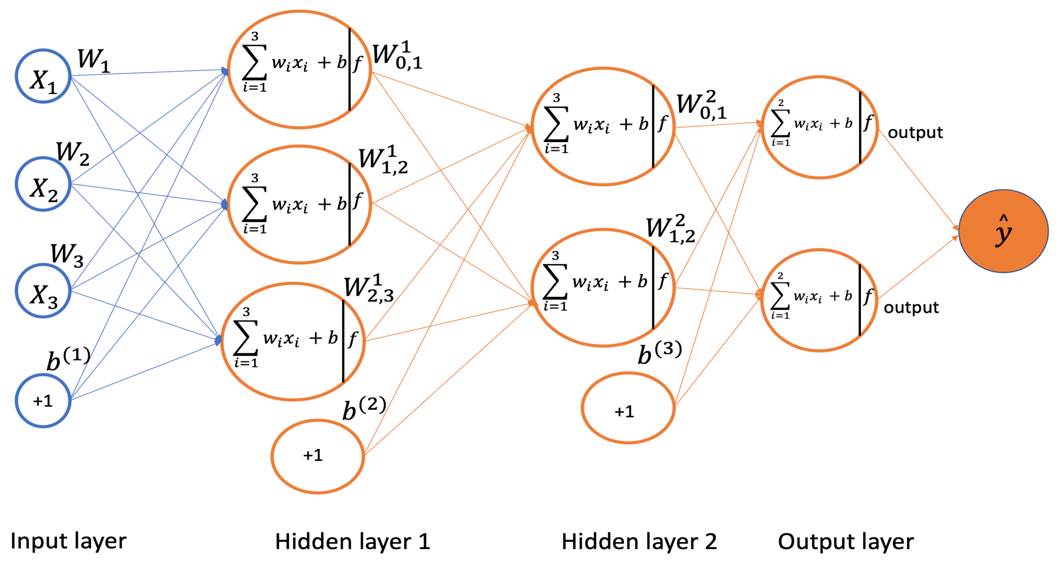


Figure 2.15: *Illustration of a multi-layer neural network [49] with two hidden layers.*

.

The input layer is from the training examples with a bias unit (+1). In this case, the input is vectors, but the network possesses the ability to handle multidimensional data (such as images) by formally representing them as vectors. Layers 2 and 3 are called *hidden layers*.

Each neuron in a hidden layer computes a weighted sum of its inputs, offset by a bias before transformation by a non-linear activation function. The hidden layers contribute to the network's capacity for learning complex non-linear functions. The downside to this is that the model becomes more complex and thus more prone to overfitting.

The final layer is called the output layer and calculates the output of the network. Usually, when training and applying a neural network for predictions, data is fed as input to the first (input) layer. The network then executes its computations in the subsequent hidden layers before the output layer calculates the network output (a prediction). This characteristic *forward propagation* of information is emphasized by referring to these models as *feedforward neural networks*.

2.5.2 Activation functions

It was mentioned earlier in section 2.5.1 that activation functions are used in neural networks to non-linearly transforms the weighted sum (including the bias) in a neuron. We will primarily focus on two choices of activation functions: sigmoidal function and rectified linear unit (ReLU). Firstly presented is the sigmoidal function.

Sigmoidal activation function

The sigmoidal activation function is primarily known for its characterized S-shaped curve. It is most commonly found in machine learning in the logistic function [50]. The function is given by:

$$\theta_{Sigmoid}(x) = \frac{1}{1+e^{-x}} \tag{2.13}$$

The output of this function is a number in a small range between 0 and 1. Therefore, the sigmoid function can convert a real value into one interpreted as a probability. A plot of the sigmoid function and its derivative is observed in figure 2.16:

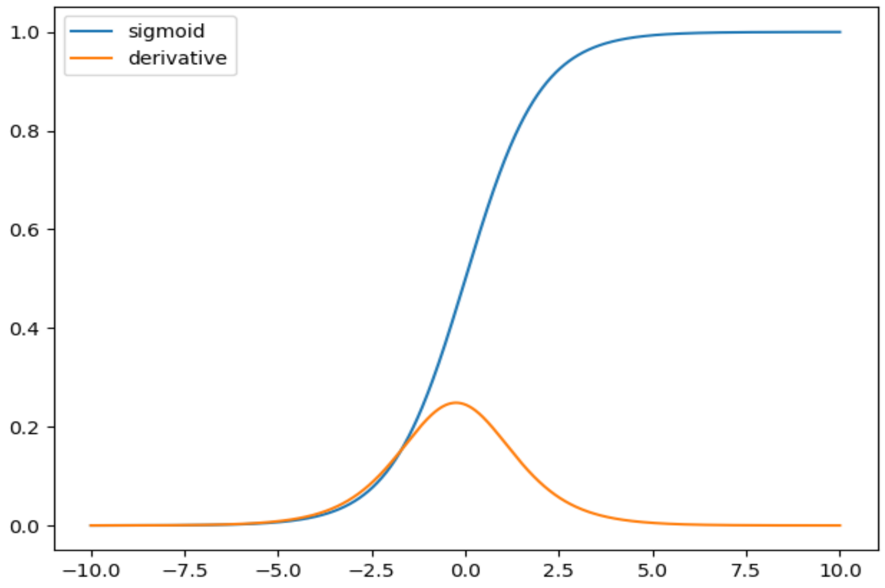


Figure 2.16: *Plot of the sigmoidal activation function and its derivative [51].*

One issue with the derivative of the sigmoidal function is that the derivative will be close to zero if the input is sufficiently large. There is a use of the derivative in neural networks when using backpropagation. Backpropagation was briefly mentioned in chapter 2.5.1, and is in essence a computationally efficient approach to compute the partial derivative of complex cost functions in multi-layer neural networks [52].

In this efficient approach, a first-order optimization method is used to minimize the loss. Therefore, the network weights update is proportional to the partial derivative of the loss function concerning the weights. As the gradient's magnitude is proportional to the function's derivative, the weight will be negligible when the derivative is small. This presents the vanishing gradient problem [53].

Rectified linear unit

The rectified linear unit (ReLU) is defined as:

$$\theta(x) = max(0, x) \tag{2.14}$$

The introduction of the ReLU activation function is mainly known for solving the computational problem of vanishing gradients in neural network training as the derivative of $\theta(x)$ is 0 for all negative inputs and 1 for all positive inputs, see figure 2.17:

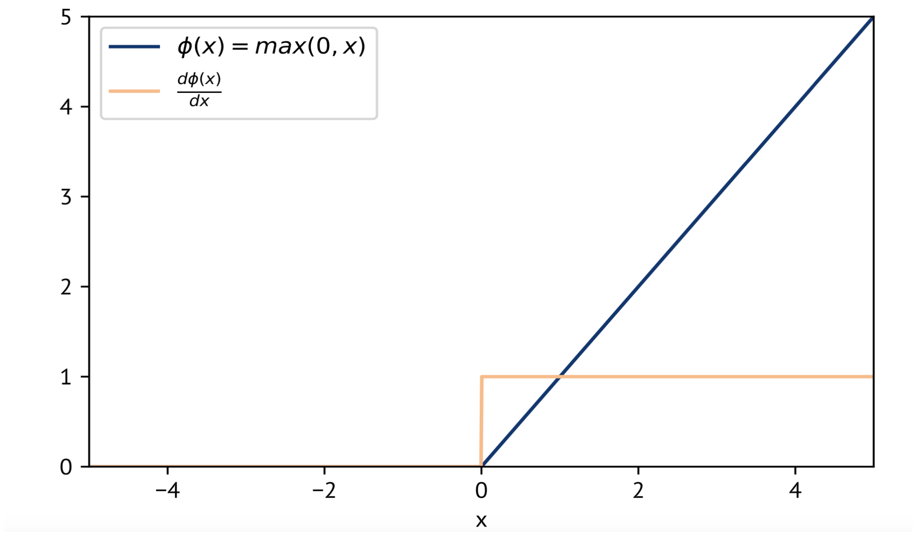


Figure 2.17: *Plot of the ReLU activation function and its derivative [53].*

2.5.3 Loss functions

In most networks, a distance score is calculated through the following equation:

$$J(w) = p - \hat{p} \tag{2.15}$$

Here, p is the actual class label, $\hat{p}$ is the predicted label, and $J(W)$ is the distance score. Furthermore, the function that calculates this distance is known as the loss function. The distance score gives the network an impression of how well it performed [54]. Different loss functions will return different errors, making loss functions critical regarding the learning for a model [55]. There are different loss functions for unique types of tasks. For example, Binary cross-entropy for binary classification, categorical cross-entropy for multi-classification, and mean squared error for regression problems [56]. As this thesis deals primarily with a binary classification problem, only binary cross-entropy is of relevance.

Binary Cross-Entropy / Log Loss

The loss function could further be elaborated [57]:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{n} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i)) \tag{2.16}$$

Where y is the actual label and p(y) is the predicted probability that the sample belongs to class 1 (binary). When examining the equation closer, it is observed that when the actual label is 1, the $log(p(y))$ (log probability of the label being 1) is added to the loss. On the contrary, it adds $log(1 - p(y))$ (log probability of the label being 0) to the loss when the actual label is 0, see figure 2.18:

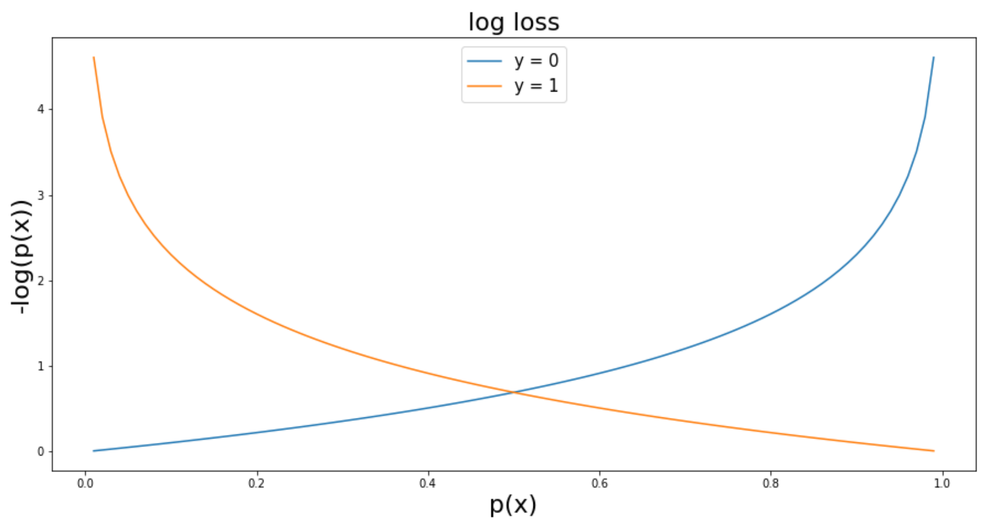


Figure 2.18: *Plot of the binary cross-entropy loss function and how it behaves. In this figure, the behavior when the actual label is 1 and 0 is illustrated, respectively.*

Figure 2.18 details how the worst predictions get penalized hardest. The further the prediction is from the actual target, the higher the loss. Vice-versa, the closer the prediction is to the actual label, the lower the loss.

2.5.4 Optimization

Training of non-trivial NN models is known to be a non-convex optimization problem. Non-convex optimization regards functions that typically have multiple local optima where a global optimum on the loss surface is hard to find [58]. In the descriptions below, we will focus on the *Gradient Descent* method.

Gradient Descent

Gradient descent is one of the most popular optimization methods. One can describe the main idea behind gradient descent as climbing down a hill until a local or global minimum is reached. This is seen in figure 2.19 [59]:

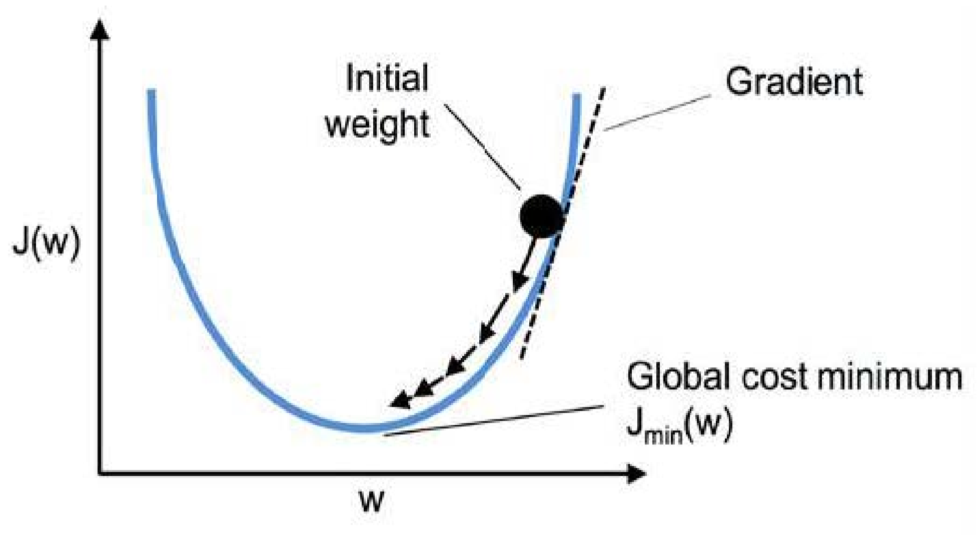


Figure 2.19: *In each iteration, a step in the opposite direction of the gradient is taken. The step size is further determined by the learning rate and the slope of the gradient.*

Mathematically, by iteratively computing the gradient of $J(w)$ with respect to the model parameters w, the loss affiliated with each network parameter can be calculated. Each iteration updates the parameters in the opposite direction of the gradient of $J(W)$ with respect to the parameters where the gradient gives the direction of the steepest ascent [60]. With the assumption that J is differentiable with respect to w, gradient descents update rule states [61]:

$$w \longleftarrow w - \alpha \Delta_w J(w), \tag{2.17}$$

where α is the learning rate. The step size in each iteration to reach the local minimum is determined by the learning rate. The direction of the slope is followed until a local minimum is reached, per figure 2.19. It is worth mentioning that algorithms like *Stochastic gradient* and *Adam* also use the concept of utilizing the gradient for updating parameters.

2.5.5 Training a neural network

In summary, based on the above theoretical descriptions, the training process of a neural network with a dataset $D = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$ consisting of N input vectors (x) with known target outputs (y) goes as follows:

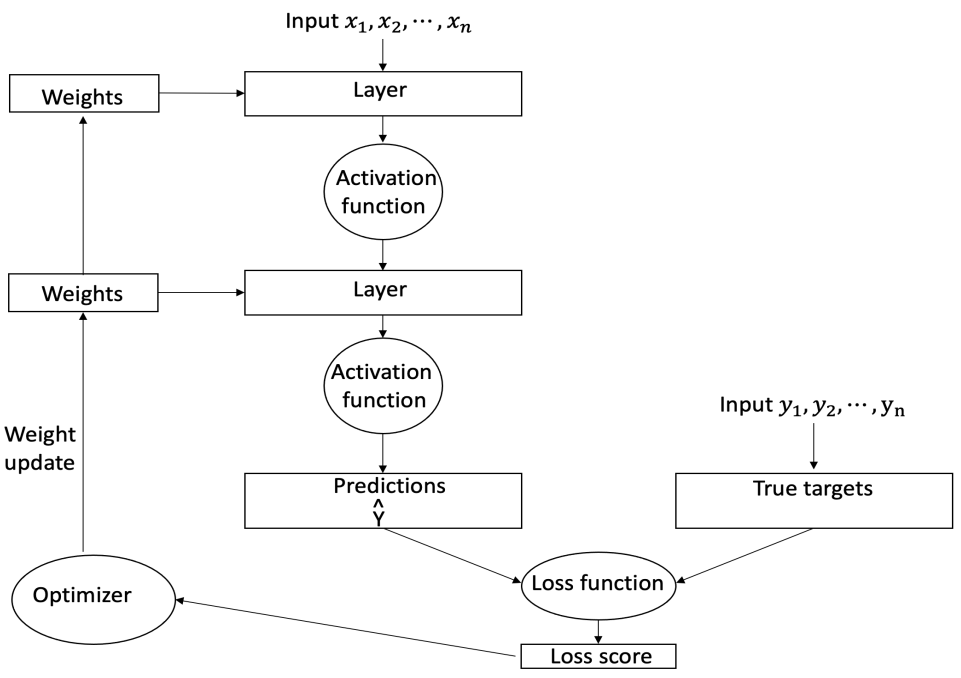


Figure 2.20: *Relationship between layers, activation function, loss function, and optimizer. Inspired by [54].*

The inputs pass on to a layer, where an activation function is applied. After passing through each hidden layer, the activation functions provide a non-linear transformation. Finally, a loss score (depending on the choice of loss function) is calculated. The acquired loss score is passed forward to an optimizer to adjust the network parameters before the next training epoch. With every example (new input) the model processes, the weights are adjusted in the correct direction [62]. This is the training loop. With enough repetitions, the network should yield weight values that minimize the loss function.

2.5.6 Convolutional neural networks (CNN)

Convolutional layers

Network models with convolutional layers are an alternative to the fully connected layers in neural network modeling. Convolution layers can handle (multidimensional) arrays such as images as input data. Images are comprised of their height times width represented as matrices. The inclusion of colors, formally called ’channels’, expands the image to 3-dimensional arrays. Grayscale and audio spectrograms are represented by only one channel, while RGB images have three channels.

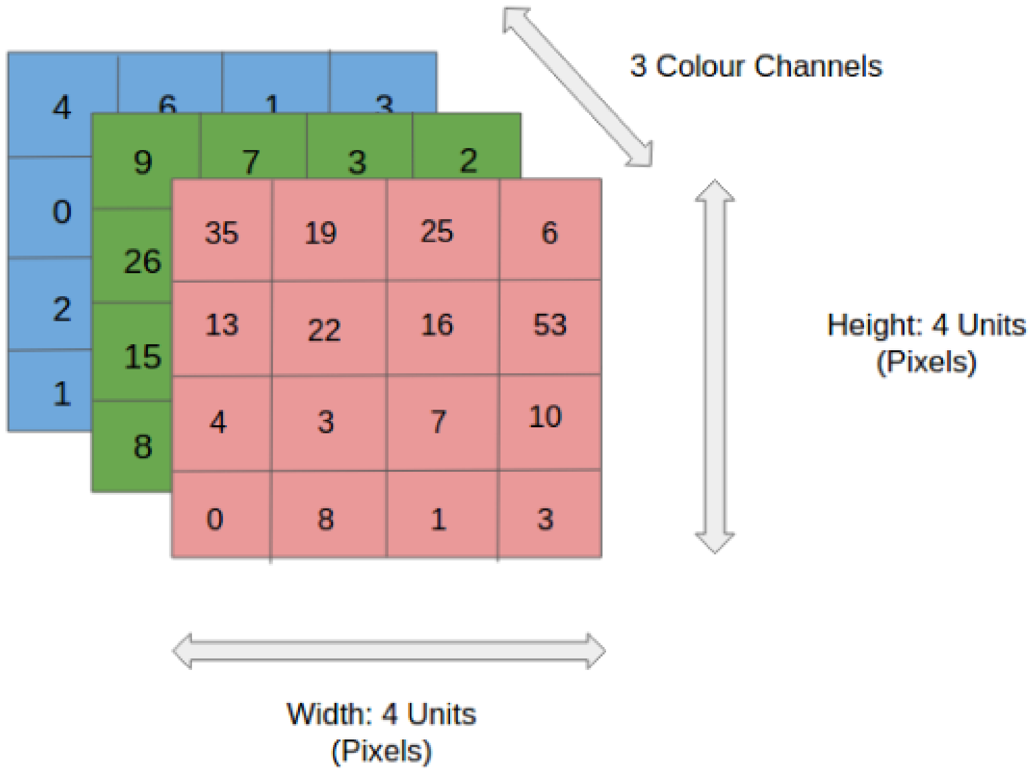


Figure 2.21: The dimension of an image [63]. This image have 3 channels (RGB).

The figure above shows a $4 \times 4 \times 3$ image, where the color information is represented in the channels: red, green, and blue. Convolutional layers are designed to process input of this type by performing discrete convolutions. Convolution kernels carry out the operations in the first part of a convolutional layer [63].

By convolving the following kernel of size 2×2

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

in strides of 2 with the first layer of the above $4 \times 4 \times 3$ image we obtain:

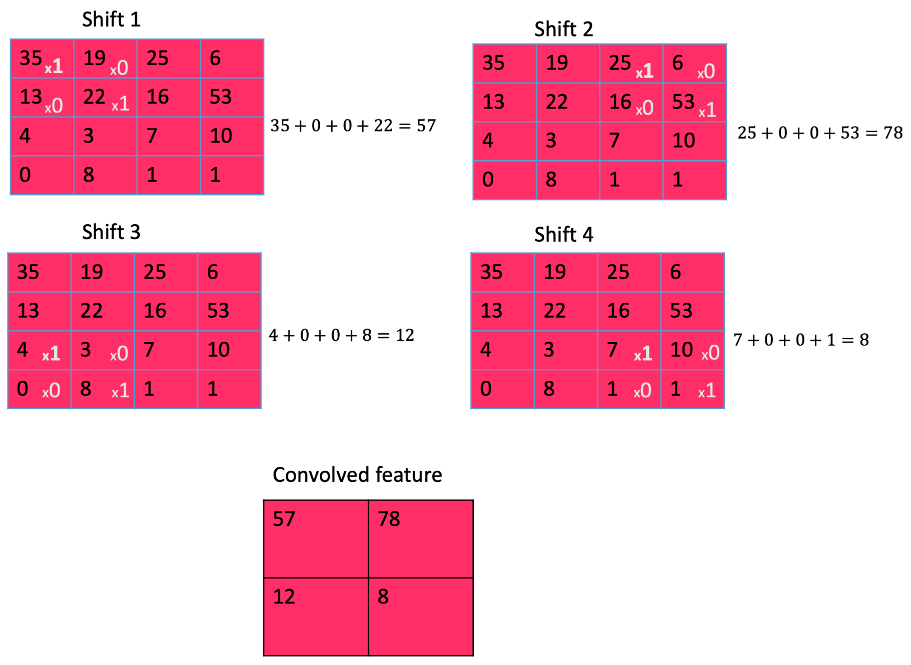


Figure 2.22: *The shifting process for the red image with a* 2×2 *kernel with strides* $= 2$.

The sum of each matrix multiplication yields the convolved features, and convolutions with the other image layers are calculated similarly.

Generally, the output size depends on the choice of padding strategy and stride length [64]. There exist three padding strategies: full, same, and valid. Full padding increases the dimension of the output and is therefore rarely used in CNN architectures. The same padding ensures that the output vector has the same size as the input vector (given a stride length of 1). Valid padding refers to the case where no padding is done Below is an example of the three different padding strategies for a 5 x 5 input with a kernel size of 3 x 3 and a stride of 1.

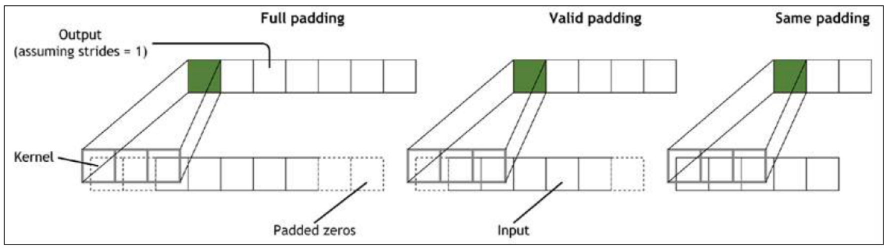


Figure 2.23: *The three different padding modes for a 5 x 5 input with a kernel size of 3 x 3 [64].*

Subsampling layers - pooling

Subsampling is an essential operation that is often used in CNNs. The subsampling can be applied in two forms of pooling: max pooling and mean pooling. The pooling layer is denoted by $P_{n_1 x_{n2}}$, which determines the pooling size [64].

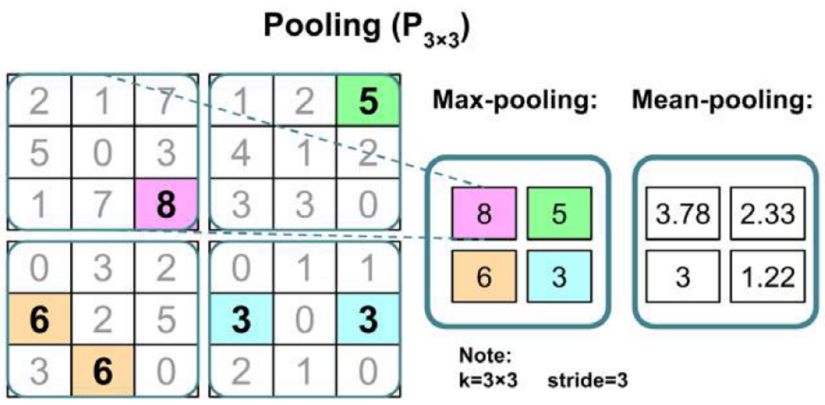


Figure 2.24: *How max and mean-pooling works. [63].*

The figure above displays how the pooling process is done. It determines a pooling size (3x3) and chooses the max value/mean value from each pool. Max pooling has an advantage over mean-pooling in terms of not being affected by small changes in the pool. Pooling gives the advantage of decreasing the size of the features. This reduces computational time and may reduce the degree of overfitting.

Convolutional network

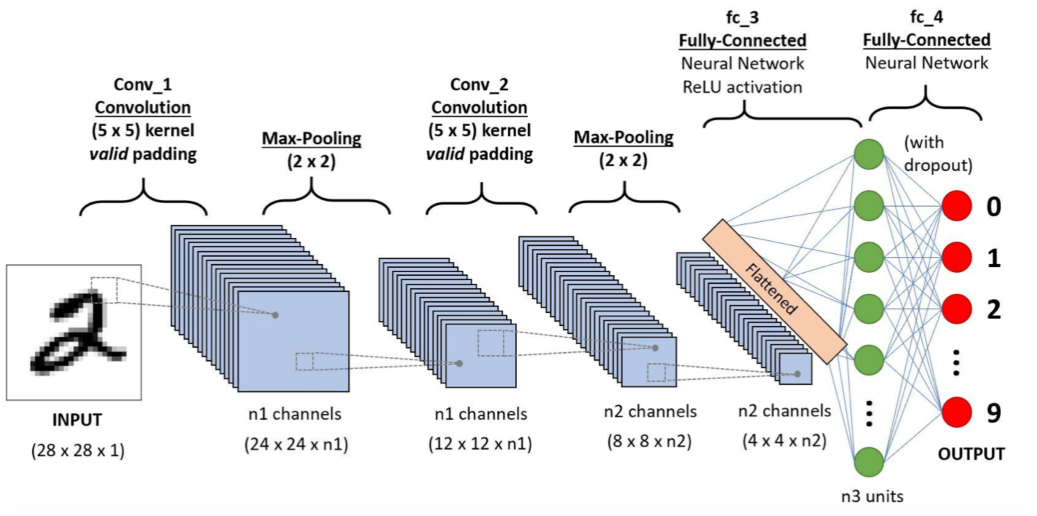


Figure 2.25: *A convolution neural networks architecture [63].*

The convolutional network is defined by introducing convolutional layers in addition to fully connected layers. The example above displays a network that takes an image as input and uses convolution layers and max-pooling in the building blocks of the network architecture. The model is then flattened after the last max pooling, making it possible to apply subsequent traditional (fully connected) layers in the final parts of the network architecture.

To summarize, a convolutional neural network is a neural network that include convolution layers, in addition to/or instead of the ordinary hidden layers. A convolutional neural network can be considered as constructed by feature hierarchy, combining low-level features in a layer-wise fashion to form subsequent high-level features [64].

Chapter 3

Materials and methods

The following chapter will describe how the sound data sets were acquired, the data-prepossessing steps and detailed description of the model building (training) procedures.

3.1 Soundsensing IoT soundsensors

The sensors used by PNB are small IoT (Internet of things) sound sensors that stream real-time sound data. When installed, the sensor automatically connects to a mobile network and actively uploads data to Soundsensing's cloud database. This cloud platform possesses the ability to integrate a sound classification solution. The sensor uploads data with timestamps every 15-minutes. When using multiple sensors, alignment mismatches in timestamps could occur. One sensor may be ± 10 seconds ahead/behind another sensor in the worst cases.

3.2 Data collection

In most applications, good data quality is essential to obtain good results. One wishes to acquire data that represents the normal situation for regular days around the facility at PNB. However, the data available for this thesis was collected before the training facility was operational. The data was collected while acousticians tested the noise abatement measures. In such tests, each type of weapon was fired every thirty seconds in five rounds. This specific task tries to obtain a solution with the help of wireless sound sensors, but sound recorders were used for collecting data.

The data collection was done on four different days. The first two trips collected data from different gun types, while the final two trips acquired data from different types of explosions. In total, 512 different events took place on the four days of recording. Neighbors at the settlement were notified ahead of the tests and data collection, and signs with information were in place during data collection. The recorders were placed away from areas where pedestrians abide, avoiding recording of speech. All recordings were stored on a Google Drive folder for the later data analysis and classification modeling presented in this thesis.

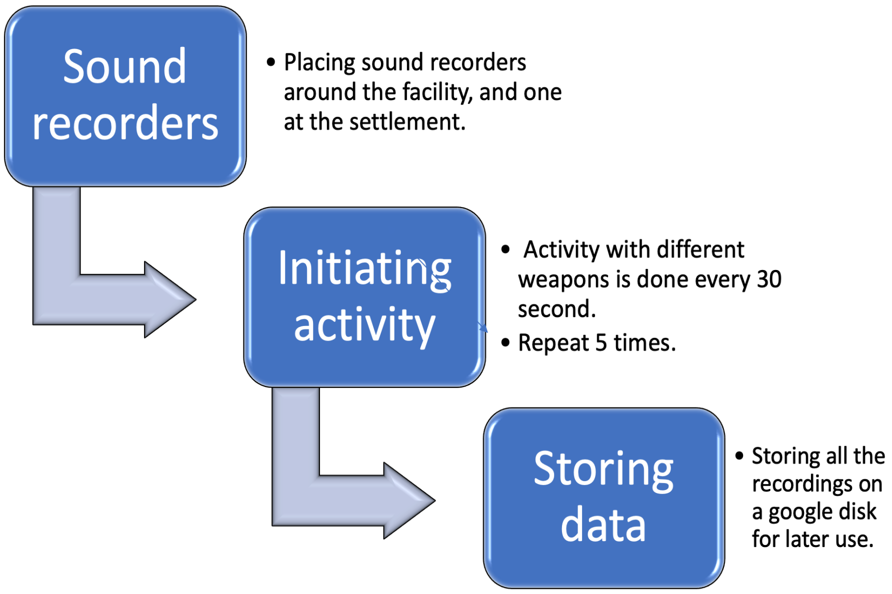

Sound recorders
• Placing sound recorders around the facility, and one at the settlement.
Initiating activity
• Activity with different weapons is done every 30 second.
• Repeat 5 times.
Storing data
• Storing all the recordings on a google disk for later use.

Figure 3.1: *Flow chart that describes the steps taken in collecting data.*

3.3 Data labelling

All activity at PNB took place under controlled environments, where we noted the specific time of activities. Based on the notes, recordings from PNB acted as verification on whether activity had taken place. When the activity was confirmed, the recordings from the settlement were checked to verify whether the noise was detected.

The program Audacity was used for the listening, which provides spectrograms and audio for a sound clip. Three labels were created from these clips:

- not heard
- faint
- clear

There were challenges regarding the distinction between faint and clear. It could be a misconception regarding how clear a sound is when actively listening for sounds. Therefore, when detecting a sound of high noise in the correct timeframe, the spectrograms were actively used to separate faint from clear. This task is handled as a binary issue where faint and not heard was labeled 0, while clear would be 1. The purpose of introducing the label faint was to serve as a helper for later evaluation of models. In total, 114 of the 512 events taking place at PNB were labeled as clearly heard around the settlement.

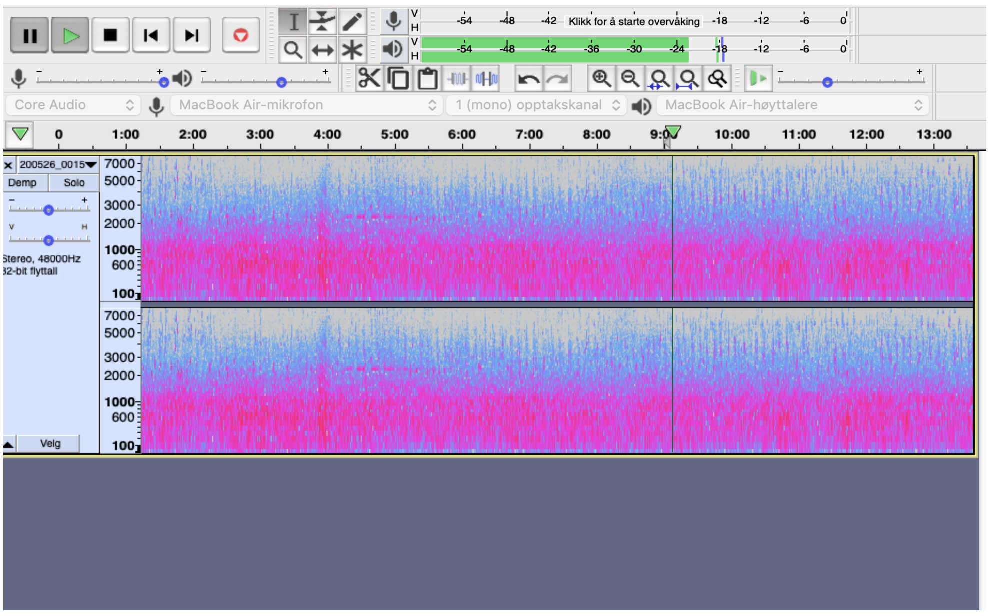


Figure 3.2: *The layout of Audacity. The program displays the spectrogram and sound for an audio clip.*

3.4 Handling privacy issues

The goal was to create a solution that could be implemented on wireless sound sensors. Due to privacy laws prohibiting speech recording, audio data should not be transmitted from or stored on the sensor. Wireless sound sensors possess the ability to preprocess data on the sensor. That allows the sensor to record audio, process it, and send only acoustical information to the cloud. The audio must be preprocessed with an integration time of at least 125 ms seconds for speech not to be understandable [65].

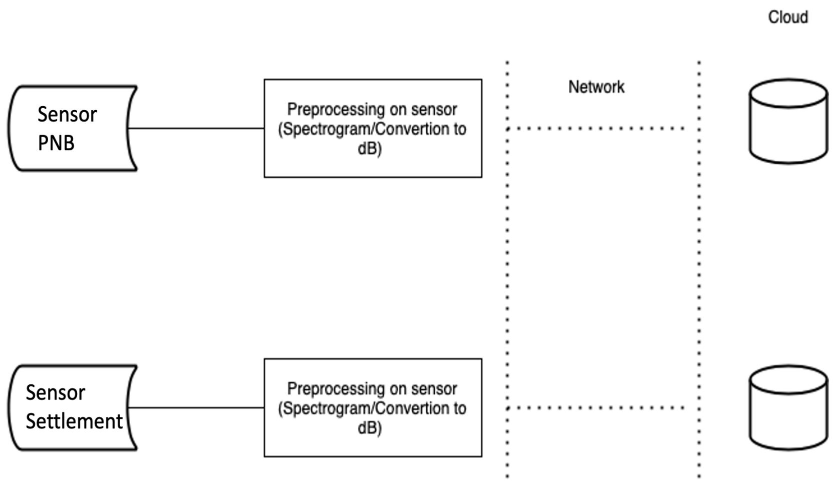

Cloud
Network
Sensor PNB
Preprocessing on sensor (Spectrogram/Convertion to dB)
Sensor Settlement
Preprocessing on sensor (Spectrogram/Convertion to dB)

Figure 3.3: *Illustration that describes how the data is preprocessed on the sensor before being sent up to the cloud. This is to avoid the privacy issues regarding storing audio.*

Figure 3.3 shows how the sensor would preprocess the data before sending it to the cloud, where implementation of the final solution is done. The sensor will send up audio in the form of spectrograms and dB measurements. These two types of data representation are what the solution of this thesis will build upon.

3.5 Choosing evaluation metric

The choice of evaluation metric represents a significant part of evaluating results, especially when handling data with a skewed distribution of classes. For this particular binary classification problem, we have around 3.6% of the input data (windows in the time series) being **noise** (group 1), whereas the remaining 96.4%

is **no noise** (group 0). Because of the group imbalance in the data, a simple evaluation metric as *accuracy* will most likely cause misleading models (if one were to predict no noise on all windows, an accuracy score of 96.4% would be achieved). Even though this score is good, such a classifier will detect no events (noise) of interest (the purpose of the current work is to establish a solution that detects noise around the settlement).

Therefore, we will consider the alternative metrics *precision* and *recall* described in chapter 2.4.2. The recall will serve as a metric giving an estimate regarding detecting noise events. Precision will serve as the metric evaluating the quality of predictions of noise. These metrics will serve as a good indicator regarding the performance of the different models. We will use the F1 score as a metric summarising the precision and recall score. Additionally, the average precision score is the metric used in optimizing some of the models.

3.6 Preprocessing

3.6.1 Time delay and creation of windows

Our problem had multiple time delay aspects. One was the time delay regarding the noise's movement from its source to the receiver. Based on the movement of sound, noise generated from PNB would take approximately 3 seconds to reach the settlement. Moreover, the sensors could have alignment mismatches regarding time. If this were to occur, the worst-case scenario would have one sensor +- 10 seconds ahead/behind another. Taking these aspects into consideration, we created overlapping windows. Each window covered 26 seconds of data with 13 seconds overlap to the next window.

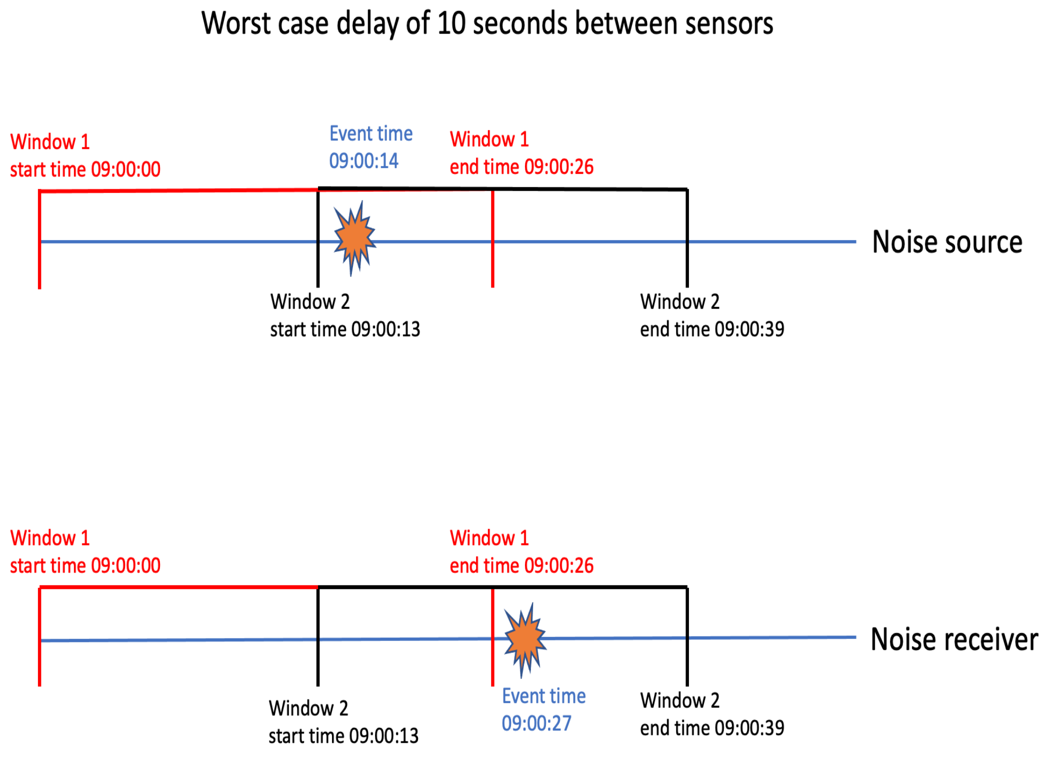


Figure 3.4: *Visualizing how overlapping windows catch a noise event in the worst-case scenario regarding time alignment. In this example, it takes 4 seconds before the sound generated at PNB reaches the settlement.*

The figure above displays how the overlapping windows govern the worst-case time delay. Window 1 at the source catches the event, while window 1 at the receiver misses it. However, due to overlap, window 2 at both the origin and the receiver catches the event.

3.6.2 Overlapping within windows - data augmentation

One creates overlapping windows to prevent a model from overfitting on where noise is detected. The technique is known as data augmentation. Data augmentation prohibits the model from learning patterns regarding the location of an impulsive noise event. We first created the windows and then implemented further overlap. From a window, sub-windows were created with 0.375 second overlap to the following sub-window. Each sub-window had 1 second of data.

With this implementation, each window would consist of multiple sub-windows.

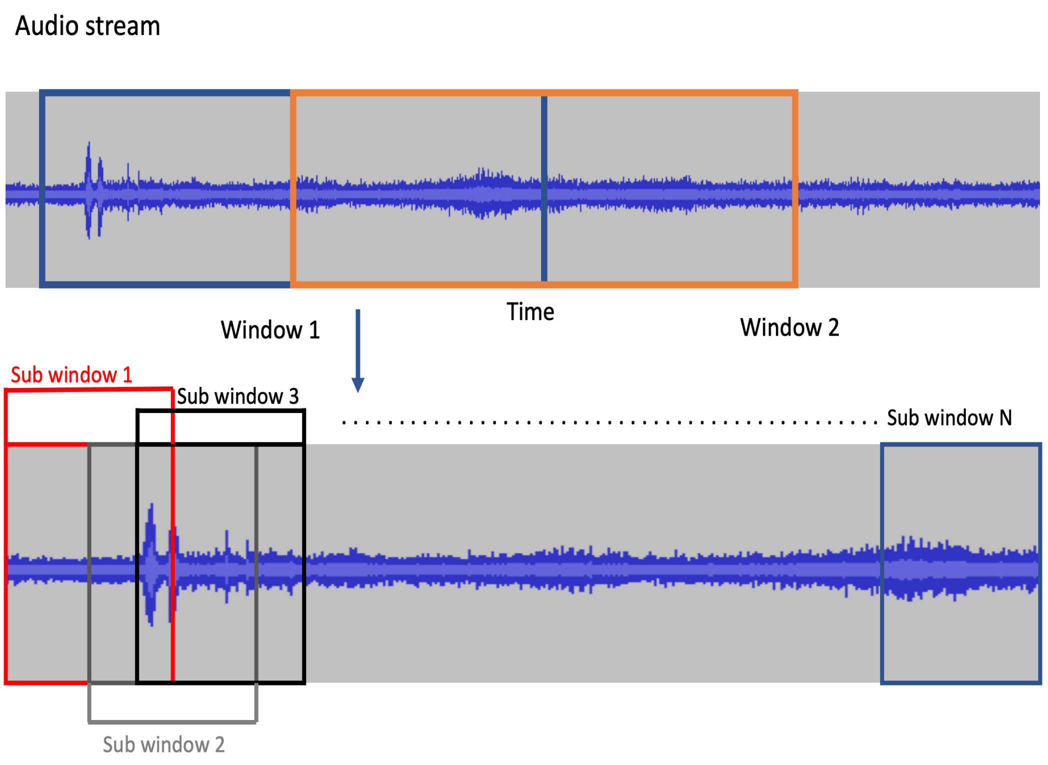


Figure 3.5: *The figure shows the creation of two overlapping windows from the continuous audio stream. Furthermore, multiple sub-windows are created from window 1.*

3.6.3 Mel-spectrograms and delta-spectrograms

Some of the models received spectrograms as input features. The spectrograms features were preprocessed with the following settings designed to prohibit recovery of speech [65]:

Settings	Values
Samplerate (Hz)	16000
Melfilter bands	32
FFT length (samples)	2048
FFT hop (samples)	2000

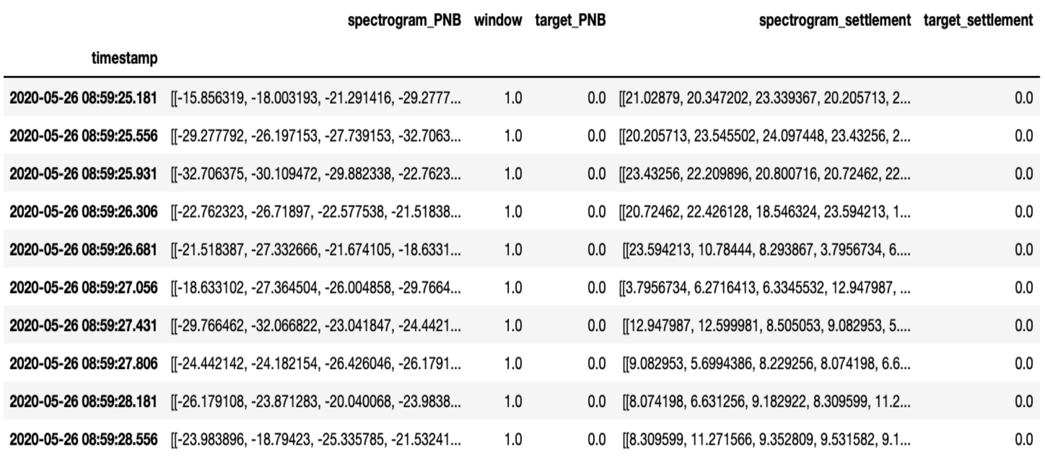
The models received input in the form of mel-spectrograms and delta-spectrograms.

The Python package Librosa was used in the creation of the spectrograms. For mel-spectrograms, one feature was decisive for detecting noise through spectrograms: the reference decibel level. Consequently, the reference level was the median of the data. With a reference level, every value lower than the reference will be negative. Contrarily, every value higher than the reference will be positive.

The delta-spectrograms were created with different delta order, varying in the range from 1 to 3.

3.6.4 Stacking delta-spectrograms

In search of increasing data quality to feed the model, we stacked multiple delta-spectrograms on each other. Different stacking combinations were used, but one constant was the inclusion of mel-spectrograms. The stacking consisted of different delta-spectrograms with varying delta order in the range of 1 to 3. The final data frame for the models taking spectrogram as input looked the following:

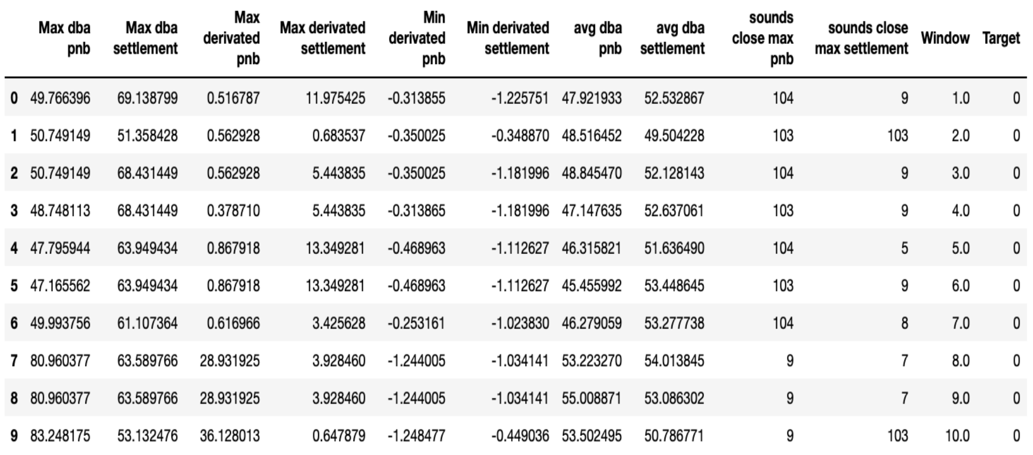


	spectrogram_PNB	window	target_PNB	spectrogram_settlement	target_settlement
timestamp					
2020-05-26 08:59:25.181	[[-15.856319, -18.003193, -21.291416, -29.2777...	1.0	0.0	[[21.02879, 20.347202, 23.339367, 20.205713, 2...	0.0
2020-05-26 08:59:25.556	[[-29.277792, -26.197153, -27.739153, -32.7063...	1.0	0.0	[[20.205713, 23.545502, 24.097448, 23.43256, 2...	0.0
2020-05-26 08:59:25.931	[[-32.706375, -30.109472, -29.882338, -22.7623...	1.0	0.0	[[23.43256, 22.209896, 20.800716, 20.72462, 22...	0.0
2020-05-26 08:59:26.306	[[-22.762323, -26.71897, -22.577538, -21.51838...	1.0	0.0	[[20.72462, 22.426128, 18.546324, 23.594213, 1...	0.0
2020-05-26 08:59:26.681	[[-21.518387, -27.332666, -21.674105, -18.6331...	1.0	0.0	[[23.594213, 10.78444, 8.293867, 3.7956734, 6....	0.0
2020-05-26 08:59:27.056	[[-18.633102, -27.364504, -26.004858, -29.7664...	1.0	0.0	[[3.7956734, 6.2716413, 6.3345532, 12.947987, ...	0.0
2020-05-26 08:59:27.431	[[-29.766462, -32.066822, -23.041847, -24.4421...	1.0	0.0	[[12.947987, 12.599981, 8.505053, 9.082953, 5....	0.0
2020-05-26 08:59:27.806	[[-24.442142, -24.182154, -26.426046, -26.1791...	1.0	0.0	[[9.082953, 5.6994386, 8.229256, 8.074198, 6.6...	0.0
2020-05-26 08:59:28.181	[[-26.179108, -23.871283, -20.040068, -23.9838...	1.0	0.0	[[8.074198, 6.631256, 9.182922, 8.309599, 11.2...	0.0
2020-05-26 08:59:28.556	[[-23.983896, -18.79423, -25.335785, -21.53241...	1.0	0.0	[[8.309599, 11.271566, 9.352809, 9.531582, 9.1...	0.0

Figure 3.6: *The dataset was created for models taking spectrogram as input. Only the spectrogram and target columns were input to the models. The window column is used in the the final evaluation.*

3.6.5 Feature engineering

Feature engineering is best described as good layers of representation for a dataset [66]. A machine learning model is only as good as the data it receives, hence the phrase 'garbage in, garbage out'. As explained earlier, this thesis works with window representations of the continuous stream of data. Each window has a classification mapped to it, and the prediction is made windows-wise with the random forest model. With feature engineering, a representation of the dataset is created.



	Max dba pnb	Max dba settlement	Max derivated pnb	Max derivated settlement	Min derivated pnb	Min derivated settlement	avg dba pnb	avg dba settlement	sounds close max pnb	sounds close max settlement	Window	Target
0	49.766396	69.138799	0.516787	11.975425	-0.313855	-1.225751	47.921933	52.532867	104	9	1.0	0
1	50.749149	51.358428	0.562928	0.683537	-0.350025	-0.348870	48.516452	49.504228	103	103	2.0	0
2	50.749149	68.431449	0.562928	5.443835	-0.350025	-1.181996	48.845470	52.128143	104	9	3.0	0
3	48.748113	68.431449	0.378710	5.443835	-0.313865	-1.181996	47.147635	52.637061	103	9	4.0	0
4	47.795944	63.949434	0.867918	13.349281	-0.468963	-1.112627	46.315821	51.636490	104	5	5.0	0
5	47.165562	63.949434	0.867918	13.349281	-0.468963	-1.112627	45.455992	53.448645	103	9	6.0	0
6	49.993756	61.107364	0.616966	3.425628	-0.253161	-1.023830	46.279059	53.277738	104	8	7.0	0
7	80.960377	63.589766	28.931925	3.928460	-1.244005	-1.034141	53.223270	54.013845	9	7	8.0	0
8	80.960377	63.589766	28.931925	3.928460	-1.244005	-1.034141	55.008871	53.086302	9	7	9.0	0
9	83.248175	53.132476	36.128013	0.647879	-1.248477	-0.449036	53.502495	50.786771	9	103	10.0	0

Figure 3.7: *The dataset created with feature engineering. It has 10 exploratory variables. Only the first 10 rows are displayed.*

One window contains multiple samples, with decibel measurement and timestamp mapped to it. The feature engineering process creates the ten exploratory variables observed in the figure above, window-wise.

3.7 Model evaluation

Throughout the process, the models trained on different representations of the same data. The data from the settlement had an additional 10 seconds added to its timestamps to reflect a worst-case scenario of time delay between sensors.

Splitting the data

The movement of sound is affected by multiple factors but mainly the weather. The weather affects how fast the sound moves and how clear its heard. Higher temperature creates an increase in the speed of sound. Contrarily, the speed decreases when it is cold. Due to sound propagating slower in a less dense material, different types of weather would affect how well sound is heard [14].

The factors of the weather contributed to the splitting of data. Data within specific timeframes was in the same train/test split to assure no correlation between the splits. That was due to weather and environmental surroundings' effect on the sound. Additional quality checks were done based on the timestamps to ensure no data leakage between the splits.

Training phase

Every model was trained with K-fold cross-validation to achieve the best generalized model. The number of folds used in the training process was $K = 5$. For models based on CNN, the epoch with the highest score on the validation data were selected in each k-fold. Based on the scores achieved in the cross-validation, the best performing models were used to evaluate the test data.

The models were evaluated differently in the training phase. The random forest model was optimized and trained on window-wise data, while the neural networks were trained on sub-windows. That is explained in detail in chapters 3.6.1 and 3.6.2 below.

However, the models were compared based on their precision and recall score achieved on the test data. The prediction on test data was made window-wise for both models, making the results comparable.

3.7.1 Predicting noise with random forest

After finalizing the training and test splits for the decibel representation of the data, the data were standardized. Optimization of the hyperparameters was done with Sklearn's built-in grid search. From this optimization, the parameters that returned the highest precision score would serve as the final model. Optimizing on precision appeared to give better overall results when training on decibel representation of the data. The hyperparameters tuned could be observed in the table below:

Parameter	**Value**
number of estimators	[10,50,100,200,500]
maximum depth	[1,10,50,100]
minimum samples leaf	[2, 50, 200, 1000, 2000]
max features	[1,3,5,8,10]

Table 3.1: *The hyperparameters tuned in the random forest model*

3.7.2 Bundled model - convolutional neural networks combined with cross correlation

In order to utilize the two sensors placed at PNB and around the settlement, a combined solution was created. Firstly, two individual convolutional neural network models were created based on the data from the two locations (PNB and the settlement). Hence, a model for the data from PNB and a separate model for the data from the settlement. CNN was a natural choice due to achieving good results on environmental sound classification [67][68]. Each model got data from the same windows, broken down into sub-windows within the main window as described in 3.6.2. The sub-windows give a more detailed spectrogram from which the model could learn patterns.

Subsequently, the two models predicted whether or not an impulsive noise event was detected in the sub-windows. The output (prediction) for each sub-window inside a window was stored and compared. The steps could be summarized in the following way:

1. If the PNB model detects no activity (noise) at the training facility, noise around the settlement must have another origin. The final prediction for the window is 0.
2. If the PNB model detects activity (prediction > 0.8),
   the maximum prediction from the settlement model is collected.
3. If the settlement model has a maximum prediction higher than 0.6, it is considered the representative prediction for the window.
4. If the maximum prediction is lower than 0.6, we calculate the maximum cross-correlation between the predictions from both models. The maximum

cross-correlation between the prediction stands as the final prediction for the window.

Figure 3.8 shows how the model uses the PNB and settlement model predictions to acquire a prediction for a window.

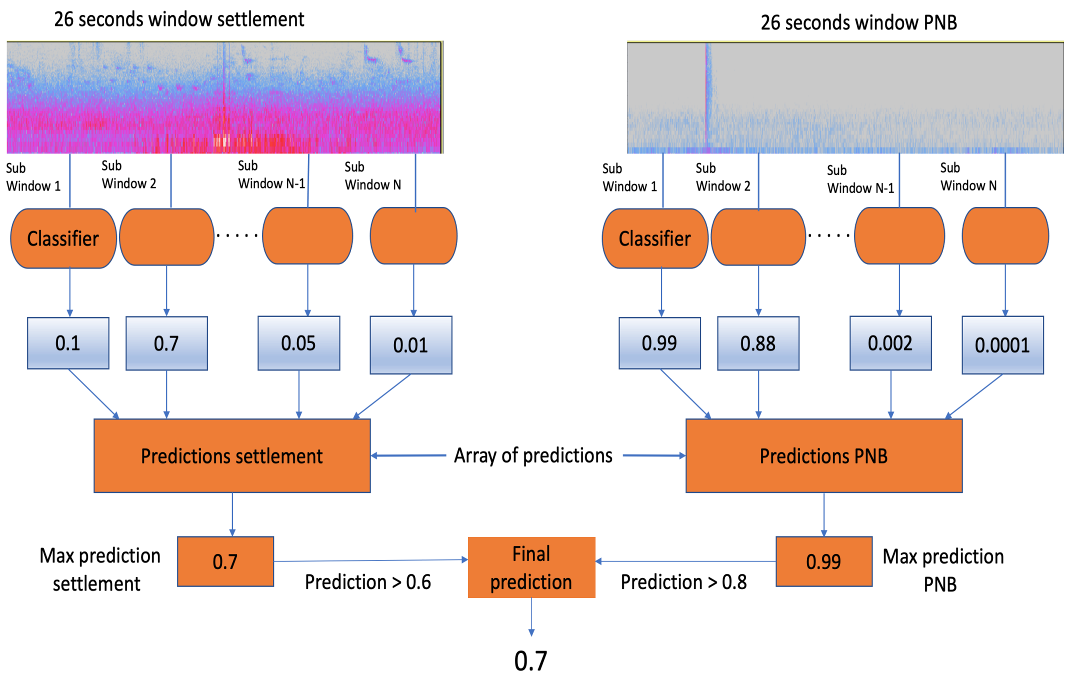


Figure 3.8: *The process of combining two convolutional neural networks to predict noise. Each model gives a prediction on sub-window n created from the 26 seconds window. Every prediction is stored in an array, and the highest probability for noise predicted is extracted. If the prediction at the PNB model is over 0.8, an event at PNB has taken place. Additionally, if the settlement model's max prediction is bigger than 0.6, that will serve as the final prediction.*

The model explained above (implementing and combining the PNB and settlement model) will be referred to as the bundled model.

Both the models for predicting noise at PNB and the settlement were optimized regarding the average precision score. The two models had the same architecture. The architecture can be seen in the figure below:

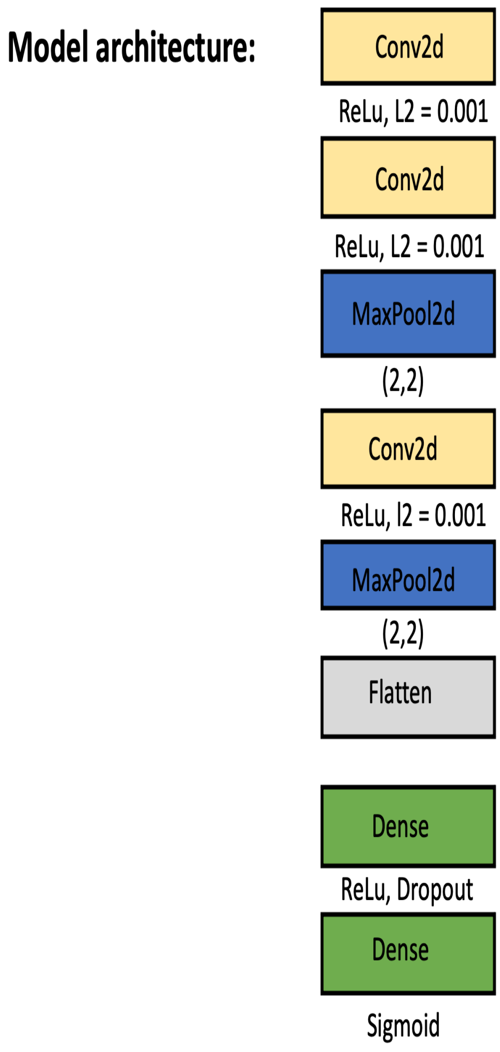


Figure 3.9: *The architecture of the models to predict noise. Both the PNB and settlement model have simple architectures.*

3.7.3 Difference in representation of data

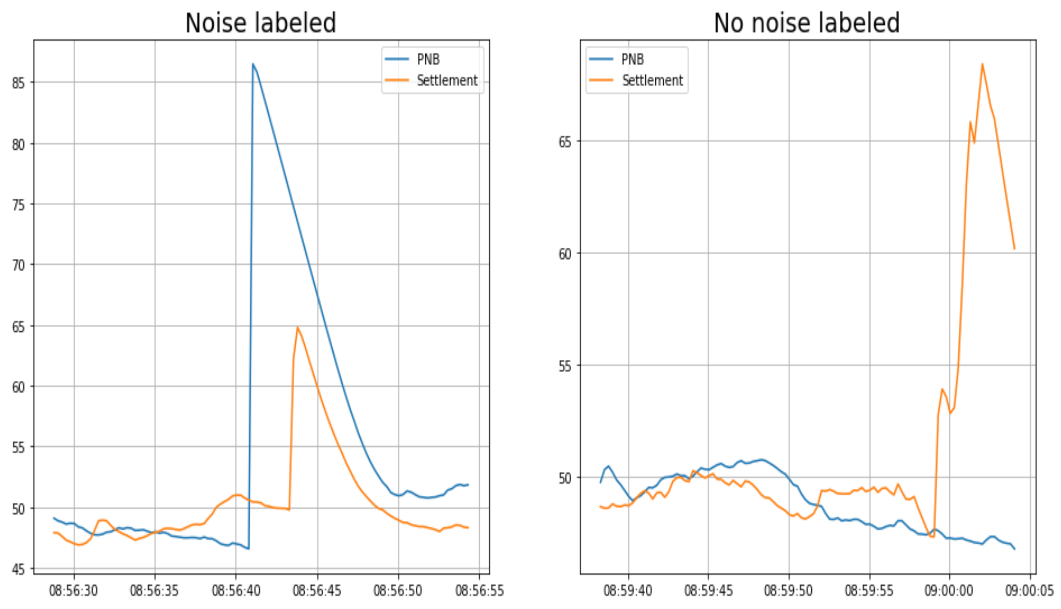


Figure 3.10: *The figure displays the decibel measurement representation of the data.*

The figure above visualizes the decibel measurement representation of the data. The first plot displays an event where PNB was the origin of the impulsive noise event at the settlement. We observe a peak in the decibel measurements at the settlement and PNB within the same timeframe. The second plot displays another scenario. No event transpired at PNB, but there is a clear spike in the decibel measurements from the settlement.

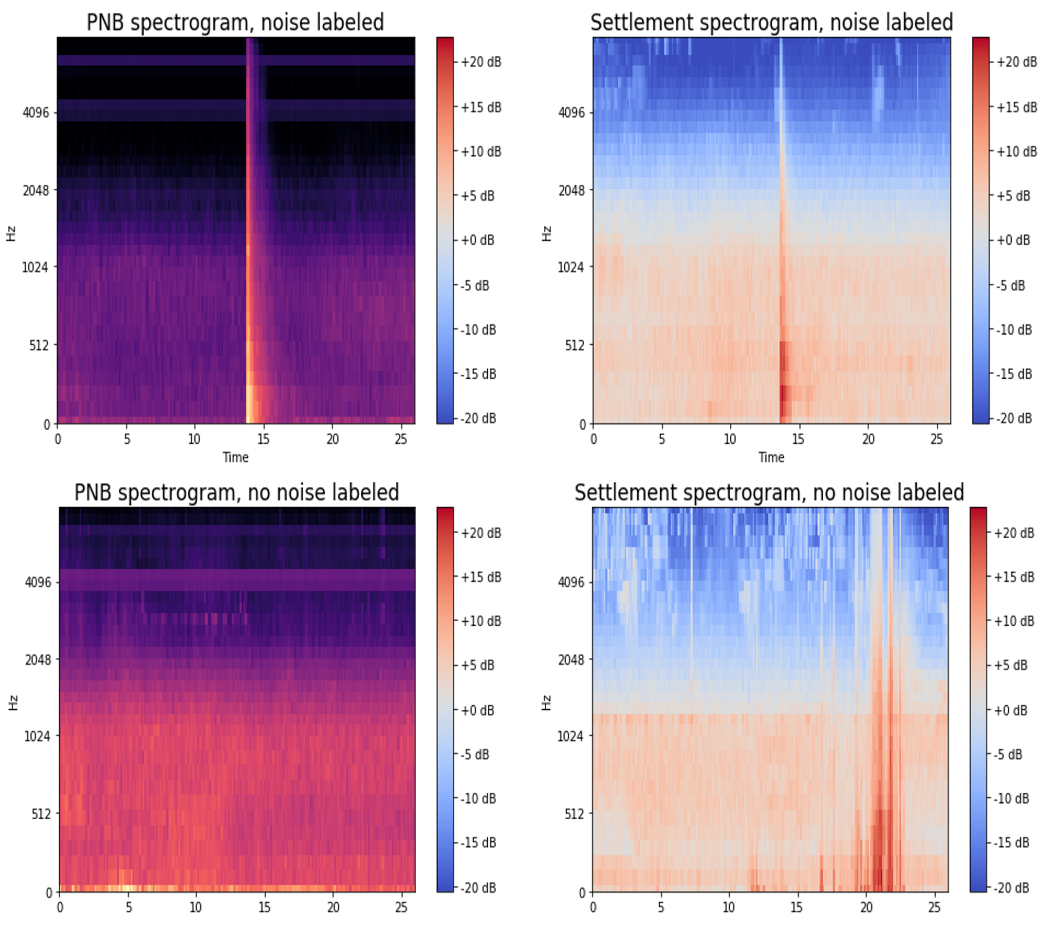


Figure 3.11: *The figure displays the spectrogram representation of the data.*

Figure 3.11 shows the spectrogram representation of the same data. The top plot displays the event where PNB was the origin of the noise measured at the settlement. The bottom plot displays the event where PNB was not the origin of the noise measured at the settlement. Note how different the two noise patterns in the settlements spectrogram are. There is a higher amplitude/dB in the lower frequency of the spectrogram when PNB is the origin of the noise.

Chapter 4

Results

The following chapter presents and discusses the classification results obtained by the different modeling approaches (the random forest modeling and bundled model).

4.1 Noise detection with a random forest classifier

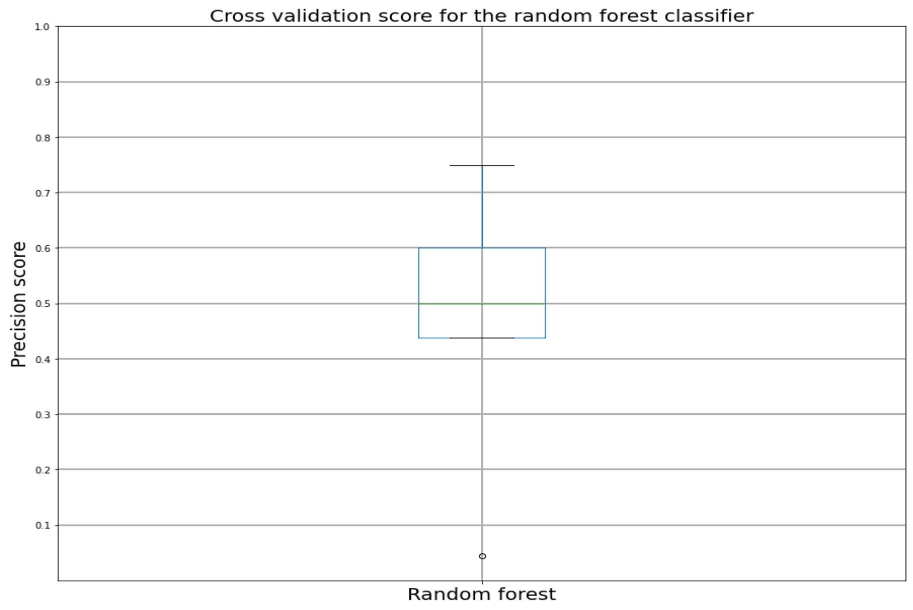


Figure 4.1: *The figure displays the precision score from the cross-validation for the random forest model.*

The figure above visualizes the precision score obtained by the random forest model in the cross-validation. The plot indicates a low mean precision score and high variation in scores. The mean precision score from the cross-validation is 46.6% and has a standard deviation of 23.3%, which confirms the high variation. That implies that some of the models in the cross-validation process score a very low precision.

From the cross-validation process, the best model was chosen for predicting the test data:

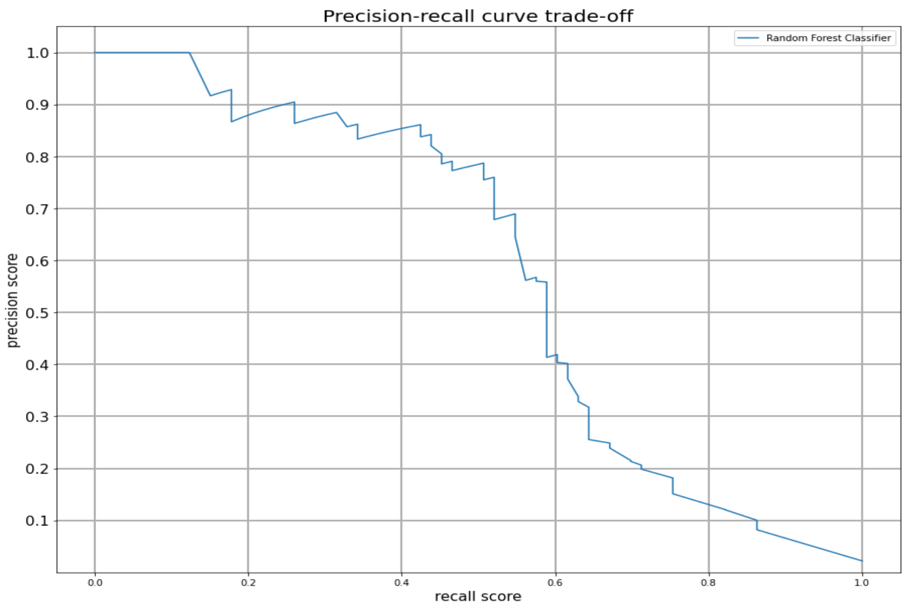


Figure 4.2: *The precision-recall curve for the random forest model*

Figure 4.2 shows the trade-off between recall and precision. The lower the precision score, the higher the recall score. Note that the precision score decreases relatively slowly until the recall reaches about 0.5. From this point, the precision decreases rapidly, and when the recall is approximately 1, the precision is close to 0. The optimal F1 trade-off appears to be where the score is 61.7% (when the precision is 76.0% and the recall 52.0%). In this situation, the model detects 52.0% of the events where PNB causes noise at the settlement and correctly detects noise in 76.0% of the positive predictions.

4.2 Convolution Neural Network models result

Here we present PNB's and settlement's convolutional neural network models results from the training process. These models were trained on sub-windows, consisting of 1 second of audio in a spectrogram representation.

4.2.1 Detection of impulsive noise at the settlement

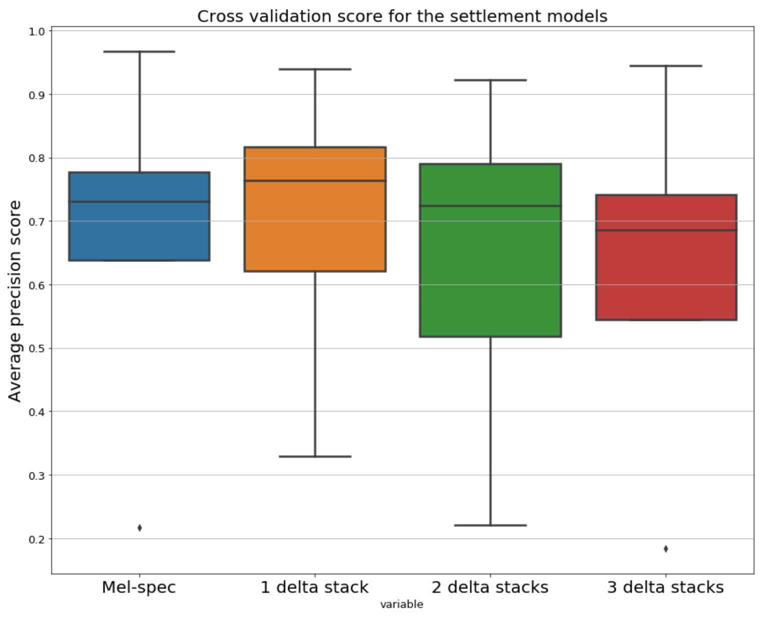


Figure 4.3: *Boxplot of the average precision scores from the cross-validation. Mel-spec is the model evaluated simply on the mel-spectrogram representation. 1 delta stack is the model that stacks the delta stack of first-order onto the mel-spectrogram. 2 delta stack is the model that stacks the delta stack of first and second-order onto the mel-spectrogram. 3 delta stack stacks three delta stacks with an order from 1 to 3 onto the mel-spectrogram.*

Figure 4.3 shows the convolutional neural network's average precision scores when predicting impulsive noise around the settlement with different stacking methods.

The method using mel-spectrogram achieves a better mean average precision score than 2 and 3 delta stacks. However, 1 delta stack scored the highest mean average precision. The improvement is an increase of 2.8% from the next best model, the mel-spec. The boxplot indicates a considerable variation between scores from each fold in the cross-validation. That is due to some folds of validation data consisting mainly of noise generated from explosions at PNB. Generally, such impulsive noise events are easily detected. Oppositely, noise originating from weapons with silencers is harder to detect. The results from the different methods are shown in the table below:

Model	Mean average precision	Standard deviation
Mel-spec	66.5%	27.8%
1 Delta Stack	69.3%	23.3%
2 Delta Stacks	63.4%	27.3%
3 Delta Stacks	62.0%	28.3%

Table 4.1: *The scores from the cross validation for the settlement models.*

The table displays that the mean average precision of the 1 delta stack model is the highest. Coincidentally, the model has the lowest variation (23.3%) in score from each fold in the cross-validation. The Mel-spec model performs better than the 2 and 3 delta stack models, which scores some percentages lower.

4.2.2 Detection of impulsive noise at PNB

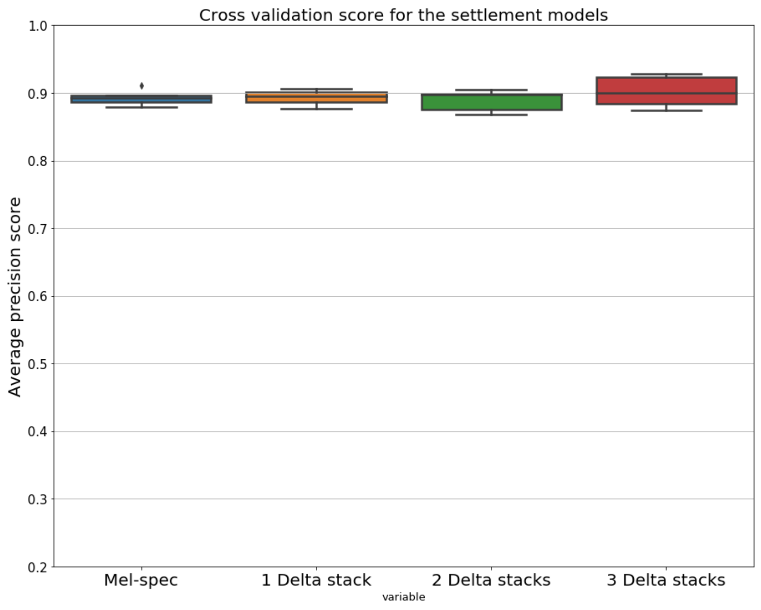


Figure 4.4: *The figure displays the mean average precision of the models. The definition of the columns is the same as in figure 4.3*

The same stacking of delta spectrograms for the settlement models was used in the PNB models. The models mel-spec, 1 delta stack, and 2 delta stacks achieve similar scores. However, 3 delta stacks shows a slightly larger value in the mean average precision score (by 0.9%). The variation in scores between each fold was significantly lower than for the settlement models. Activity at PNB appears relatively easy to detect, regardless of the weapon type. The table below shows the mean average precision scores with corresponding standard deviations for each of the PNB models:

Model	Mean average precision	Standard deviation
Mel-spec	89.2%	1.1%
1 Delta stack	89.3%	1.1%
2 Delta stacks	88.9%	1.6%
3 Delta stacks	90.1%	2.3%

Table 4.2: *The average precision scores from cross validation for the PNB models.*

Table 4.2 displays the mean average precision and standard deviation for the models. As mentioned above and confirmed by the table, all the models have low variation within the folds.

4.3 Impulsive noise detection with the bundled models

Section 4.2.1 and 4.2.2 presented the individual models created for detecting impulsive noise around the settlement and PNB. These models would serve as the foundation of a bundled model, consisting of a model each from PNB and the settlement (ref section 3.7.2). Furthermore, two bundled models were created. These two models are denoted *bundled one* and *bundled two*. The *bundled one* model was a compound of the mel-spec model from the settlement and PNB's 3 delta stacks. The *bundled two* model consisted of the best models from the settlement (1 delta stack) and PNB (3 delta stacks).

The bundled models were used to predict the test data (window-wise) and determine whether PNB caused impulsive noise around the settlement. The scores are visualized in the figure below.

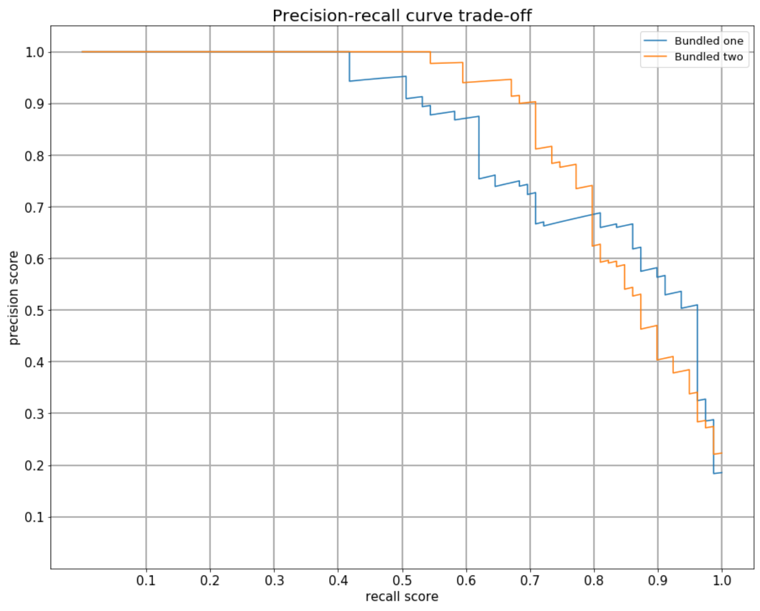


Figure 4.5: *Precision-recall curve for the two best models from the settlement. The labels* bundled one *and* bundled 2 *represent the bundled solutions.*

We observe an optimal trade-off for the *bundled one* model in terms of maximum F1 score (79.43%). In this optimal trade-off, the recall score is 70.8%, and the precision score is 90.3%. Conclusively, this model identifies 70.8% of the impulsive noises caused by PNB and correctly predicts these on 90.3% of the instances.

For the *bundled two* model, an optimal trade-off result yields an F1 score of 75.13% based on a recall score of 86.07% and a precision score of 66.6%. In the optimal trade-off point, the model would identify 86.07% of the instances where PNB causes noise in the settlement, and 66.6% of its noise predictions are correct.

The two bundled models achieve optimal trade-offs, which is contrary to one another. The *bundled one* model detects fewer cases of impulsive noise events. However, it detects the events with high accuracy. The *bundled two* model detects many cases of impulsive noise but detects these with moderate accuracy.

Note that the bundled solution benefited from the settlement and the PNB model overlaps from sub-windows. Recall the results from section 4.1.1 and 4.1.2, where the PNB and settlement models' results were presented. The results were based on the mean average precision score achieved on detecting noise in sub-windows. Furthermore, the sub-windows had overlaps that exposed the models to impulsive noise events at different points in a spectrogram. Both bundled models stored predictions from each window, taking the highest prediction from the settlement model and PNB model. Thus, creating multiple chances for the model to detect impulsive noise in a window.

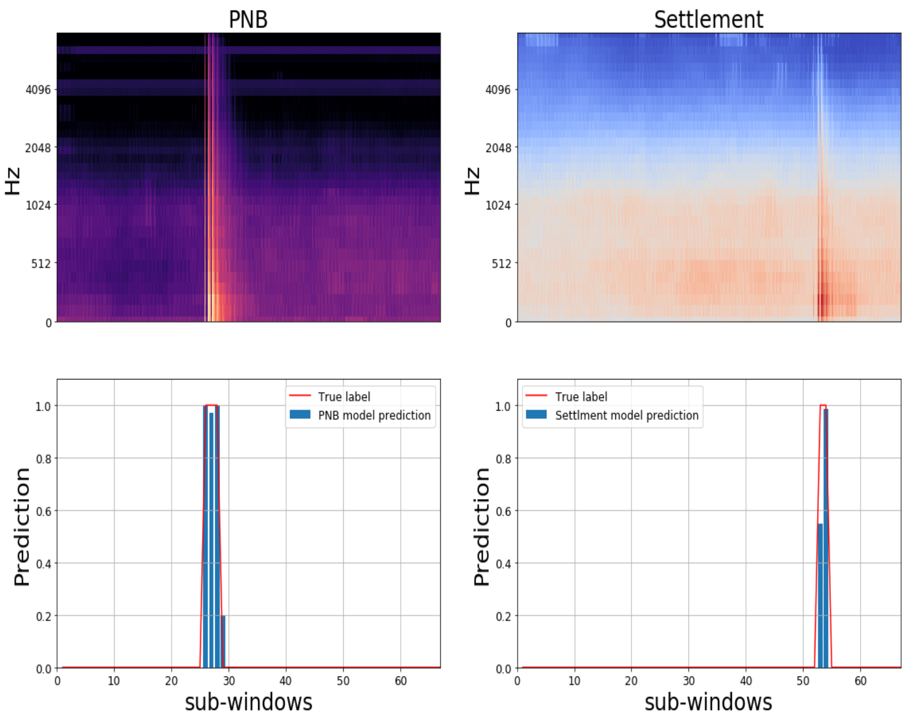


Figure 4.6: *The spectrograms from the same windows at PNB and the settlement are visualized. The overlapping spectrograms in a window are merged for the visualization. Below the spectrograms is the models' prediction within the same window.*

The figure above shows the spectrograms for a window at PNB and the settlement where noise is detected. Furthermore, it displays the *bundled one* model's prediction on the sub-windows within a window. The PNB model detects each overlapping sub-window with impulsive noise. In contrast, the settlement model detects one of the two sub-windows with impulsive noise in them. Subsequently, the overlap gave the settlement model two chances to detect the noise. The maximum prediction from the settlement stood as the final prediction for the window (predictions higher than 0.6 do not use cross-correlation), giving the bundled solution the benefit of overlap.

4.4 Comparing recall/precision trade-off between the models

To summarize this section, the three final models will be compared in regards to the precision-recall trade-offs. The training process was different for the two models, as the random forest model trained on window-wise data. Contrarily, the foundation models (PNB and settlement models) in the bundled models were trained on sub-windows. However, prediction on test data was similar for both models (window-wise predictions), making them comparable.

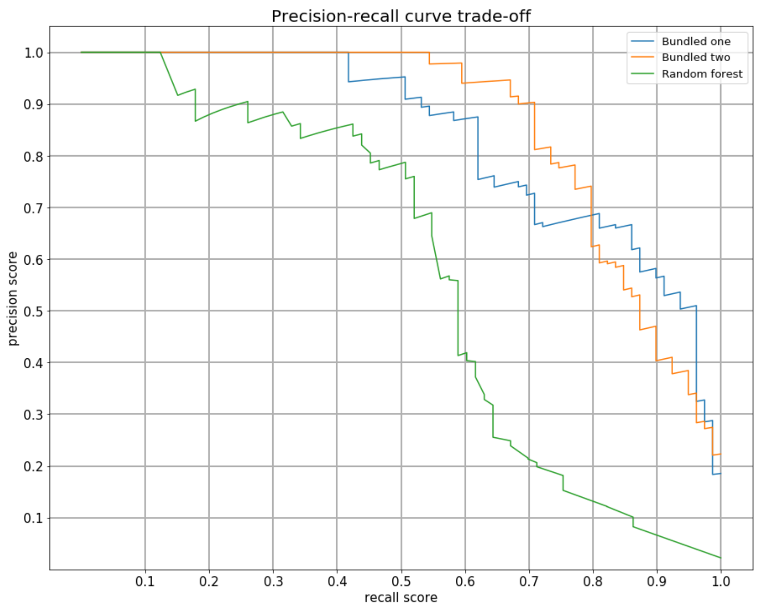


Figure 4.7: *The precision-recall curves for all the models.*

Figure 4.7 shows the precision-recall curves for all the models. The bundled models separate from the random forest model, achieving better recall and precision scores independent of the choice of thresholds. The decrease in the precision score is rapid for the random forest model around a recall score of 0.5. There is a trade-off in precision and recall score following this point. The precision decreases significantly as the recall score increases. The *bundled one* and *bundled* *two* models decrease in precision around a recall score of 0.5 and 0.57, respectively. The precision score of the *bundled one* model is higher than *bundle two* until a recall score of 0.8. Following this point, *bundled two* have higher precision with an equivalent recall score, giving it the better F1 score.

Model	Optimal tradeoff (F1)
Random Forest Classifier	61.7%
Bundled one model	75.13%
Bundled two model	79.43%

Table 4.3: *The F1-optimal tradeoffs for the different models results. The numbers are obtained from the underlying data of figure 4.7.*

The table above summarizes the F1-optimal trade-offs from all the models. The *bundled two* model scores the best, followed by the *bundled one*. Both CNN solutions combined with cross-correlations (denoted bundled model) achieve a better F1-score than the random forest model based on decibel measurements. To summarize, the results were decent for the bundled models. The random forest model resulted in poorer predictions and a correspondingly low optimal F1 score.

Chapter 5

Discussion

5.1 Discussion of method

5.1.1 Shortage in data collection

In this study, data was, as mentioned in chapter 3.1, collected under controlled conditions. Every activity was completed once every 30 seconds, which in future cases is not realistic. Activity at shooting fields will typically consist of multiple gunshots within few seconds. We assume that multiple gunshots will create an increase in noise. Therefore, the results could look different with these scenarios included.

5.1.2 The creation of data labels

The processes of data labeling lacked the robustness of multiple operators. Classifying whether impulsive noise was detected at the settlement was largely subjective. Throughout listening to recordings, the times of the noise events at PNB were known. That created a situation where the listener was aware of activity at PNB with a specific timeframe and actively listened for impulsive noise in the recordings from the settlement. There could be a significant difference between pedestrians walking around the settlement and their perception of noise heard. The spectrogram displayed in Audacity provided remarkably, but more clarity would be achieved with none active listeners. Detailed in section 2.4, a model learns from labeled training data as examples of the input-output relationship to be learned. The model could struggle to learn patterns if the difference between faint (labeled 0) and clear (labeled 1) is unclear.

5.1.3 Choosing models

We evaluated different approaches to solve the task. One fundamental interest was creating two different models trained on different representations of the same data. Hence, the random forest on decibel measurements and the bundled model (CNN's) on spectrograms. Cross-correlation emerged initially as a natural technique to detect impulsive noise events with decibel measurements. Cross-correlation would compare two signals and account for time delays. However, this approach had little success in predicting the data. We, therefore, chose the random forest classifier to represent the model based on decibel measurements.

Centered on the promising results achieved by CNN's on environmental sound classification [67][68], the thesis sought to create a solution implementing it. The bundled model was created, implementing two individual CNN's and cross-correlation. Creating a bundled model was a consequence of the time delay issues this thesis encountered.

Time delays implication on model choice

One significant aspect of this thesis was the multiple time delays. We had time delays regarding the movement of sound from PNB to the settlement and misalignment between sensors.

This issue was dealt with by creating overlapping windows. The overlap from one window into the subsequent window had to be long enough to capture both the time delay in the movement of sound from PNB and the worst-case scenario of misalignment between the sensors. We assumed that the sound would take approximately 3 seconds to move from PNB to the settlement. Conclusively, the overlap was 13 seconds to account for the time delays.

The time delay affected the final solution. In a scenario with no time delays, we would implement a Siamese network. One convolutional neural network taking two inputs. The inputs would be spectrograms (1 second long) from PNB and the settlement for a specific timestamp. That approach would utilize the two data streams, possibly enhancing a model's ability to detect PNB's effects on the settlement. Nonetheless, time delays prohibited this approach. The timeframe between two inputs in the form of spectrograms would not be alike in a time delay situation.

One could, however, implement a siamese network receiving spectrogram representation of data from a window (26 seconds long). The issues concerning this are the lack of data. The dataset (for spectrograms) would be significantly reduced in size with 26 seconds long spectrograms as input.

Due to this, the bundled model presented itself as a promising approach. The bundled model is simple and does not optimally utilize the two data streams. However, it tackles the issues of time delays while implementing the use of CNN's.

5.1.4 Architecture selection for CNN's

This thesis presented one architecture for the CNN models. However, multiple architectures were implemented and tested. The architectures evaluated built upon the present architecture, increasing the complexity. The increasing complexity implied adding further convolutional and dense layers. Subsequently, the increased complexity led to overfitting. Therefore, we chose the presented architecture as it achieved the best bias-variance tradeoff. Coincidentally, both the PNB and settlement model achieved the best results with identical architecture.

5.2 Discussion of results

5.2.1 Random forest results

The random forest model differed from the other models in regards to optimizing the precision score. The better results stemming from this could be due to the effects noise had on decibel measurements. When detecting noise around the settlement, the sudden increase in decibel was easily observed. However, there were issues surrounding noise not originating from PNB having similar patterns and noises not creating sudden spikes in decibel. Therefore, optimizing on precision would detect the noise events originating from PNB with a better percentage. Consequently, it would lead to detecting fever impulsive noise events.

Overall, the random forest model had substandard noise detection, with a mean precision score of 46.60%. Figure 4.4 showed the model achieving an optimal tradeoff with an F1 score of 61.7% on the test data.

5.2.2 Bundled models results

The bundled model had the advantage of comparing and extracting the highest prediction from each window. With overlaps within the sub-windows, impulsive noise events could appear multiple times in a window. That gave the bundled model the benefit of only needing to detect one sub-window with impulsive noise, which we observed in figure 4.6.

With PNB being the origin of impulsive noises generated, we assumed that the PNB model would easily detect activity taken place. The models' performance confirmed the assumption that falsely predicting activity at PNB was a rare incident.

Detecting noise at the settlement originating from PNB had a different degree of difficulty. That proved to be a more difficult task due to the background noises originating from other sources in the surroundings (such as lifting containers, cars driving, people screaming, and birds singing). We used different approaches for increasing detection, such as the use of delta stacks. The delta stacks did not improve the average precision scoring percentage, except with the 1 delta stack. This model scored the highest mean average precision during the training process with a precision score of 69.3%. The multiple delta stackings' poorer results could stem from the increased channel dimension in the shape of the spectrograms. With each stacking of delta-spectrograms, we get an extension in channel dimensions. That created increased variation in the models (2 and 3 delta stacks), which lead to overfitting.

Figure 4.5 showed that the *bundled two* model achieved an optimal F1 trade-off score of 79.43%, followed by the *bundled one* model achieving 75.13%. The best model (*bundled two*), which included 1 delta stacking in the settlement, achieved a better optimal trade-off by approximately 4%. The results suggest that including one delta spectrograms in the training process leads to more precise models.

5.2.3 Comparing the models

The models (bundled models) trained on spectrograms predicted significantly better than the random forest model based on decibel measurements. The random forest model was relatively insensitive to the impulsive noise events not creating sudden spikes in the decibel level. Furthermore, as visualized in chapter 3.6.3, there are more details in a spectrogram. The impulsive noise events originating from PNB had detectable patterns, especially in the frequency level. The frequencies were rarely above 4000 Hz, making it possible to separate the noise events originating from PNB from some of the daily background noise around the settlement. Contrarily, spikes in decibels originating from PNB were challenging to separate from spikes created by regular background noise around the settlement.

Note that the models rarely produced false positives (predicting that PNB caused impulsive noise in the settlement when there were no activities at PNB). That implied that the models detected activity conducted at PNB with good precision. Data recording happens continually throughout a day when implemented in a realistic situation. Due to PNB only being allowed to carry out activity between 7-19 (described in 2.1.6), a considerable proportion of the data recording is when PNB is closed. Models detecting activity at PNB with good precision prohibit the model from predicting noise at the settlement when no activity at PNB is conducted. That filters out the models predicting noise originating from elsewhere.

5.2.4 Regarding thresholds

In chapter 4.4, focusing on the summary and presentation of the results, no thresholds were selected. That was mainly due to the specific use case with PNB and Soundsensing AS. When training machine learning and neural network models, one would usually explore multiple thresholds during the training process. From this process, the best threshold could be identified. Due to this thesis focusing on both precision and recall score, we would optimize the threshold based on these metrics. However, as this thesis sought to create a solution to a real-life problem, skepticism raised whether an optimized threshold would capture what was needed.

The focus of PNB is to detect whether the impulsive noise events they generate affect the settlement. As illustrated in the trade-off curve presented in figure 4.4, there is a continuous trade-off in precision and recall score when the threshold changes. If the optimal threshold obtained from training were low, the recall score would be excellent. That implies that the model detects a good portion of the noise events generated by PNB. Contrarily, the low threshold would give a lower precision score, increasing the false positive detection of the model.

In summary, the preferred trade-offs should be decided by the customer, i.e., the threshold should be chosen according to the preferences of PNB. Assuming that they want to detect as many noise events as possible, a low threshold may be chosen. Alternatively, if they want to detect noise events with a low fraction of false positives, a higher threshold should be chosen.

5.3 Practical evaluation of the solution

The specific use case has mainly two interest parties, PNB/municipality and the population in the settlement. One assumes that PNB/municipality wants a solution that detects impulsive noise events with high precision. That is due to PNB taking responsibility when such an event occurs. A solution with low precision creates situations where PNB takes responsibility for noise not originating from them. Contrarily, the neighbors in the settlement may prefer a solution that detects most of the noise events originating from PNB.

Per the solution presented and evaluated in chapter 4.4 the *bundled two model* had an optimal trade-off (based on optimal F1 score) predicting 90.3% of the noise events correctly and detecting 70.8% of the events. PNB would benefit significantly from this model's optimal trade-off, as precision is high. Contrarily, the detection of noise events is not high enough for the settlement. The next best model (*bundled one*) had an opposite optimal trade-off scenario, where the detection of noise events was high. However, the precision was low, causing false detection of some events.

As the developed solution has a considerable trade-off between precision and recall, the performance may not be sufficient to implement into an automated noise monitoring system. The solution could alternatively be used as a semi-automated tool, where detected events can be verified afterward by a human.

5.4 Future work

Future work should include a collection of data that represents days of more realistic PNB-activities. The present models are trained and optimized based on data where an event took place every 30 seconds. Activities with multiple shots in the same time frame could create different impulsive noise patterns. The model building could benefit significantly from learning the patterns of noise generated by such activities. It is likely to assume that multiple gunshots fired at the same time may generate more noise. However, future work could implement synthesizing noise events with more than one activity per window with the current data if this is not possible. There may also be challenges regarding synthesizing the data around the settlement due to the constant background noise.

The author argues that a large proportion of difficulty in this thesis is creating the pipeline, handling privacy concerns, and handling time delays. Therefore, future work could focus on optimizing the model that represents the settlement in the bundled solution.

The implementation of siamese networks is of interest in future work. If one could develop a method handling the variation in time (time delays), the siamese network presents itself as a promising idea. It could utilize the two sources of data (data from PNB and the settlement) better than the bundled model.

In future work, one could implement more wireless sound sensors as this thesis focused on solving the problem using only two sound sensors (one at PNB and one at the settlement). An additional sensor could be placed somewhere between the two already existing sensors, gathering additional information. The inclusion of thresholds for the classification models must be done in future work to determine the model's primary focus: Detection of many noise events or detection of noise events with high accuracy.

Chapter 6

Summary and conclusion

In the present work, we described the entire pipeline for implementing a noise detection solution based on data from sound recorders. The process included data collection, data labelling, pre-processing, and model building.

There were two primary concerns regarding the audio recording with wireless sound sensors: i) privacy concerns and ii) misalignment between multiple sensors. The work in this thesis has suggested a solution to handle both concerns.

Two different data representations were generated for exploring different solution approaches to the detection of impulsive noise. The two solutions were

1. Random forest classifier trained on the decibel measurement data.
2. Convolutional neural networks combined with a cross-correlation trained on the spectrogram representation of data (bundled model).

The model evaluations showed that the model based on convolutional neural networks and cross-correlation in combination were the best, achieving useful detecting results.

Furthermore, we arrived at an answer to the initial question: Is it possible to detect and track the origin of impulsive noise events with the help of multiple wireless sound sensors?

One could indeed determine whether PNB's activity, which revolves around shooting events and explosions, caused noise in the nearby settlement with the help of wireless sound sensors. Detecting the origin of impulsive noise events with wireless sensors is possible, but there is a trade-off between cases detected and cases correctly detected.

Bibliography

[1] “The NOISE Observation & Information Service for Europe.” [Online]. Available: https://noise.eea.europa.eu

[2] B. Stephanie, “Loud Noises Aren’t Just Annoying, They’re Bad for Your Health.” [Online]. Available: https://www.healthline.com/health-news/loud-noises-bad-for-your-health

[3] “Measuring sound,” *Published 10 May 2011*. [Online]. Available: https://www.sciencelearn.org.nz/resources/573-measuring-sound

[4] “Noise.” [Online]. Available: https://www.eea.europa.eu/themes/human/noise

[5] T. H. T. V. Panu Maijala, Zhao Shuyang, “Environmental noise monitoring using source classification in sensors,” Tech. Rep. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003682X17307533

[6] J. Nordby, “Sounsensing pictures.”

[7] “What is sound?” [Online]. Available: https://www.fibertex.com/business-areas/acoustics/theory

[8] “Sound.” [Online]. Available: https://courses.lumenlearning.com/physics/chapter/17-1-sound/

[9] C. Andrew, “Periodic Wave.” [Online]. Available: https://www.eeweb.com/periodic-wave/

[10] “Speed of Sound, Frequency, and Wavelength.” [Online]. Available: https://courses.lumenlearning.com/physics/chapter/17-2-speed-of-sound-frequency-and-wavelength/

[11] “Sound Theory.” [Online]. Available: https://acoustics.no/sound-measurement/sound-theory

[12] "The Speed of Sound." [Online]. Available: https://www.physicsclassroom.com/class/sound/Lesson-2/The-Speed-of-Sound

[13] "The Propagation of sound." [Online]. Available: https://pages.jh.edu/virtlab/ray/acoustic.htm

[14] "The Speed of Sound in Other Materials." [Online]. Available: https://www.nde-ed.org/Physics/Sound/speedinmaterials.xhtml

[15] F. Nemazi, "Using cross correlation to classify probability of origin of sound," Tech. Rep.

[16] E. B. editors, "Decibel, unit of measurement." [Online]. Available: https://www.britannica.com/science/decibel

[17] "Comparative Examples of Noise Levels." [Online]. Available: https://www.iacacoustics.com/blog-full/comparative-examples-of-noise-levels.html

[18] "NIGHT NOISE GUIDELINES (NNGL) FOR EUROPE," Tech. Rep. [Online]. Available: https://ec.europa.eu/health/ph_projects/2003/action3/docs/2003_08_frep_en.pdf

[19] R. Brice, "Got to Get You into My Life – Sound recording," Tech. Rep. [Online]. Available: https://www.sciencedirect.com/topics/engineering/digital-audio

[20] "Digital Audio Fundamentals," Jun. 2020. [Online]. Available: https://manual.audacityteam.org/man/digital_audio.html

[21] G. Brown, "Digital Audio Basics: Sample Rate and Bit Depth," Jul. 2019. [Online]. Available: https://www.izotope.com/en/learn/digital-audio-basics-sample-rate-and-bit-depth.html

[22] J. Nordby, "Environmental Sound Classification on Microcontrollers using Convolutional Neural Networks," Ph.D. dissertation, NMBU, 2019. [Online]. Available: https://nmbu.brage.unit.no/nmbu-xmlui/bitstream/handle/11250/2611624/report-print1.pdf?sequence=3&isAllowed=y

[23] "What is a Spectrogram?" [Online]. Available: https://pnsn.org/spectrograms/what-is-a-spectrogram

[24] "https://vibrationresearch.com/blog/what-is-a-spectrogram/." [Online]. Available: https://vibrationresearch.com/blog/what-is-a-spectrogram/

[25] T. Bäckström, "Cepstrum and MFCC." [Online]. Available: https://wiki.aalto.fi/display/ITSP/Cepstrum+and+MFCC

[26] L. P. Jason Wang, "The Effectiveness of Data Augmentation in Image Classification using Deep Learning." [Online]. Available: http://cs231n.stanford.edu/reports/2017/pdfs/300.pdf

[27] Erica, "Introduction to the Fundamentals of Time Series Data and Analysis," Feb. 2020. [Online]. Available: https://www.aptech.com/blog/introduction-to-the-fundamentals-of-time-series-data-and-analysis/

[28] "Cross Correlation Functions and Lagged Regressions." [Online]. Available: https://online.stat.psu.edu/stat510/lesson/8/8.2

[29] a. R. D. B. V. K Vijaya Kumar, Abhijit Mahalanobis, "Correlation Pattern Recognition." Cambridge university, 2010, vol. 1. eidtion.

[30] "Cross correlation." [Online]. Available: https://en.wikipedia.org/wiki/Cross-correlation

[31] P. S. G. Ricardo Queiros, Raul Carneiro Martins, "A new method for high resolution ultrasonic ranging in air," Tech. Rep., Jan. 2006. [Online]. Available: https://www.researchgate.net/publication/239912163_A_new_method_for_high_resolution_ultrasonic_ranging_in_air

[32] "The fourier transform," 2010. [Online]. Available: https://www.thefouriertransform.com

[33] G. Sanderson, "But what is the Fourier Transform? A visual introduction," Jan. 2018. [Online]. Available: https://www.youtube.com/watch?v=spUNpyF58BY

[34] "Cross-Correlation." [Online]. Available: https://www.sciencedirect.com/topics/chemistry/cross-correlation

[35] S. R. . V. Mirjalili, "Python Machine Learning," 2019, p. 2.

[36] M. H. Tesfazion, "A data-driven approach for power loss detection in utility-scale solar power plants," Ph.D. dissertation, NMBU, 2019. [Online]. Available: https://nmbu.brage.unit.no/nmbu-xmlui/bitstream/handle/11250/2612002/A%20data-driven%20approach%20for%20power%20loss%20detection%20in%20utility-scale%20solar%20power%20plants%20Thesis%20MHT%20%28Final%29%20%282%29.pdf?sequence=1&isAllowed=y

[37] S. R. . V. Mirjalili, "Python Machine Learning," 2019, pp. 207–210.

[38] ——, "Python Machine Learning," 2019, p. 92.

[39] ——, "Python Machine Learning," 2019, p. 100.

[40] "sklearn metrics - recall score." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html

[41] "sklearn metrics - precision score." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html

[42] "sklearn metrics - f1 score." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

[43] S. Singh, "Understanding the Bias-Variance Tradeoff," May 2018. [Online]. Available: https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229

[44] S. R. . V. Mirjalili, "Python Machine Learning," 2019, p. 76.

[45] A. Nagpal, "Over-fitting and Regularization." [Online]. Available: https://towardsdatascience.com/over-fitting-and-regularization-64d16100f45c

[46] "Difference between model parameters vs hyperparameters." [Online]. Available: https://www.geeksforgeeks.org/difference-between-model-parameters-vs-hyperparameters/

[47] G. Bland, "Train/Test Split and Cross Validation – A Python Tutorial," Oct. 2020. [Online]. Available: https://algotrading101.com/learn/train-test-split/

[48] F. Chollet, "Deep Learning with Python," 2018, p. 1.

[49] "Multi-Layer Neural Network." [Online]. Available: http://deeplearning.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/

[50] T. Wood, "What is the Sigmoid Function?" [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/sigmoid-function

[51] "Role derivative of sigmoid function in neural networks," 2018. [Online]. Available: https://datascience.stackexchange.com/questions/30676/role-derivative-of-sigmoid-function-in-neural-networks

[52] S. R. . V. Mirjalili, "Python Machine Learning," 2019, p. 416.

[53] Y. M. Moe, "Deep learning for automatic delineation of tumours from PET/CT images," Ph.D. dissertation, NMBU, 2019. [Online]. Available: https://nmbu.brage.unit.no/nmbu-xmlui/bitstream/handle/11250/2597305/Yngve_Mardal_Moe_Masteroppgave.pdf?sequence=1&isAllowed=y

[54] F. Chollet, "Deep Learning with Python," 2018, p. 11.

[55] A. Agrawal, "Loss Functions and Optimization Algorithms. Demys-

tified." [Online]. Available: https://medium.com/data-science-group-iitr/loss-functions-and-optimization-algorithms-demystified-bb92daff331c

[56] F. Chollet, "Deep Learning with Python," 2018, p. 60.

[57] D. Godoy, "Understanding binary cross-entropy / log loss: a visual explanation," Nov. 2018. [Online]. Available: https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a

[58] M. Stewart, "Neural Network Optimization," Jun. 2019. [Online]. Available: https://towardsdatascience.com/neural-network-optimization-7ca72d4db3e0

[59] S. R. . V. Mirjalili, "Python Machine Learning," 2019, pp. 36,37,38.

[60] I. Dabbura, "Gradient Descent Algorithm and Its Variants." [Online]. Available: https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

[61] D. Tavakoli, "Autonomous Drone Landing using Deep Reinforcement Learning," Ph.D. dissertation, NTNU, Jun. 2020.

[62] F. Chollet, "Deep Learning with Python," 2018, p. 10.

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

[64] S. R. . V. Mirjalili, "Python Machine Learning," 2019, pp. 519–524.

[65] P. A. A. C. a. C. L. Felix Gontier, Mathieu Lagrange, "An Efficient Audio Coding Scheme for Quantitative and Qualitative Large Scale Acoustic Monitoring Using the Sensor Grid Approach," Ph.D. dissertation, Nov. 2017.

[66] F. Chollet, "Deep Learning with Python," 2018, p. 18.

[67] Q. Kong, M. Yin Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition."

[68] J. P. B. Justin Salamon, "Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification," Nov. 2016.