Delft University of Technology
Master of Science Thesis in Embedded Systems

Gunshot Detection in Wildlife using Deep Learning

Deniz Danaei

TU Delft Delft University of Technology

Gunshot Detection in Wildlife using Deep Learning

Master of Science Thesis in Embedded Systems

Embedded and Networked Systems Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Deniz Danaei
D.Danaie@student.tudelft.nl
denizdanaie@gmail.com

17/12/2021

Author
Deniz Danaei (D.Danaie@student.tudelft.nl)
(denizdanaie@gmail.com)
Title
Gunshot Detection in Wildlife using Deep Learning
MSc Presentation Date
23/12/2021

Graduation Committee

Marco Zuniga	Delft University of Technology
Jie Yang	Delft University of Technology
Arjan Smith	Alten group
Babak Ghafary	Alten group

Abstract

The fight against the illegal hunting of African wildlife is a never-ending process. In order to preserve animal habitats and save them from extinction, many national parks utilize surveilling solutions to prevent, detect and locate intruders. One strategy to detect and locate the illegal hunters or so-called *poachers* is to detect and locate the gunshot sounds using an acoustic surveillance system consisting of embedded devices scattered within the park. The embedded devices–so-called end-nodes surveil the environment continuously, processing the sound events gathered and converted to audio by the acoustic sensors. Then, using a deep learning algorithm, any sound event classified as a gunshot is reported to the authorities.

This research study proposes a deep learning model for gunshot sound recognition in African wildlife. It also investigates a potential correlation between gunshot sound recognition accuracy, signal-to-noise ratio (SNR), and shooter distance.

To this end, gunshot and ambient sounds such as Savanna wildlife, rain, and thunder were collected and synthesized to simulate different scenarios. Various experiments were conducted using this data to investigate the influence of different parameters on gunshot recognition accuracy.

Our analysis revealed the negative effect of the weather conditions, such as rain and thunderstorms, on the model accuracy. The obtained results also showed a positive correlation between the gunshot recognition accuracy and SNR. Since SNR is negatively correlated to the shooter distance when both noise and signal levels are constant, we have proved that the gunshot recognition accuracy also negatively correlates with the distance.

Finally, a single CNN (convolutional neural network) model is proposed for gunshot sound recognition in African wildlife. The model performs acceptably in three different weather conditions. The gunshot recognition accuracy for five shooting ranges is also provided based on the uncovered correlation.

"For instance, on the planet Earth, man had always assumed that he was more intelligent than dolphins because he had achieved so much—the wheel, New York, wars and so on—whilst all the dolphins had ever done was muck about in the water having a good time. But conversely, the dolphins had always believed that they were far more intelligent than man—for precisely the same reasons."

– Douglas Adams, The Hitchhiker's Guide to the Galaxy

Preface

Surveillance systems have been one of the most researched topics within human history. From building watchtowers to installing wireless sensor networks within a property to automated machine learning techniques, many solutions have been introduced to the problem. Surveillance systems are primarily used to detect and prevent intrusion or illegal activity. Advance in technology has opened many possibilities in this field. The Internet of Things (IoT) and wireless technologies allow us to detect and alert unwanted activities almost instantly.

The Chengeta project is aimed at exploiting the newest advances in IoTs to get closer to preserving wildlife and stopping illegal hunting. I was fascinated when this project was pitched to me, and I knew at the moment that this is what I would like to work on for my master's thesis. I believe that it is our responsibility as humans to do our part to save the life on land. This project is my opportunity to have helped me reach this goal.

Many people have helped me during this thesis. Firstly, I would like to thank Arjan and Babak for their enthusiasm, support, and guidance during my thesis. Secondly, I would like to thank Marco for all his help and time. Thirdly, I would like to thank Jie for the insight and feedback he provided. Additionally, I would like to thank the Alten group for offering me the opportunity to work on this project and always providing help when needed. Finally, I would like to thank my family for all their love and support, without whom I would have never reached this point in my life and career.

Deniz Danaei

Delft, The Netherlands
December 17, 2021

Contents

Chapter 1

Introduction

The hunting of wildlife by humans is a long-standing practice in many communities, and it continues in numerous forms throughout the world. In order to preserve animal habitats and save them from extinction, hunting has been declared illegal in many countries and wildlife sanctuaries over the world. The surveillance of illegal hunting or so-called *poaching* is challenging since the authorities are forced to control vast territories spread over hundreds of kilometers, often without any infrastructure available and sometimes under complex geological conditions [1]. Even though international organizations have offered solutions such as banning the trade of animal products and establishing local conservancy committees, poaching is still a key concern for many wildlife sanctuaries [1]. Consequently, many national parks have had to look for active solutions to prevent, detect and locate intruders. Various manual surveillance techniques have been implemented, such as employing park rangers and installing gates and barriers. As technology advances, many novel intelligence-driven solutions are offered to the poaching problem, such as tracking the animals, motion and vibration detectors, and making the protected territory *smart* [2].

Intelligence-driven solutions for anti-poaching are usually based on IoT [3, 4, 2]. Internet of Things (IoT) is the physical objects embedded with (wireless) sensors that have processing ability and are able to connect and exchange data over the Internet or other communications networks [5]. The concept of these solutions is to detect unusual activities with sensors and alert the authorities using their communication system. The utilized sensors can be motion and vibration detectors, RFID tags and ultrasonic technology, video cameras, or acoustic sensors such as microphones. Motion sensors require a clear line of sight in order to be operatable [6] and therefore are not a viable solution in regards to denser outdoor areas. Video cameras can gather valuable information and, combined with current image processing methods, could provide accurate results for poaching prevention; however, they are costly to implement in vast open areas and consume much power. Using microphones and acoustic surveillance can be a viable solution for the cost-accuracy trade-off [7, 8, 9].

Chengeta Wildlife [10] is an organization that directly supports anti-poaching efforts on the ground in Africa. One solution they seek is to detect and pinpoint gunfire within the national parks. Such a system would allow the rangers to

chase the poachers more efficiently, as their general location would be revealed. This project is dedicated to providing Chengeta with an anti-poaching system based on acoustic surveillance to detect gunshot sounds in wildlife.

The desired system consists of a base station and embedded devices–so-called *end-nodes* scattered within the park that surveil the environment continually. End-nodes process the sound events gathered and converted to audio by the acoustic sensors. Then, using a deep learning algorithm, any sound event classified as a gunshot is reported to the base station. Subsequently, the location of the sound source will be estimated in the base station with the information gathered from the nodes using localization techniques. The project is further split into the gunshot sound recognition system, embedded device design, communication system design, and locating the sound source.

This graduation thesis is dedicated to building the gunshot sound recognition system utilizing a deep learning algorithm. The process involves preparing a dataset and creating and improving a CNN model that recognizes and classifies gunshot sounds in the African Savanna. To estimate the detection range of the end-nodes, this study also aims to research and investigate a potential correlation between the shooter distance and the sound recognition accuracy.

1.1 Problem Statement

Gunshot detection in wildlife can be considered a sub-field of surveillance systems that have been investigated extensively in the past. With the renaissance that deep learning has brought in many areas of signal processing, surveillance systems have also been updated to use deep learning as a state-of-the-art solution. From the use of image processing and ultra-wideband (UWB) signal classification [11] in intrusion detection systems to avoiding Human-animal conflict [4, 3], and to scream and gunshot detection systems [12], deep learning methods have been a reliable way of implementing innovations in surveillance.

The problem of detecting gunshot sound in *wildlife* however, is not fully investigated yet. There are solutions offered by Ghiurcau Et al. in their works of *TESPAR* [13, 14, 15] to classify humans, birds, and cars using acoustic sensors. There are studies regarding gunshot detection systems in urban areas and noisy environments [12, 16, 17, 18]. However, none of the mentioned studies provide a solution for gunshot detection in wildlife using deep learning.

This thesis aims to: *provide a deep learning model for gunshot sound detection in African savanna and investigate the model accuracy range when deployed on the end-nodes.*

1.2 Document Structure

In Chapter 2, related works relevant to this thesis are presented. Chapter 3 then explains the basic concepts in signal processing and deep learning used in this thesis. It also provides information regarding gunshot acoustics. Subsequently, Chapter 4 describes the methods and tools used during the experiments, followed

by Chapter 5 presenting the results. Finally, Chapter 6 discusses the obtained results along with ideas for possible future research.

Chapter 2

Related work

As stated in Chapter 1, gunshot detection in wildlife can be considered a subfield of surveillance systems and intrusion detection systems. Both fields have been investigated extensively through the years. In this Chapter, related works to this research study are presented in three separate Sections. Section 2.1 reviews the intrusion detection systems similar to this research study. Followed by Section 2.2 surveying gunshot acoustic detection systems. Finally, Section 2.3 browses through challenges involved in applying deep learning methods in audio processing.

2.1 Intrusion Detection Systems

The intrusion detection applications vary from guarding private property to protecting a natural reserve and avoiding the human-wildlife conflict. However, the core concept remains the same, and they all utilize sensor networks to detect unwanted activities. For instance, in [4] authors have proposed "*an automated wild animal detection and repellent system*" that uses motion sensors and cameras to alert if wild animals are likely to enter a specified area. First, motion sensors are placed to sense the intrusion activity, and if there is any, the camera is initiated to record and live stream the event. A *YOLOv3* model is then used to process the received image from the camera to confirm if the detected intrusion is, in fact, accurate. The system then repels the animal back to the forest by playing a buzzer sound and flashlights. Similarly, in [3], authors have proposed a system that uses motion sensors and cameras to detect animal intrusion and communicate to a central station using *LoRaWAN* technology.

In [11] however, the deployed intrusion detection system, is based on UWB technology. The wireless sensor network placed in the area analyzes the characteristics of UWB signals upon object entry, and by extracting the signal features using a convolutional neural network(CNN), the object is classified as a human or an animal.

From intrusion detection systems proposed, the closest to this research study is the work of Ghiurcau Et al. [13, 14, 15] which aim to classify the sounds that

originate from humans, birds, and cars. The proposed solution is based on sound classification and TESPAR algorithm–a language made for describing complex waveforms in digital terms [19]. Ghiurcau Et al. have created a 3-class database and have added several types of noise (white Gaussian noise, rain sound, and wind sound) to simulate different outdoor environments.

Introducing environmental sounds to the dataset used in [14] is proven helpful to simulate more realistic and general data. Such as many other related works mentioned in the following Section, our work is also motivated to utilize the environmental sounds.
On the other hand, [14] has incompatibilities to our case, such as the lack of the gunshot sounds within the experimental data. Additionally, the classification method is desired to be a deep learning model in this study. However, in [14] sound classifications are directly done on the binary matrices resulted from zero-crossing points in the audio signals [14].

2.2 Gunshot Acoustic Detection

The problem of detecting the gunshot acoustic and locating the shooter has been studied extensively for the purpose of military or civil applications. The earlier works on shooter localization are directly based on analyzing the gunshot acoustics. For instance, in [20], authors have proposed a solution for shooter localization based on the time difference of acoustic muzzle blast (MB) from the gunfire and the ballistic shock wave (SW) from the bullet at each sensor (microphones). The shooting direction is also revealed based on SW detection time. Their solution, however, is assumed to be on a perfectly synchronized network and highly accurate microphones. This method may not be applicable for civil projects due to the extensive costs. Chapter 3.3 looks closely to the characteristics of gunshot acoustic, and the effects of the recordings and environmental factors.

Another similar work in this topic is the gunshot and scream detection system proposed in [12]. The system classifies the scream or gunshot event from ambient noise. The authors have focused on a thorough comparison of feature extraction methods to reduce the dimensionality of the problem and have proposed a feature vector that gives acceptable results at a more negligible computational cost. They obtained an accuracy of 90% and a false rejection rate of 8% using two Gaussian mixture models (GMM) in parallel to discriminate between screams and noise, as well as gunshots and noise. One of the experiments they carried out is to investigate the effects of the noise level on the classification accuracy by adding noise to the audio events, changing the Signal-to-Noise ratio from 0 to 20dB, with a 5dB step. Their results have verified a performance degradation as the SNR decreases.

Another work on gunshot detection is presented in [17]. Authors have compared the classification accuracy of gunshots in different settings of clean, mildly noisy, and heavily noisy environments. Their database contains example sounds of handclapping, balloons explosions, rifle and pistol shots, and speech. They have compared different feature extraction methods and used a Hidden Markov model (HMM) for classification. The novelty of the proposed method is that

before classifying the audio event, it is compared to a gunshot template, and then a threshold value is used to determine whether it is a gunshot. The authors claim to have achieved a computationally cheap procedure and comparable performance to algorithms adapted from speech processing, especially in noisy environments and in impulsive sound classification tasks.

From the subject of template matching, another novelty for gunshot detection is presented in [21]. Ahmed et al. have proposed a system that is impervious to noise with low computational complexity. It consists of a two-step approach, an impulsive event detection framework followed by a relatively complex gunshot recognition stage. In the second stage, a template matching measurement is used to train a Support Vector Machine (SVM) classifier. The database used for the study consist of G3 and MP5 gunshot sounds acquired for shooter distances of 100, 200, and 300 meters. Also, ambient sounds like claps, door slams, and people talking are used as outsider signals for the gunshot recognition system. The authors have also mapped their model accuracy to three SNR thresholds. The false alarm rate for a minimum of 10 dB is 5%, which drops to 2% for an SNR value of 25dB.

In contrast to the systems reviewed above, recent works on gunshot detection utilize deep learning methods such as Artificial and Convolutional Neural Networks (ANN, CNN, respectively). For example, in [22], authors have compared the model accuracy of the ANN model to SVM. The main objective of their study is to present a gunshot detection system that can be used on a smartphone for patrolling soldiers. The system classifies the captured sounds into one of six different kinds of gunfire, AK-47, AR15, .38 Caliber, Shotgun, 9-mm., and .45 Caliber. They have also added different noise levels to compare two model accuracy for different SNRs. The nature of the injected noise, however, is unclear. Their results show a drastic improvement for lower SNRs when using ANN.

In [18], Bajzik et al. have investigated the usage of different image processing methods in the field of audio event recognition. Several convolutional models and the effect of signal downsampling are analyzed on gunshot sounds mixed with background sounds such as traffic noise, human voice, and other forms of environmental sounds. Authors have also visualized different two-dimensional representations of the audio signals, such as spectrograms, MFCCs, and similarity matrix, and used the combination of them to train the CNN network.

Finally, the closest work to this project is [16]. Authors have proposed a low-cost and accurate gunshot detection system that autonomously identifies and alerts authorities of gunfire occurrences in a city. They have collected sound clips from online audio databases as well as clips recorded in residential areas and at a gun range. One and two-dimensional CNNs were then trained on the sound data and spectrograms to recognize gunshots. The model was also deployed on an embedded device to test at a gun range. The authors state that the model performance began to diminish when the device was 225 meters away from the site of gunfire. When stationed more than 300 meters away, it could no longer recognize any samples as true positive. Even though the papers reviewed in this Section have offered various solutions for gunshot detection, each has a certain incompatibility to the case study investigated in this thesis. For instance, none of the gunshot detection systems proposed have included

sounds for wildlife simulations. Moreover, only one of the proposed solutions has considered the recognition model to be deployed on an embedded device [16]. This solution, however, is only accurate for a radius of 300 meters, and it would be costly to implement it in a park that expands over thousands of acres since so many nodes will be required.

It is worth mentioning that some of the design choices and methods used in the mentioned works are borrowed in this research study. For instance, instead of exploring classical machine learning models, CNN models are chosen as the state-of-the-art solutions. Additionally, the idea of altering the noise level and simulating different SNR values [12, 21] is also done in this research study.

Finally, One novelty introduced in this research study is mapping the shooter distance to SNR and model accuracy range. Since the gunshot recognition model is aimed to be deployed on the end-nodes, it is desired to investigate the model accuracy in different ranges.

2.3 Deep Learning Challenges for Audio Processing

Even though most of the reviewed works utilize deep learning methods, there are still challenges to overcome. It is true that deep learning has outperformed traditional audio processing methods, which as a result, has enabled practical applications on a larger scale such as speech processing, music, and environmental sound recognition [23]. However, deep learning is known to be most beneficial when applied to large training datasets. Unlike the availability of ImageNet for the breakthrough of deep learning in computer vision, there is no such a well-labeled dataset that can be shared across domains, including speech, music, and environmental sounds [23].

Another important difference between image and audio processing is feature extraction methods. For instance, raw audio samples form a one-dimensional time-series signal which is fundamentally different from two-dimensional images. Audio signals are commonly transformed into two-dimensional time-frequency representations for processing. However, the two axes, time and frequency, are not homogeneous as horizontal and vertical axes in an image. Furthermore, images are instantaneous snapshots of a target and are often analyzed as a whole or in patches with little order constraints; however, audio signals have to be studied sequentially in chronological order. These properties gave rise to audio-specific solutions [23].

From the practical applications of deep learning in audio processing, acoustic gunshot detection can fit the environmental sound recognition (ESR) branch. ESR is one of the challenging subjects due to the noisy nature of the signals and the small quantity of labeled data that is readily available. Even though the performance of core techniques in this field has been steadily improving, their effectiveness still tends to be degraded under very noisy conditions. For instance, one of the various issues that need to be addressed for the practical use of ESR technology is to improve sound-source separation performance. One promising approach is the development of more powerful modeling techniques

such as those based on deep learning [24].

On the other hand, The well-known publicly available datasets for ESR such as AudioSet [25] and UrbanSound8k [26] are not necessarily suited for every project. This is because environmental sounds are very diverse and case dependant. A solution introduced to this problem is *data generation* and *data augmentation*. For example, according to [27] for environmental sounds, linearly combining training examples along with their labels improves generalization. Likewise, for source separation, models can be trained successfully using synthesized datasets by mixing separated tracks.

Having stated the related works and the challenges above, the next Chapter looks closer to the basic concepts one should be familiar with to grasp the work done on this study thoroughly. These concepts are then used to design experiments in Chapter 4.

Chapter 3

Background

This chapter provides background information regarding the topics covered in this research study. Section 3.1 provides general information about signal and audio processing. Subsequently, Section 3.2 demonstrates how shooter distance can be mapped to different SNRs, and how the gunshot recognition accuracy might link to the distance. Next, Section 3.3 discusses gunshot sound characteristics. Section 3.4 looks through the topics of interest in deep learning.

3.1 Audio Signal Processing

Audio signal processing is a subfield of signal processing concerned with the electronic manipulation of audio signals. Audio signals are electronic representations of sound waves. Before looking deeper into audio signal processing, the key difference between sound and audio is worth mentioning. Sound is mechanical wave energy, whereas audio is made of electrical energy that represents sound electrically [28].

Sound, in physics, is produced in the form of a pressure wave consisting of compressions and rarefactions traveling through a medium [29]. It is made by a vibrating object causing the surrounding air molecules to vibrate [30]. Sound waves are generally made of many overlapping frequencies that, when combined, yield the character of the sound itself. As humans, we are generally born to hear within the frequency range of 20 Hz to 20 kHz (audible sound). Frequencies below 20 Hz (Infrasound) and above 20 kHz (Ultrasound) are inaudible to humans. Four major sound properties are used to differentiate sounds: *pitch*, a method for organizing sounds based on a frequency-based scale, *amplitude* (loudness or softness) determining the relative loudness of a sound wave, *duration*, and *timbre*. Timbre is the quality of a musical note and is related to the presence of harmonics of a primary frequency.

Section 3.1.1 looks into noteworthy audio features, and Section 3.1.2 explains the concept of signal-to-noise ratio.

3.1.1 Audio Features

As stated in Section 3.1, audio signals are electronic representations of sound waves. Audio can be further differentiated into two categories of analog and digital. As with most engineering topics, each domain has pros and cons. The specific application, performance requirements, transmission medium, and operating environment can determine whether analog or digital signaling (or a combination) should be used [31].

The process of converting an analog into a digital signal is called *sampling*. This process is done by an Analog-Digital Converter (ADC) converter. The resolution of the ADC, or the *sampling rate*, determines the quality of the digital sound. A commonly seen unit of sampling rate is Hertz (Hz), which indicates "*samples per second*". The most common audio sampling rates that cover the entire range of human hearing are 44.1 kHz, 48 kHz, 88.2 kHz, or 96 kHz [32].

Performing an in-depth analysis of an audio signal requires the extraction of the appropriate parameters of the audio signal. This process is called *audio feature extraction* [33]. The most important audio features can be categorized into time, frequency, and time-frequency domains. The following sections describe some of the well-known techniques reported in each domain.

Time Domain

Possibly the most significant trait of time-domain features is that they do not require applying any change or transformation on the original audio signal. Thus, any computation could be performed directly on the samples of the signal itself [33]. Figure 3.3 presents a time-domain representation of an audio signal.

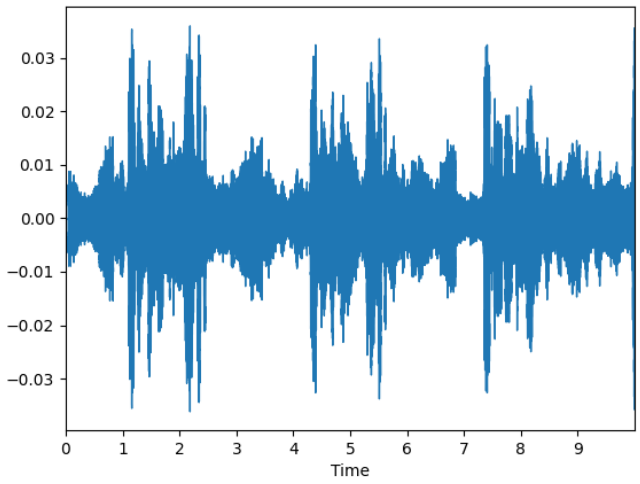


Figure 3.1 – Time domain presentation of an audio signal from [34]. In time domain, the signal can be presented with a one dimensional vector of amplitudes in time units (μs, ms or second).

It is worth mentioning again that sound is a *Longitudinal* wave, meaning it is produced in the form of a pressure wave consisting of compressions and rarefactions [35]. The concept comes into the picture where compression is a period of higher pressure than the ambient pressure (for instance, when the output is 0), and rarefaction is a period where the pressure is lower than the ambient pressure. One type of representing the longitudinal waves is to call the ambient pressure zero, model the compressions as positive and the rarefactions as negative values of the amplitude [35].

Some of the notable time domain features are *Zero-Crossing Rate* (ZCR), *Amplitude Envelope* (AE), and *Short-time energy* (STE). ZCR defines the number of times the audio signal waveform crosses the zero amplitude level, which provides a rough estimator of the dominant frequency component of the signal. AE aims to extract the maximum amplitude within each time frame, which can be used for detecting the beginning of a sound. STE can be defined as the average energy per signal frame, which could be utilized in detecting the transition from unvoiced to voiced speech.

Frequency Domain Features

Any audio recording can be decomposed into multiple signals with different frequencies and magnitudes. Since the recordings are within the time domain, they need to be converted to their representation in the frequency domain to extract the underlying frequencies. Such conversion is done by performing a *Fast Fourier Transform* (FFT) calculation. Since digital signals use discrete values, a *Discrete Fourier Transform* (DFT) calculation must be performed to convert it to the frequency domain. Figure 3.2 shows the frequency domain representation of the signal from Figure 3.1. This Figure depicts the existing frequencies within the audio signal and their magnitude. Some of the notable frequency domain features are *Band Energy Ratio* (BER), *Spectral Flux* (SF), and *Spectral Centroid* (SC). BER provides the relation between the lower and higher frequency bands, which is mostly used in music/speech discrimination and music classification [36]. SF describes sudden changes in the frequency energy distribution of sounds, which can be applied to detect significant changes in the spectral distribution. Finally, SC describes the center of gravity of spectral energy. In other words, it gives the frequency band where most of the energy is concentrated.

Time-frequency Representation

These features combine both the time and frequency components of the audio signal. The time-frequency representation is obtained by applying the *Short-Time Fourier Transform* (STFT) on the time domain waveform. For example, consider an audio recording split into frames of predefined interval size. STFT is when Fourier Transform is performed on each frame separately. In contrast with the Fourier Transform calculation performed on the entire recording, STFT reveals the frequencies that appear at different moments in time. Some examples of this feature domain are Spectrogram, Mel-spectrogram, and MFCC.

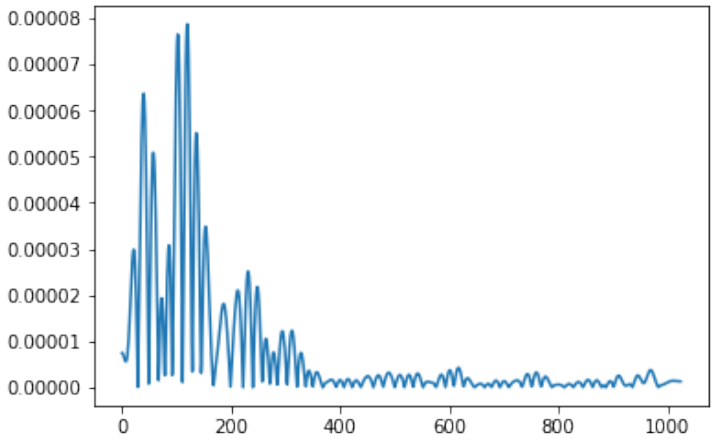


Figure 3.2 – Frequency domain presentations of an audio signal. The y axis presents the magnitude of the frequencies (x axis) within the audio signal. Figures are from Article [34]

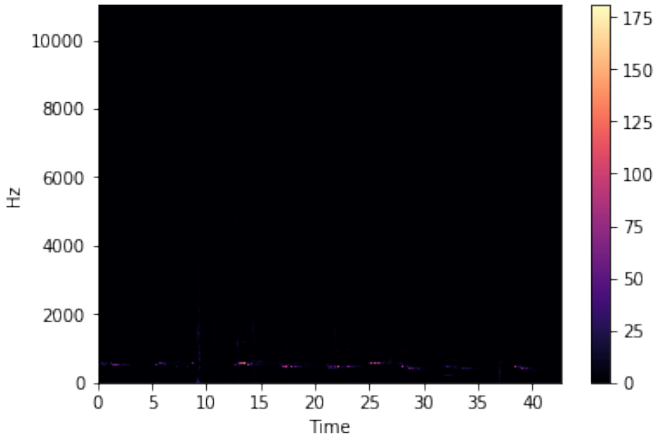
- **Spectrogram** is a visual way of representing the signal power or signal "*loudness*" overtime at various frequencies present in a particular waveform. This signal representation allows one to examine the power level difference at different frequencies and how these levels fluctuate over time for each chosen frequency. Figure 3.3a shows a spectrogram on a linear scale, and Figure 3.3b present it on log scale.

- **Mel-Spectrogram** is a spectrogram where the frequencies are converted to the Mel scale. Humans do not perceive frequencies on a linear scale. They are better at detecting differences in lower frequencies than higher frequencies. For example, they can easily tell the difference between 500 and 1000 Hz but will hardly differentiate between 10 kHz and 10.5 kHz, even though the distance between the two pairs are the same [38]. The *Mel* scale maps frequencies to equally spaced pitches or Mels. Conversion from frequency (f) to Mel scale (m) is given by Equation 3.1.

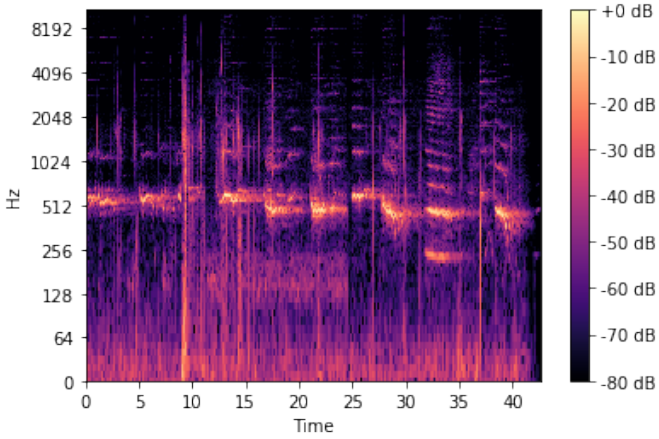
$$m = 2595.\log\left(1 + \frac{f}{700}\right) \tag{3.1}$$

  While linear audio spectrograms are ideal for applications where all frequencies have equal importance, Mel-spectrograms are better suited for audio classification applications and applications that need to model human hearing perception. Figure 3.4a displays the Mel-Spectrogram of three audio event.

- **Mel Frequency Cepstral Coefficients** (MFCCs) are coefficients that collectively make up an MFC (*mel-frequency cepstrum*). The information of the rate of change in spectral bands of a signal is given by its *cepstrum* [39]. A cepstrum is a spectrum of the log of the spectrum of the time signal. Mel-frequency cepstrum utilizes the Mel scale for frequency band spacing which approximates the response of the human auditory system. Whereas



(a) Linear scale



(b) Log scale

Figure 3.3 – Time-frequency domain presentation of the audio signal from Figure 3.1. Figures are borrowed from [34]

the normal spectrum uses linear spacing for frequency bands. Figure 3.4 shows the Mel-spectrograms and MFCCs of three audio signals.

The MFCC feature extraction technique includes windowing the signal, applying the DFT, taking the log of the magnitude, and then warping the frequencies on a Mel scale, followed by applying the inverse DCT.

MFCCs use a reduced set of frequency bands; however, this process removes information and destroys spatial relations [23]. Depending on the task, it might be feasible to use log-mel spectrums. Nevertheless, most of the related works presented in this study use MFCCs for acoustic feature representation of the gunshot audio, showing acceptable results. Additionally, MFCCs yield lower computation time since the number of parameters for training the deep learning model is lower than Mel spectrograms. If used in the end-nodes, calculating the MFCCs and classifying the sound events would take less time than using Mel

spectrograms.

3.1.2 Signal-to-Noise Ratio

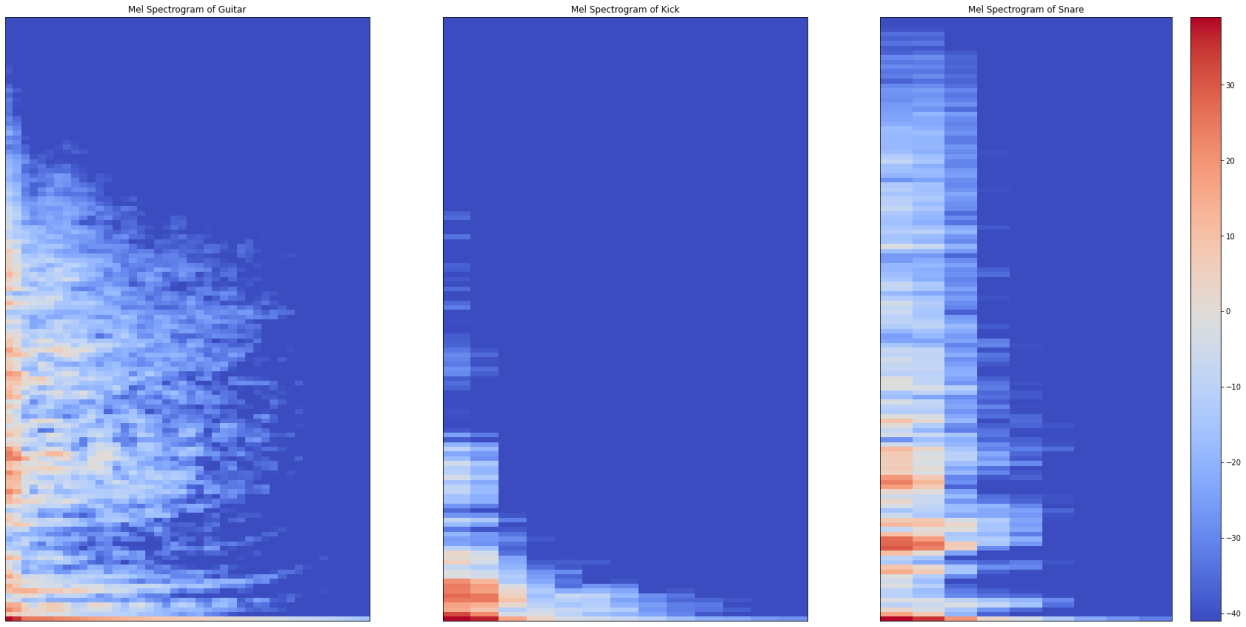
Signal-to-noise ratio (SNR) means the dimensionless ratio of the signal power to the noise power contained in a recording [40]. In audio processing, SNR is a measure to compare the level of the desired signal to the level of background noise and is often expressed in decibels. Higher ratios than 1:1 (greater than 0 dB) indicate higher signal levels than noise [41]. Based upon the definition of decibel, if signal and noise are expressed in decibels (dB) as:

$$P_{signal,dB} = 10\log_{10}(P_{signal}) \tag{3.2}$$

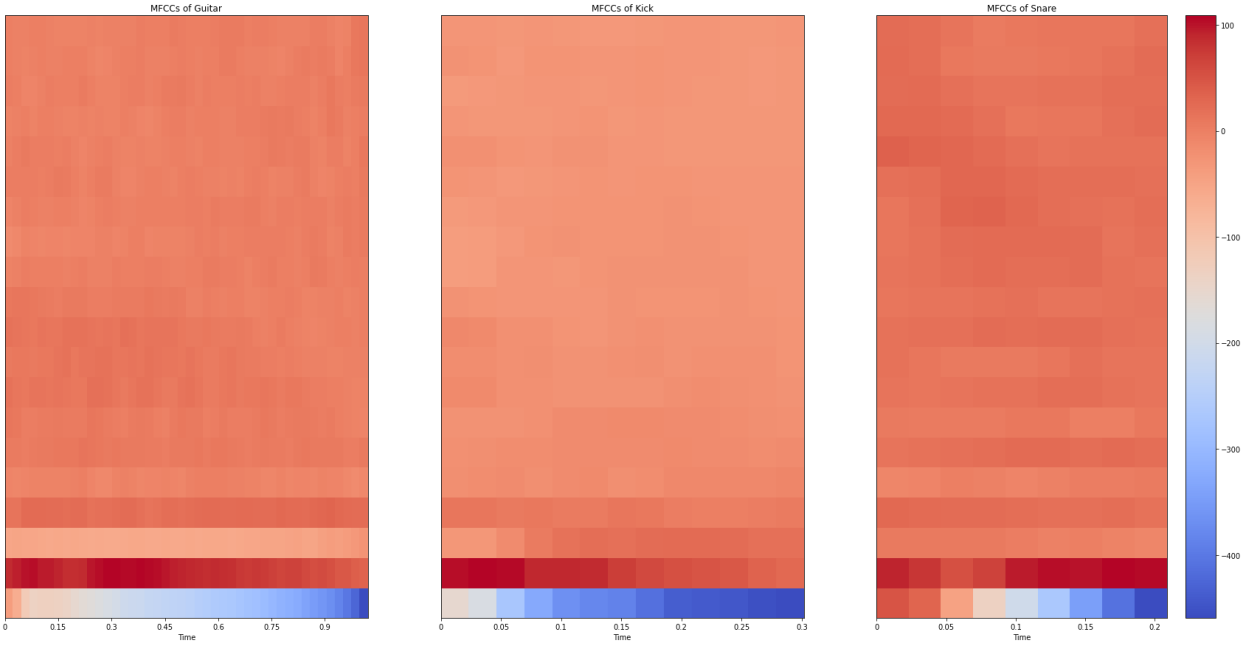
$$P_{noise,dB} = 10\log_{10}(P_{noise}) \tag{3.3}$$

The SNR can be calculated with the equation below:

$$SNR = P_{signal,dB} - P_{noise,dB} \tag{3.4}$$



(a)



(b)

Figure 3.4 – (a) Mel spectrograms and (b) MFCCs of audio signals from a guitar, kick and snare. Figures are borrowed from [37].

3.2 Mapping SNR to Distance

This section investigates the correlation between sound pressure level and the distance of the sound source from the receiver. The first step of this investigation is to look into the sound pressure levels of gunshots and the possible sound sources within the savanna wildlife. The final purpose of this section is to link the SNR to distance. If there is a correlation between the gunshot recognition accuracy and SNR, we can map it to the shooter's distance from the receiver (microphones). Equation 3.5 presents this hypothesis.

$$(distance \propto SNR) \wedge (SNR \overset{?}{\propto} Accuracy) \Rightarrow distance \overset{?}{\propto} Accuracy \tag{3.5}$$

According to [42, 43, 44], average unsuppressed gunshot level is around 150dB close to the gun. Following the inverse-square law for acoustics, the sound pressure of a spherical wavefront radiating from a point source decreases by 50% as the distance is doubled; measured in dB, the decrease is 6.02 dB [45]. In [44], the authors present the result of experimental shootings done to measure the noise level of the firearms. Conducted experiments confirm the reduction in sound level for each doubling of distance, which follows the inverse square law as expected.

The sound level of Savanna wildlife is assumed to be at an average value of 60dB. This assumption is based on the gathered information presented the Table 3.1. African wildlife has some of the loudest animal sounds in the world [46], which mostly happens during the mating season. Since the louder sounds are less frequent than the sound of vegetation, bugs, and birds, the savanna sound level for this project is assumed to be 60 dB on average throughout the experiments. However, it should be noted that the momentary values differ from one point in time to another.

Noise source	**Sound Pressure Level (dB)**
Thunderclap	120
Hyena, Hippo, Lion, African Elephant	112
African Cicada	107
Noise from trees wind speed 8m/s	60
Moderate rainfall	50
Bird calls (44 dB); lowest limit of urban ambient sound	40
Quiet rural area	30
Light rain	30
Quiet forest wind speed 1 m/s	20
Buzzing fly	4

Table 3.1 – Examples of different sound levels [47, 48, 46]

Except for thunderstorms, rainfall sound level typically measures around $20 \sim 30$ dB [49], which can increase to 60dB during heavy rainfall. However, this is dependent mainly on the surface that the rainfall lands. For example, rainfall on metal roofs is much louder than on soil.

A clap of thunder typically registers about 120 dB near the groundstroke.

Nevertheless, the sound level picked by the microphone depends on the distance to the sound source.

Taking into consideration the facts presented above and the SNR definition on dB scale explained in section 3.1.2, Table 3.2 presents an approximation of gunshot sound pressure at different distances to the receiver. The SNR values are calculated based on the assumption that the background noise is constant at 60 dB. It is worth emphasizing that the mappings are only an approximation, and the precise values depend on many parameters such as atmospheric density, humidity, temperature, surroundings, etc.

Shooter distance	**Sound Level*** (SPL_{dB})	SNR_{dB}
1m	150	90
10m	125	60
100m	105	45
1km	85	25
10km	60	0

Table 3.2 – Distance-related decrease of sound level for gunshot [50]. This table is calculated based on the assumption that the background noise is constant from microphone's point of view. *Sound levels are calculated related to the gunshot sound level being 150dB on average within one meter of a gunfire.

Since the signal-to-noise ratios are investigated on a linear scale in this study 4.1.2, Table 3.3 presents the linear SNR, its relative dB changes, and its mapping to the shooter distance. For instance, assuming that the background noise level (Savanna ambient sounds) is 60 dB if the linear SNR and its relative SPL (dB) change are 2:1 and 6 dB, respectively, the shooter distance from the acoustic sensor would be around 5 km.

Linear Ratio (X:1)	**dB Change in Root Power Quantity Sound level** (SPL_{dB})	**Transmitter-Receiver Shooter Distance (km)**
1:1	0 dB	>10
2:1	6 dB	4.7
3:1	10 dB	3
4:1	12 dB	2.4
5:1	14 dB	1.9

Table 3.3 – Linear ratio of SNR and its relative SPL_{dB} change [51]. The background noise is assumed to be 60 dB, and the gunshot sound level is assumed to be 150 dB near gun. The distances are calculated based on Table 3.2.

3.3 Gunshot Audio Signal

Assessing and evaluating gunshot acoustics requires a thorough understanding of the gunshot sound characteristics. Firearm sound is made up of one or more

discrete acoustic events. However, the presence of these events depends on the type of the gun, ammunition, microphone position, and even the surrounding environment. In the following Section 3.3.1, some of the main characteristics of the gunshot sound is addressed. Subsequently, the practical and theoretical issues encountered in gunshot sound analysis is presented in Section 3.3.2.

3.3.1 Gunshot Acoustics

Two primary attributes characterize gunfire and enable the detection of gunfire discharges; the *Muzzle Blast* and *Supersonic Projectile*. Once the gunpowder combustion is complete, the firearm itself may produce much more subtle mechanical sounds. These sounds can only be detected if the microphone is sufficiently close to the firearm to pick up the tell-tale sonic information [52].

- **Muzzle Blast**: A conventional firearm uses confined combustion of gunpowder to propel the bullet out of the gun barrel. Even though the sound of the explosion emits from the gun in all directions, the majority of the acoustic energy expels in the direction the gun barrel is pointing. Muzzle Blast is the explosive shock wave and sound energy emanating from the barrel, which typically lasts for less than three milliseconds [53].

- **Supersonic Projectile**: If the bullet travels at supersonic speed, the second source of acoustic gunshot information is present. The supersonic projectile's passage through the air emits an acoustic shock wave, which expands in conic fashion behind the bullet, with the wavefront propagating outward at the speed of sound. The angle at which the shock wave cone expands is dependent on the bullet's speed [53].

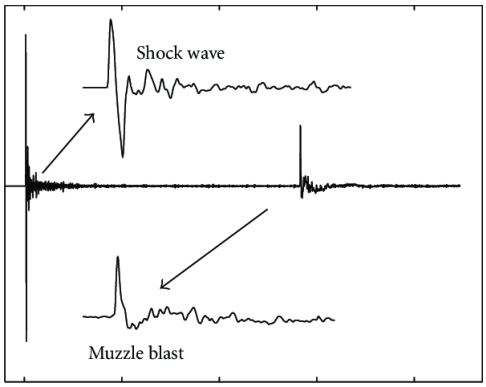


Figure 3.5 – Signal received by a microphone located 180 m from a firing gun. The speed of the bullet is 767 m/s and it passes the microphone at a distance of 30 m. The shockwave from the supersonic bullet reaches the microphone before the muzzle blast [54].

3.3.2 Analysis of Gunshot Recordings and Environmental Effects

The presence of acoustic events in the firearm sound depends on several parameters and will affect the detection procedure. Hence, when building an acoustical gunshot detection system, it is crucial to consider these parameters. In other words, different scenarios should be introduced into the training data to generalize the deep learning model.

Consider the microphone placement; As depicted in Figure 3.6, Supersonic Projectile expands behind the bullet in conic fashion and not in every direction, which means that depending on the microphone's position, the projectile's expanding shock wave cone may vary in strength or even not intercept the microphone at all. Additionally, the muzzle blast would also be of lesser amplitude than for the shots made with the muzzle facing towards the microphone [53].

Other acoustic events that may be present within the firearm sound are the reflection of both the shock wave and the muzzle blast. Like any other physical wave phenomena, they are subject to modifications as they propagate. This means that the microphone will also receive the acoustic pressure waves arriving later from other directions due to reflections and scattering [53]. Figure 3.7 shows the audio signal that the microphone recorded in different distances and positions. It is also worth reminding that the significance of the reflections depends on the environment. For instance, the amount of the reflection will vary from indoors to rural areas, mountainsides, or wide-open areas.

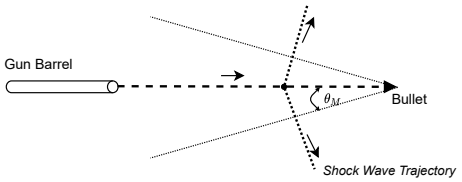
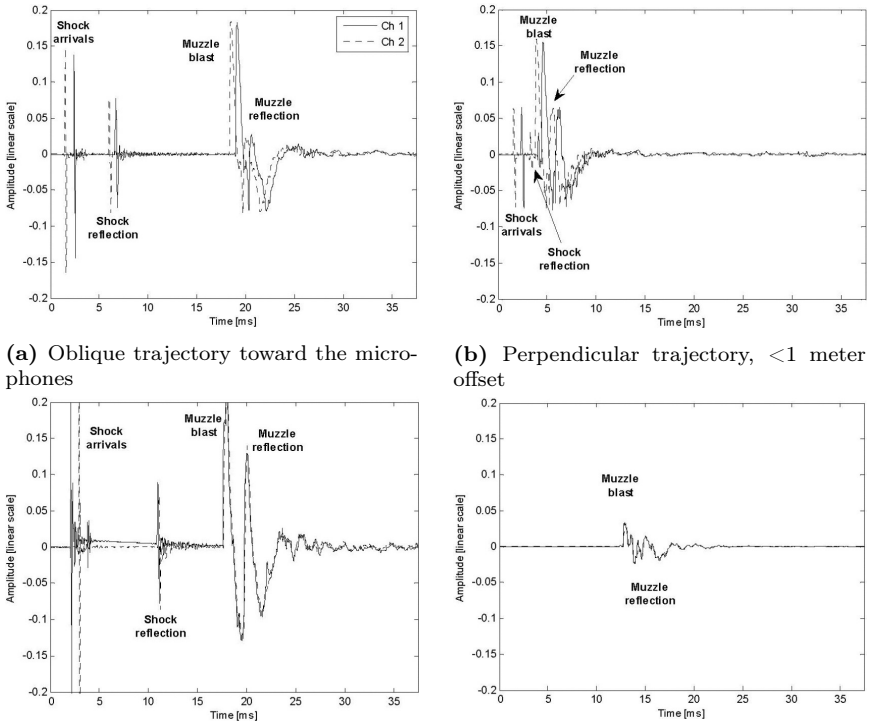


Figure 3.6 – Shock wave geometry for a supersonic projectile. The Mach Angle θ_M is small for $(V/c) \gg 1$, and close to 90° for $(V/c) \approx 1$[53]

The muzzle blast acoustic wave may be influenced by the surrounding obstacles, ground surface, temperature and wind gradients in the air, and atmospheric absorption. A recording microphone located close to the firearm could detect the direct sound of the muzzle blast as the primary acoustical signal. However, the further the microphone is located from the firearm, the higher the chance that the direct sound path is obscured. The received signal might therefore exhibit propagation effects, multi-path reflections, and reverberation. Another discrepancy in the received signal might be caused by a fire gun with a suppressor. Suppressors are designed to reduce the muzzle blast's audible report to lower the likelihood of detection or prevent hearing damage. Thus, a decent gunshot acoustical detection system that relies on the muzzle blast must take the possibility of suppressor use into account.



(a) Oblique trajectory toward the microphones

(b) Perpendicular trajectory, <1 meter offset

(c) Perpendicular trajectory, 8 meter offset

(d) Opposing trajectory (muzzle pointed away from microphones)

Figure 3.7 – Different positioning of a two-channel microphone [53]

3.4 Deep Learning for Audio Processing

This Section aims to explain some of the deep learning subjects used in general. Since deep learning is a sub-field of machine learning, the common concepts of machine learning are introduced in Section 3.4.1. Subsequently, Section 3.4.2 overviews the concepts of Neural Networks and methods used in this research study.

3.4.1 Basics of Machine Learning

Machine learning is a sub-field of artificial intelligence (AI) focusing on using data and algorithms for the program to imitate the way that humans learn, gradually improving its accuracy [55]. A *machine learning model* is the generated output of a learning or training procedure through the use of statistical methods and data.

There are several steps to generate a machine learning model. After the problem is defined, the first requirement is to collect and process data that the model will be trained on. Subsequently, a training algorithm, so-called a model, is chosen. There are a variety of techniques that can be used for the learning

procedure, such as supervised, unsupervised, and reinforcement learning.

Next, regardless of the method, the learning system of a machine learning algorithm breaks out into three main parts; a *Decision Process*, an *Error Function*, and a *Model Optimization Process*. In general, machine learning algorithms make a prediction or classification based on some input data. Subsequently, A loss function evaluates the model's prediction, and based on the known examples, makes a comparison to assess the model's accuracy. Finally, if the model can be improved and fit better to the training data, the weights are modified to reduce the discrepancy between the known examples and the model estimate. The algorithm will repeat this process and optimize the model, updating weights autonomously until a threshold of accuracy has been met [56].

The learning process or *fitting* is distinguished between two general categories of supervised and unsupervised. Supervised techniques are used to predict or explain the data. It is done using previous input-output pairs to predict output for new input [57]. By contrast, unsupervised machine learning evaluates data in terms of similarities called traits. These traits are, in turn, used to form clusters of items [57].

No machine learning algorithm necessarily fits every purpose. Depending on the needed output, different learning algorithms can be chosen. e.i., to build a gunshot recognition system, a CNN can be trained based on previous pairs of *gunshot audio events-labels* to classify the presence of gunshot sound in an audio frame.

To improve the model's performance, the model must be able to generalize. In other words, high accuracy on the dataset that the model is trained on does not necessarily mean that the model has comparable results on unseen data. For this reason, the dataset is usually split into three sets during the training process. The model is trained on the "*training*" set, while its performance is observed on the "*validation*" set. Subsequently, the model accuracy is determined based on its performance on the "*test*" set.

Confusion Matrix

In the case of classification problems, a commonly used metric for accuracy measurements is the Confusion matrix. A confusion matrix provides valuable information about the accuracy of different labels/classes. The rows of the matrix represent the instances in an actual class, and the columns represent the instances in a predicted class, or the other way around [58]. Table 3.4 presents a confusion matrix for binary classification. If the actual values are defined as *True* and *False*, and the predictions are presented as *Positive* an *Negative*, each element in the confusion matrix are as below. A matrix similar to an identity matrix indicates a %100 accurate model. FP and FN are the classification algorithm's errors.

Furthermore, useful metrics such as Recall, Precision, Accuracy can be extracted from the confusion matrix. The metrics for binary classification are explained below.

Actual \ Predicted	N	P
N	TN	FP
P	FN	TP

Table 3.4 – Confusion matrix for binary classification. TN is True Negatives, FP is False Positives, FN is False Negatives, and TP is True Positives.

- Accuracy: is the percentage of correctly predicted classes.

$$Accuracy = (TN + TP)/(P + N) \tag{3.6}$$

- Recall: is the percentage of correctly predicted positive class.

$$Recall = TP/(TP + FN) \tag{3.7}$$

- Precision: is the ratio between the True Positives and all the Positives.

$$Precision = TP/(TP + FP) \tag{3.8}$$

3.4.2 Deep Learning

Deep learning is a sub-field of Machine learning based on artificial neural networks. It can use unstructured data in its raw form as input and automatically determine the features that separate various classifications of data from one another. As opposed to machine learning, deep learning does not require human intervention to extract data features. Hence it allows to scale the learning process in more extensive ways [55]. Deep Learning models are designed to emulate how the human brain works to deal with abstractions and problems that are poorly defined [55]. They analyze data with a logical structure similar to how humans conclude [59]. To understand how deep learning algorithms work, one must get familiar with neural networks.

Neural Network

A neural network or an *artificial neural network* (ANN) is a series of algorithms that attempts to seek and understand underlying relationships in a set of data via a process that mimics the human brain.

A neural network consists of three or more layers: an input layer, one or many hidden layers, and an output layer. Data is obtained through the input layer, modified in the hidden layer(s), resulting in the output layer. Each layer contains a number of nodes that are connected to the next layer of nodes, as depicted in Figure3.8.

At a basic level, one can model the neural network similar to linear regression, where the nodes are associated with a weight (w_i) and a bias (b_i). The algebraic formula for a single node is then $y = \sum_{m}^{i=1} w_i x_i + b_i$ where x_i are inputs or

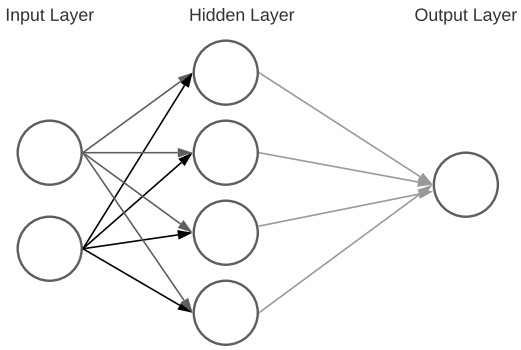


Figure 3.8 – A simple Neural Network

former layer nodes and y is the output or a next layer node. Subsequently, an *activation function* is used to define whether the produced y value is going to be transformed and passed to the next layer in the network. The most famous activation functions are *ReLU*, *Sigmoid*, and *Tanh*.

Depending on the problem, one activation function might perform better than the others. For example, the sigmoid function generally works better in the case of classifiers [60]. However, to find the best fit, it is advised to begin with one of the functions and test the others. For instance, in this project, the activation function chosen for the initial CNN model is ReLU. However, all of the activation functions are used and compared to find the best fit in the later stages.

ReLu:
$$f(x) = \begin{cases} x & if \quad x > 0 \\ 0 & if \quad x \leq 0 \end{cases} \tag{3.9}$$

Sigmoid:
$$S(x) = \frac{1}{1 + e^{-x}} \tag{3.10}$$

Tanh:
$$tanh(x) = \frac{e^{x} + e^{-x}}{e^{x} + e^{-x}} \tag{3.11}$$

While it is implied within the name, it is worth mentioning that the *"deep"* in deep learning refers to the depth of layers in a neural network. A deep model is a neural network that consists of more than three layers, including the inputs and the output, i.e., *DNN* (Dense NN). While DNNs are fully connected, deep models may consist of other variations of the neural networks such as CNN, RNN, or a combination of them, as well as a combination of neural networks and classical machine learning algorithms at later stages.

CNN

A Convolutional Neural Network (CNN) is a kind of artificial neural network, most commonly applied to analyze visual imagery. The name is derived from the mathematical operation called *convolution* that such networks utilize. CNNs

are specialized neural networks that use convolution in place of general matrix multiplication in at least one of their layers [61].

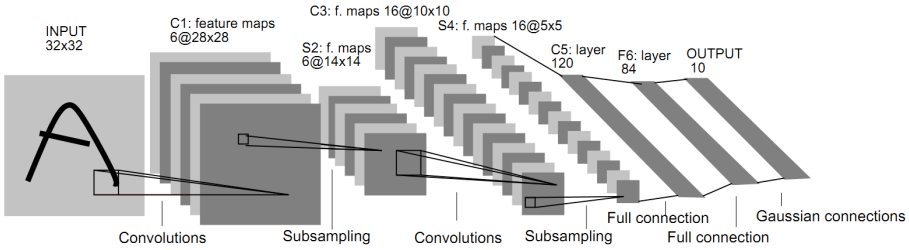


Figure 3.9 – Architecture of LeNet-5, a convolutional NN, here used for digits recognition. Each plane is a feature map, i.e., a set of units whose weights are constrained to be identical [62].

CNNs have three main types of layers, *Convolutional* layers, *pooling*, and *a fully connected (FC)* layer(s) [63]. While convolutional layers can be followed by additional convolutional or pooling layers, the fully connected layer is the final layer. A visual representation of a CNN model, a LeNet-5 Architecture, is shown in Figure 3.9. The model consists of two sets of convolutional and average pooling layers, followed by a flattening convolutional layer, then two fully-connected layers, and finally a softmax classifier [62].

A convolutional layer operates by extracting features from the input using a filter. With each layer, the CNN increases in its complexity, identifying greater portions of the input [63]. Earlier layers focus on simple features, i.e., in the case of image processing, colors and edges. As the image data progresses through the layers of the CNN, it starts to recognize larger elements or shapes of the object until it finally identifies the intended object [63].

After extracting the features by the convolution layer, pooling is usually used to reduce the number of parameters for the successive layers. Two common types of pooling layers are max pooling and average pooling. A *max pooling layer* downsamples the input by taking the maximum value over a certain window on the input data. An *average pooling layer* downsamples the input data by calculating the average value of the input data over a certain window [63].

Before the data is put into the fully connected layers, it is usually flattened to reduce dimensionality. It is converted from a matrix to a vector with only one dimension. Subsequently, fully connected or dense layers are used as the final part of the model. The last layer classifies the input based on the features extracted through the previous layers and their different filters [63].

Hyperparameter Optimization

When building a deep learning model, there are design parameters that define the model architecture. For instance, in the case of a CNN model, these parameters are the number of layers, number of nodes per layer, kernel size, learning rate, optimizers, and activation functions. This so-called *hyperparameters* govern the training process. Hyperparameter optimization, also known

as *tuning*, is the problem of choosing a set of hyperparameters to optimize a learning algorithm [64].

There are several reasons for using hyperparameter optimization instead of relying on intuition [65], such as vanishing or exploding gradient, encountering local optima. Additionally, tuning the hyperparameters yields an optimal model by finding the best tuple.

Hyperparameter tuning works by running multiple trials in a single training job [66]. Each trial is a complete execution of the training application with values chosen for hyperparameters, set within the specified limits [66]. There are several methods for hyperparameter tuning, such as Grid search, Random search, Bayesian optimization.

Since only the Bayesian optimization is used for hyperparameter tuning in this project, a brief explanation of this method is explained below:

Bayesian optimization[64]

"Bayesian optimization builds a probabilistic model of the function mapping from hyperparameter values to the objective evaluated on a validation set. By iteratively evaluating a promising hyperparameter configuration based on the current model and then updating it, Bayesian optimization aims to gather observations revealing as much information as possible about this function and, in particular, the location of the optimum."

With stating the related works and the necessary background information, the next Chapter aims to present the methods used for this research study.

Chapter 4

Methodology

This chapter discusses the methods and experimental setup used in the project. The project consists of three steps:

- Collecting and preparing data.
- Training CNN models for different weather conditions as well as SNR values to investigate a possible correlation between model accuracy and SNR.
- Building and optimizing a single model for gunshot detection in African wildlife.

First, Section 4.1 explains how the data is obtained, altered, and created to meet the needs of this research study. Subsequently, Section 4.2 explains the experiments done using deep learning techniques to investigate the central hypothesis of this research study. Finally, Section 4.3 overviews the tools and packages used in this thesis.

4.1 Data Preparation

This section describes how the data is prepared for deep learning experiments. First, the data collection is explained. Subsequently, the data augmentation methods and the algorithm to create synthetic data are explained.

As discussed in Chapter 2, one of the challenges in Environmental sound detection is the diversity of the case studies and the lack of a complete unit dataset that covers them all. To the best of the author's knowledge, there are no publicly available datasets suited to the needs of this research study. However, there are famous audio datasets such as AudioSet or UrbanSound8K that include gunshot sounds, among other events that have a low rate of occurrence within this study. For instance, it is unlikely that urban noises such as vehicle and traffic noises, constructions, or indoor activities such as inaudible cafe noise, music take place within the savanna wildlife. Therefore, including these events in this project will only introduce unnecessary diversity and complexity.

Hence, only the gunshot sounds are collected from publicly available datasets listed bellow:

- UrbanSound8K
- 5 gunshot sounds [67]
- Gunshot audio dataset [68]
- Sound Events for Surveillance Applications [69]

The collected gunshot sounds are mostly pure signals, meaning there is no background noise. However, outdoors, specifically, the wildlife, is unlikely to be completely silent. One can usually hear a diverse range of sounds within the savanna wildlife; for instance, leaves rustling, wind, birds, insects, animal sounds, rain, and thunder. Therefore, some ambient sounds are collected to create a more realistic simulation of what microphones' will pick up within the field. The ambient sounds are treated as background noise for the gunshot sounds. To be more precise, the dataset created for this research is audio samples that are ambient sounds, with a chance of gunfire happening in 50% of the samples. This method is also used in [19, 12, 21, 22, 18]. The Ambient sounds collected from recordings hosted on YouTube are listed below:

- **Savanna sounds**; Two long tracks of ambient sounds in savanna wildlife recorded by George Vlad [70, 71]. One track is recorded in the daytime and the other at night. Insect chirps, distant bird calls, and occasional animal sounds are recognizable within the recording.
- **Rainfall** [72]; Since Savannas have 6 - 8 months wet summer season on average [73], rainfall sounds are also collected to simulate the rainy days.
- **Thunder** [74]; During the dry season, lightning often strikes the ground [73]; hence the thunder sounds.

The following section explains how the samples are created using the collected sounds.

4.1.1 Data Synthesis

After collecting the raw tracks, taking the advice of [27], new samples are synthesized using data augmentation methods to introduce more diversity and generalization to the dataset. The main idea is to stack multiple tracks on top of each other and create less pure signals. The resulting signals, thus, contain ambient sounds[1], which simulates a more realistic environment.

Since the tracks are collected from different datasets, they are of different lengths. For example, gunshot sounds are one to twenty seconds long. The longer tracks include silent sections and gunfire at an arbitrary moment. The savanna, rain, and thunder soundtracks are an hour-long at least. All the tracks

[1]Ambient sounds and background noise refer to the same thing and might be used interchangeably within the context.

are fixed at a desired duration, ten seconds for equal data distribution and removing any time-based bias. The longer tracks are split into ten-second frames, and the shorter ones are extended by adding a silent section; this resulted in more than 200 tracks per sound category.

Each track is then defined as a *signal layer* that has a few properties[2]:

- **Source**, the path to the signal file.
- **Duration:** The fixed duration of the signal after loading. Shorter signals are padded with zeros.
- **Max time shift:** The maximum range of time-shifting in seconds, applied on the signal. The signal starts later or earlier to make sure the model can detect sounds at different points in time.
  The time-shifting is a random value within the range assigned when the signal is loaded. It is, thus, varied for every signal.
- **Max amplitude:** The relative amplitude maximum applied to the signal; After the signals are normalized between $(-1, 1)$, they are scaled to the desired max amount to simulate Signal-to-Noise ratio.

After assigning the signal layers' parameters, they are stacked on top of each other, and a new sample is synthesized.

The steps taken for data augmentation are stated in the Algorithm 1. Variable n defines the number of samples chosen per category, p is the number of categories, and 2^p is the number of possible permutations between signal layers within the synthesized samples. i.e., $n = 100, p = 2$ if one hundred files are selected from two categories of savanna and gunshots, in total, using this algorithm, $n.2^p = 400$ samples are synthesized.

The data created using this algorithm is a vector of MFCCs or raw waveforms and their corresponding labels, *gunshot* or *no gunshot*.

4.1.2 Modeling the SNRs

To model different SNRs, five separate datasets are created using the algorithm explained in 1. The only difference between them is the amplitude scale of the gunshot sounds. The reason behind this is to model the shooter's distance from the microphones. The greater distance will result in a lower sound level, which would be harder to distinguish from the background noise. Table 4.1 shows the relative amplitude scales of the samples within the five datasets. $SNR_{i:1}$ represents a dataset that the gunshot sound level is i times the background noise in all the samples containing gunfire sound.

[2]Created samples contain at least one signal layer

Algorithm 1 Data Generator

Require: valid address to the categories
Ensure: an array including the synthesized samples
Define the layers and their parameters
Determine possible permutations of the layers; $2^p =$
for $i = 0$ to $i < 2^p$ **do**
Create randomized vectors of size n with time-shifting and amplitude parameters
Load n signals per category
Apply time-shifting and amplitude scaling on all loaded signals
Stack all layers on top of each other, resulting in a total of $n2^p$ stacked signals
Apply feature extraction for all stacked signals if specified
Label the synthesized sample into a binary category with respect to the presence of gunshot sound. (gunshot, no gunshot)
Save the extracted features or raw signals to a file
end for

Dataset	**Relative Amplitude Ratio**	
	Gunshot	Background Noise
$SNR_{1:1}$	1	1
$SNR_{2:1}$	2	1
$SNR_{3:1}$	3	1
$SNR_{4:1}$	4	1
$SNR_{5:1}$	5	1

Table 4.1 – Five separate datasets used in the project. The only difference between them is the amplitude scale of gunshot sounds. Higher SNR happens when the shooter is closer to the microphone.

4.2 Experiments

The next challenge after preparing the data is to build an acoustic gunshot recognition model and investigate the potential correlation between the transmitter-receiver distance and the accuracy of the sound recognition algorithm. This Section presents the experiments conducted for this research study. First, the fixed configurations throughout the experiments are discussed, such as the dataset split ratio (train, test, and validation set)—subsequently, the motivation and design of the experiments are presented. Finally, a single deep learning model is proposed to deploy on the end-nodes.

Deep learning models for various settings are trained to detect gunshot sounds using the datasets created with the method described in Section 4.1.1. Throughout the experiments, data is split into three sets of training, validation, and test, with a ratio of $0.55 : 0.2 : 0.25$, respectively. The gunshot recognition is treated as a binary classification problem, and the model performance is evaluated on the test set using the confusion matrix and the accuracy metric explained in Section 3.4.1. Table 4.1 presents the confusion matrix layout for the experi-

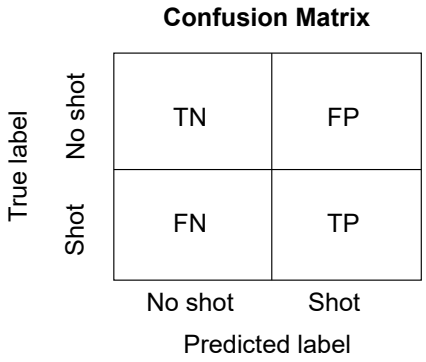


Figure 4.1 – Confusion matrix layout throughout the experiments.

Category	Parameter	Value
Training parameters	Optimizer Learning rate	SGD 0.01
1^{st} Convolutional layer	Filters Kernels Activation	32 (3,3) ReLU
Max pooling	Size	2
2^{nd} Convolutional layer	Filters Kernels Activation	16 (3,3) ReLU
Max pooling	Size	2
1^{st} Dense layer	Units Activation	500 ReLU
2^{nd} Dense layer	Units Activation	250 ReLU

Table 4.2 – CNN model parameters

ments. The architecture and the parameters of the initial CNN model used for the experiments is presented in Figure 4.2 and Table 4.2. The input features used for the experiments are 13 MFCC bins, and the epoch and batch size for the model fitting process is 100, and 8 respectively. Unless stated otherwise, this configuration is kept fixed throughout the project.

4.2.1 SNR Effect on the Model Accuracy

The first experiment investigates the correlation between gunshot recognition accuracy and SNR. The data used for this experiment consist of Savanna ambient and gunshot sounds, such as presented in Table 4.1. Savanna sounds are always present within the samples, with half of them being the daytime sounds and the other half nighttime sounds. Gunshot sounds are only present in 50% of the samples. Two hundred samples are selected from each category, which

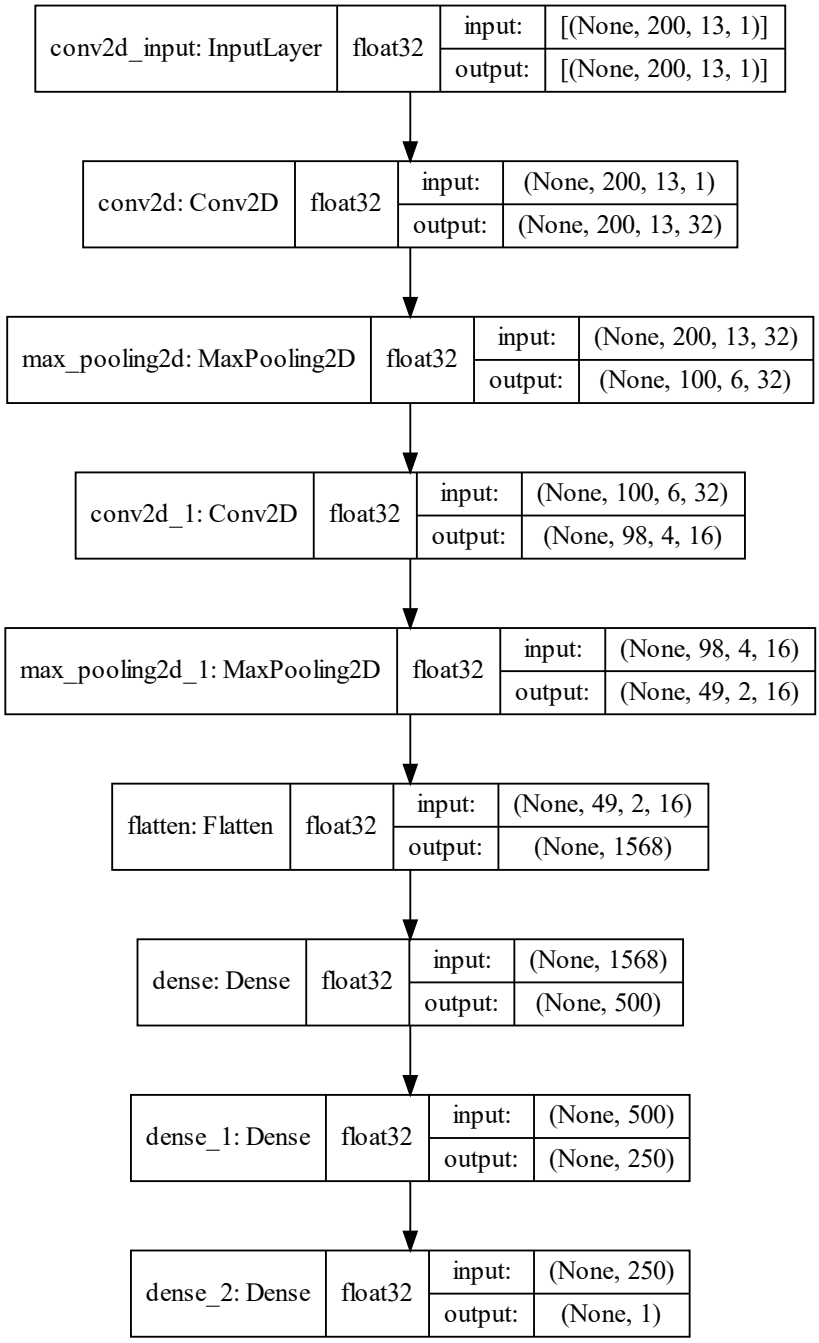


Figure 4.2 – CNN model layout used for the experiments

yields $200 \times 2^2 = 800$ samples in total per dataset. The model is fit for the five SNR datasets created in Section 4.1.2 each with a sample size of 800.

The initial CNN model presented in Figure 4.2 and Table 4.2 is used for training. The accuracy of the five trained models is then compared to see if the noise level, ergo the shooter distance, affects the predictions.

4.2.2 Rain and Thunder Effect

This experiment aims to investigate the weather effects on the model accuracy. In addition to the Savanna ambient sounds, rain and thunder sounds are added to the training data. It is expected that weather conditions affect the gunshot recognition accuracy. The experiment is done in two parts:

Part One

In the first part, the effects of rain and thunder are separately inspected at different levels. This part aims to see how heavy and light rainfalls influence classification accuracy. The same goes for thunderstorms. Depending upon the nature of the lightning and its distance from the acoustic sensor, thunder sound can range from a long, low rumble to a sudden, loud crack [75]. The thunder sounds are inspected to see if they would wrongly be classified as False Positives.

Table 4.3 presents the datasets used for this part. Two hundred files are selected from each category. Since the effect of rain and thunder are separately analyzed in this experiment, three sound categories are selected each time. Hence the sample size of each dataset is $200 \times 2^3 = 1600$. Note that for this part of the experiment, rain or thunder sounds are always present within the samples. Same as the first experiment, the CNN model presented in Figure 4.2 is used for training.

Dataset	**Relative Amplitude Ratio**			
	Gunshot	Savanna ambient	Rain	Thunder
$Rain_{1:1}$	1	1	1	-
$Rain_{1:2}$	2	1	1	-
$Thunder_{1:1}$	1	1	-	1
$Thunder_{1:2}$	2	1	-	1

Table 4.3 – Supplementary datasets used for experiment 4.2.2. The purpose of this experiment is to observe the effects of different levels of rain and thunder sounds separately.

Part Two

In the second part of the experiment, it is assumed that both rain and thunder might occur within the samples to introduce more randomness to the training set. Table 4.4 shows the five datasets used for this experiment. Note that the Savanna sounds are always present within all the samples. The other three categories, rain, thunder, and gunshot sounds, each have an independent 50%

chance of occurring, and if they do, their related power ratio will be the same as in Table 4.4. One hundred files are selected from each category, so each SNR dataset has $100 \times 2^4 = 1600$ samples.

Since more noise is injected into the training data, a decrease is expected in the model accuracy. Same as the former experiments, the CNN model presented in Figure 4.2 is used for training, and the model is fit five times.

Dataset	**Relative Amplitude Ratio**			
	Gunshot	Savanna ambient	Rain	Thunder
$SNR_{1:1}$	1	1	1	1
$SNR_{2:1}$	2	1	1	1
$SNR_{3:1}$	3	1	1	1
$SNR_{4:1}$	4	1	1	1
$SNR_{5:1}$	5	1	1	1

Table 4.4 – Five separate datasets that simulate different signal-to-noise ratios. Rain and thunder sounds are introduced as background noise. The difference between the datasets are the amplitude scale of gunshot sounds. Higher SNR happens when the shooter is closer to the microphone.

4.2.3 Sample Size Effect

This experiment investigates the sample size effect on the model accuracy. It is expected that the accuracy increases with the sample size. The data used in this experiment contains all four sound categories, such as in Table 4.4. The sample size, $n \times 2^4$, is increased based on the number of files chosen from each category. This experiment is done on one type of SNR dataset. The CNN model introduced in Figure 4.2 and Table 4.2 is trained for four sample sizes. The accuracy of the trained models is then evaluated on a fixed test set. It is expected for the model accuracy to increase for larger sample sizes.

4.2.4 Hyperparameter Tuning for CNN Model

For CNN model, instead of relying on intuition for model parameter values, hyperparameter optimization explained in Section 3.4.2 is used. SNR Datasets used for this experiment are the same as Table 4.4. Savanna sounds are always present within all the samples. The other three categories, rain, thunder, and gunshot sounds, each have an independent 50% chance of occurring. One hundred files are selected from every category, so each SNR dataset has $100 \times 2^4 = 1600$ samples.

Hyperparameter tuning is done five times for the CNN model since there are five SNR datasets. Bayesian optimization is used for hyperparameter tuning. Table 4.6 shows the search space prepared for the oracle. The maximum number of convolution layers is calculated with Equation 4.1. In this Equation, p represents the pooling layer size, and x represents the input size. Note that this experiment will result in five separate models. The purpose is to show the effect

of hyperparameter optimization on the accuracy for different SNR levels. It is expected to see an increase in model accuracy for all five models.

$$n = \lfloor \log_p(x) \rfloor \tag{4.1}$$

Algorithm	Bayesian Optimization
Objective	Maximum validation accuracy
Max trials	100
Patience	10

Table 4.5 – hyperparameters optimization setting

Parameters	**Range**
Number of convolution layers	Defined by Equation 4.1, $p = 2$; $[1, 3]$
Number of dense layers	$[1, 5]$
Optimizers	SGD, Adam, RMSProp, Adadelta, Adagrad, Adamax, Nadam, Ftrl
Learning rate	0.01, 0.001, 0.0001
Number of convolutional filters per layer	[8, 64], Step: 4
Convolutional kernel size (x and y)	[3,10]
Convolutional activation	ReLU, Sigmoid, tanh
Max pooling size (for each conv. layer)	Fixed; 2
Dense layer units per layer	[50, 500], Step: 50
Dense layer activation	ReLU, Sigmoid, tanh

Table 4.6 – Hyperparameter tuning search space for CNN model

GPU vs. CPU

In the final part of this experiment, the hardware used for training the CNN model is compared. Again, Hyperparameter tuning is used in the training process. The data used for this experiment contains all four sound categories, such as in Table 4.4. One hundred files are selected from each sound category, so the sample size is 1600. The model is trained for five SNR datasets once using CPU and GPU.

The purpose is to see if the hardware choice for training affects the model performance. Alternatively, if deployed on the end-nodes, model accuracy must not change for different hardware.

4.2.5 Feature Extraction Effect on Model Accuracy: Raw Waveform Vs. MFCC

This experiment investigates the effect of input features on accuracy. In this project, two types of features are extracted from the synthesized data, *MFCCs* and *Raw Waveforms*.

MFCCs are used with success in many papers focused on audio classification problems [76, 7, 77]. Features such as the Discrete Wavelet Transform (DWT) or Mel spectrograms could also have promising results but will not be discussed further. On the other hand, Raw waveforms avoid hand-designed features, which should better utilize the improved modeling capability of deep learning models. However, this incurs higher computational costs and data requirements, and benefits may be difficult to realize in practice [23].

When raw waveforms are used as input, feature extraction is entirely left to the convolution layers. This type of data only contains amplitude values in the time domain and can be expressed as vectors with n elements. For the Time-domain series, a specific type of Convolutional neural network, called 1D (one dimensional), is used.

SNR Datasets used for this experiment are the same as Table 4.4. One hundred files are selected from each category, so each SNR dataset has $100 \times 2^4 = 1600$ samples. Hyperparameter optimization is used to find the best fitting for 1D CNN models. The tuning setting is the same as in Table 4.5, and the search space is presented in Table 4.7. Note that the tuning is done for each dataset separately; hence, five models are built in this experiment, and the hyperparameters might vary from one model to another. The accuracy of the models then are compared to those from the experiment 4.2.4

In this project, a sampling rate of $fs = 48000$ and duration of 10 seconds are used for audio signals, which produce vectors with $n = 480000$. Training a deep learning model with such a large input vector can be time consuming depending on the system. To reduce the size of the signal vector, the average value for every 0.5 milliseconds is used. 0.5 milliseconds cover $a = 24$ elements in the original vector which is calculated with Equation 4.3, where $Fs = 48000$ and the time is $s = (0.5/1000) = 0.0005$ seconds. This leads to a reduced input vector by a factor of 24. The new input vector thus contains 20000 elements. (calculated with Equation 4.3 where $Fs = 48000$, $s = 10$ seconds).

$$n = F_s * s \tag{4.2}$$

$$n = \frac{F_s * s}{a} \tag{4.3}$$

Parameters	**Range**
Number of convolution layers	Defined by Equation 4.1, $p = 25$,$x = 20000$; $= [1, 3]$
Number of dense layers	$[1, 5]$
Number of convolutional filters per layer	[8, 64], Step: 8
Convolutional kernel size (x and y)	[2,100]
Convolutional activation	ReLU, Sigmoid, tanh
Max pooling size (for each conv. layer)	Fixed; 25
Dense layer units per layer	[50, 1000], Step: 50
Dense layer activation	ReLU, Sigmoid, tanh

Table 4.7 – Hyperparameter tuning search space for one-dimensional CNN

4.2.6 Combining SNR datasets

The end goal of the study is to come up with a single model that has acceptable accuracy for different noise ratios and different weather conditions. In other words, the gunshot recognition algorithm should be accurate enough for various weather conditions and different SNRs. This experiment is designed to extend the training data to all SNR datasets. The SNR datasets used for this experiment are the same as Table 4.8 which is a repetition of 4.4. The data consist of all four sound categories. Savanna sounds are always present within the samples, with half of them being the daytime sounds and the other half nighttime sounds. The other three categories, rain, thunder, and gunshot sounds, each have an independent 50% chance of occurring.

Dataset	**Relative Amplitude Ratio**			
	Gunshot	Savanna ambient	Rain	Thunder
$SNR_{1:1}$	1	1	1	1
$SNR_{2:1}$	2	1	1	1
$SNR_{3:1}$	3	1	1	1
$SNR_{4:1}$	4	1	1	1
$SNR_{5:1}$	5	1	1	1

Table 4.8 – Five separate datasets that will be combined in this experiment. The difference between the datasets are the amplitude scale of gunshot sounds. Higher SNR happens when the shooter is closer to the acoustic sensor. Savanna sounds are present in all the samples. However the other three sound categories have a 50% chance of occurrence.

In this experiment, SNR datasets are added on top of each other in four steps to observe the effect of noise level diversity on the accuracy. Starting from the dataset $SNR_{5:1}$, dataset $SNR_{4:1}$ is added on top of the training data. The accuracy of the newly fit model is then compared to the accuracy of the model trained only for dataset $SNR_{4:1}$. It is desired that the model learn the gunshot pattern from the higher SNRs and perform better on the lower SNRs. Figure 4.3 presents the idea of the first step.

Similarly, datasets $SNR_{5:1}$ and $SNR_{4:1}$ are added on top of dataset $SNR_{3:1}$. The accuracy of the newly trained model is then compared to the accuracy of the model trained only on the dataset $SNR_{3:1}$. This process is repeated for $SNR_{2:1}$, and $SNR_{1:1}$. Figure 4.4 presents the idea of the last step of this experiment.

The experiment is divided into two parts to separate the sample size effect from the new diversity introduced to the training data. First, the total sample size is kept fixed. For instance, if in Figure 4.3 the first model is trained on 1600 samples, this will also be the case in the second model. In other words, to train the second model, 800 files are selected from both datasets $SNR_{5:1}$, and $SNR_{4:1}$, so the total size will still be 1600. Likewise, in Figure 4.4 the sample size will be 1600. The data used for training the second model in this Figure consists five SNR datasets, $5:1$, $4:1$, $3:1$, $2:1$, and $1:1$. Hence, 320 samples are selected from each dataset.

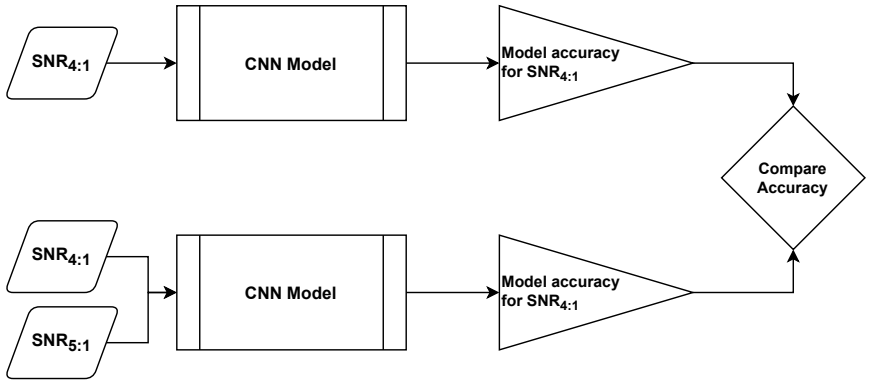


Figure 4.3 – Comparing the accuracy of two CNN models for $SNR = 4:1$. The first model is trained on one noise levels, while the second model is trained on two noise levels.

In the second part, however, the number of samples selected from each dataset is fixed; hence, the total sample size increases. For instance, if in Figure 4.3 the sample size of the first model is 1600, the second model will be trained on 3200 samples. This means that 1600 files are selected from both $SNR_{5:1}$, and $SNR_{4:1}$ datasets. Likewise, in Figure 4.4, the second model will be trained on 8000 samples since the training data contains five different SNR datasets $5:1$, $4:1$, $3:1$, $2:1$, and $1:1$.

4.2.7 Final Proposition

After conducting various experiments and observing the effects of different parameters on the gunshot classification accuracy, a unique model is proposed for gunshot detection in wildlife. Hence, the dataset characteristics and size, input features, and model parameters will be determined based on the experiments' results.

4.3 Tools

This section describes the hardware setting and software tools used for the experiments.

The programming language used for this project is Python. According to Statistics Times [78], Python is one of the top three most used programming languages in 2021 and the most used programming language for deep learning projects due to excellent community support and an extensive set of libraries [79].

Figure 4.5 presents the high-level flowcharts of the data synthesis and model training to give an idea of which libraries are used where. The important Python libraries used in this project are listed below. Since Tensorflow and Keras have user-friendly interfaces and large community support, they are used to a great extent in this project.

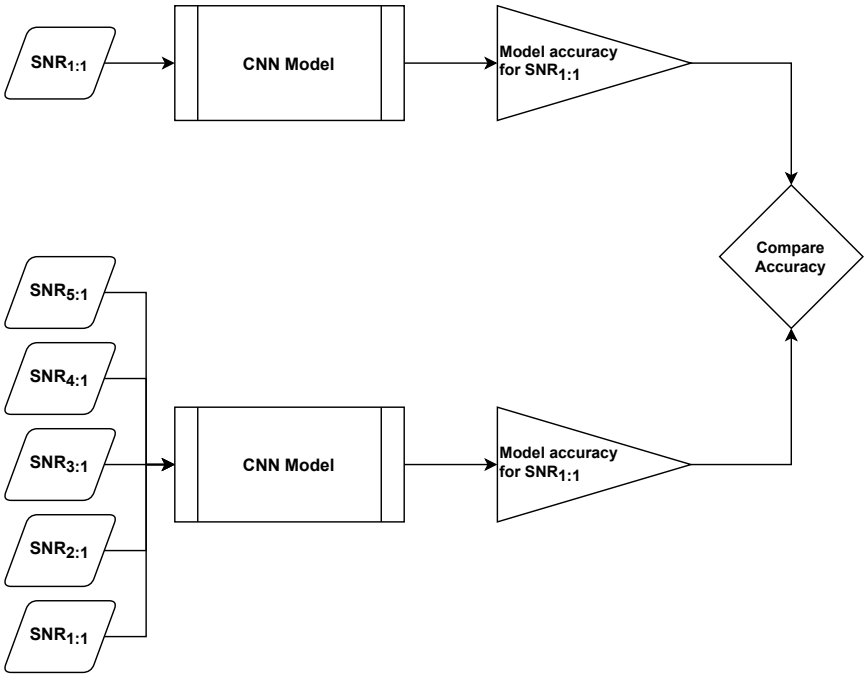


Figure 4.4 – Comparing the accuracy of two CNN models for $SNR = 1:1$. The first model is trained on one noise levels, while the second model is trained on five noise levels.

- **Librosa:** *"A python package for music and audio analysis [80]."* After gathering the soundtracks, to load and analyze *.wav* audio files Librosa package is used.

- **Pydub:** *"Manipulate audio with a simple and easy high level interface [81]."* This library is used for data preparation, to split longer audio files and extend shorter ones by adding silent segments.

- **Tensorflow:** *"an end-to-end open-source platform for machine learning [82]."* This library has been used extensively in both parts of data synthesis and model training. After loading and preparing the raw signals, Tensorflow library is used to generate the new data and extract MFCCs if chosen. This library is also used in model training, optimization, and evaluation.

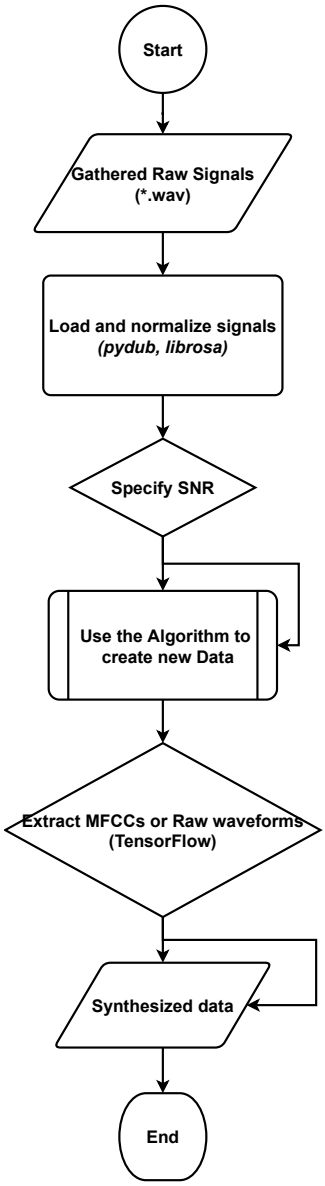
- **Keras:** is a deep learning API that runs on top of TensorFlow [83]. Keras has been used to build the CNN models and optimize them in this project.

- **Keras Tuner:** A library developed to implement hyperparameter optimization for models made with the Keras library. Keras Tuner is used for Hyperparameter optimization in this project. This tool makes it easier to define a search space and leverage algorithms to find the best hyperparameter values. The Bayesian Optimization technique is selected from the existing hyperparameter optimization techniques [84].

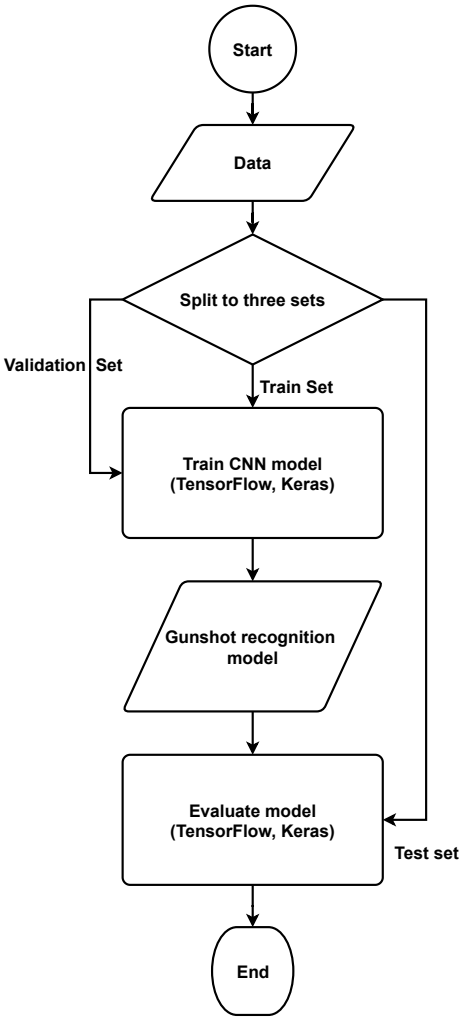
- **Matplotlib:** is a comprehensive plotting library in Python [85]. This library is used to depict the data analyses, plotting the CNN model architecture, and confusion matrices.

The hardware and operating system used for the experiments are listed bellow:

- GPU: NVIDIA GeForce GTX 1070
  - 1920 CUDA cores
  - 16 GB Shared Memory
  - 356.46 GB/s Memory Bandwidth
  - 1632 MHz Graphics clock
- CPU: Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz
- RAM: 32.0 GB
- Operating System: Windows 10 64-Bit



(a) High level flowchart of data synthesize



(b) High level flowchart of training gunshot recognition model

Figure 4.5

Chapter 5

Results

This section presents the results of the data and the experiments designed for this research study using the tools and methods previously described in Chapter 4. Section 5.1 analyzes the acquired and prepared data. Section 5.2 presents the results of the experiments overviewed in Section 4.2.

5.1 Dataset Analysis

This Section overviews the acquired data explained in Section 4.1, and created datasets using the algorithm 1.

Table 5.1 shows the total duration of the soundtracks and the number of files per category after splitting. As stated in Section 4.1, before generating the new data, all the audio tracks are altered to be of ten-seconds length. The collected ambient sounds that are more than an hour-long have been split into ten-second frames. The gunshot tracks are also split or padded with silent segments into ten-second frames depending on their prior length.

Category	**Total Duration**	**Number of Files**
African Savanna Daytime	1 hour, 40 seconds	365
African Savanna Nighttime	1 hour, 8 seconds	361
Rain	1 hour, 40 seconds	365
Thunder	1 hour, 8 seconds	361
Gunshots	-	261

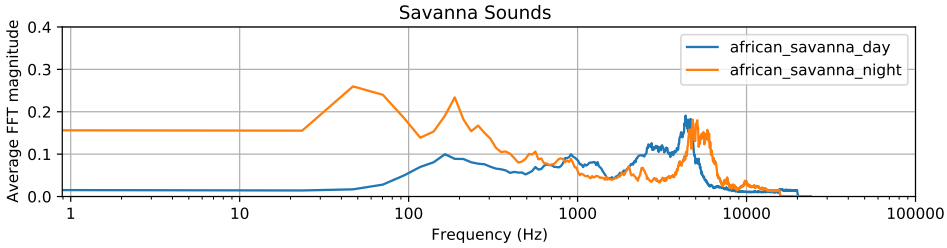
Table 5.1 – Total duration of the soundtracks and the number of files per category after split

For further analysis, a Python script is used to extract existing frequencies within the data. Figure 5.1 presents the results, which are grouped by sound category. The y axis represents the scaled FFT magnitude (all values are scaled to the range [0, 1]), and the x axis represents the frequency range in Hertz on a logarithmic scale. This analysis indicates that the FFT magnitude is higher in low-frequencies for all categories except the African savanna daytime sounds.

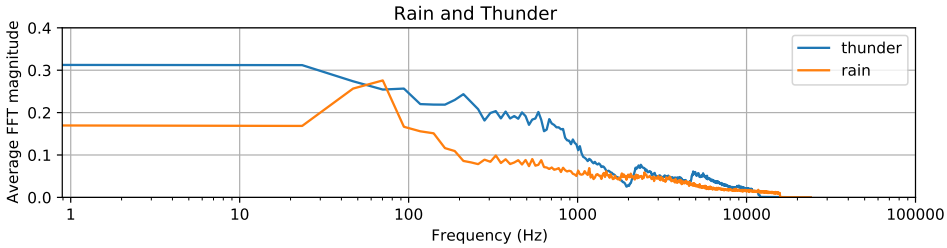
Figure 5.1a shows the frequency magnitudes for the savanna ambient sounds. The nighttime sounds have more low-frequency components compared to daytime. In both tracks, birdcalls and insect sounds are the most audible. A peak is observed around 5 kHz for both day and night, which was suspected to be the birdcalls.

To investigate this claim, additional tracks were gathered [86, 87, 88] which are sounds from various African birds. As presented in Figure 5.2a, the birdcall frequencies are from 1 Hz to 20 kHz. The magnitudes of frequencies from 1 to 5 kHz seem to be higher. However, there are also magnitude peaks around 500 Hz and 8 kHz. This observation is in line with the research done on bird hearings in [89]. The author claims that the avian hearing is primarily sensitive to sounds from 1 to 4 kHz, and depending on the species, the upper hearing limit can reach up to 20kHz.

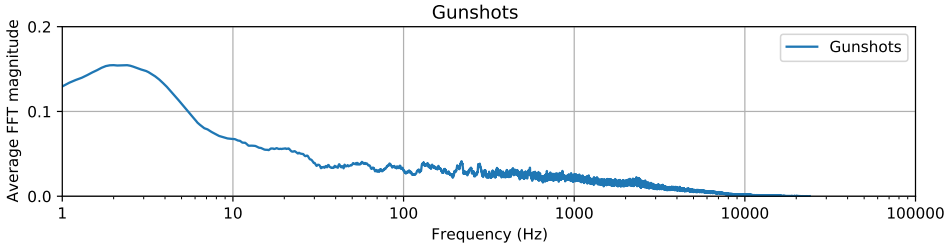
Figure 5.1b shows the frequency magnitudes for rain and thunder. Both categories have low-frequency components, $0 - 100Hz$. Thunder soundtracks also contain components in the $100 - 1000Hz$ frequency range.



(a) Frequency magnitudes of savanna sounds



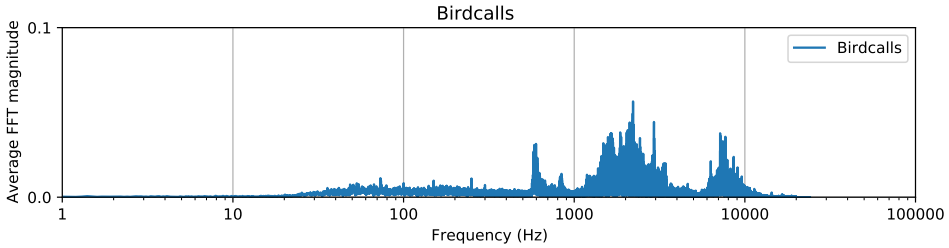
(b) Frequency magnitudes of rain and thunder



(c) Frequency magnitudes of gunshot sounds

Figure 5.1 – Existing frequencies of acquired data

Figure 5.1c shows the frequency magnitudes for the gunshot sounds. The



(a) Frequency magnitudes of birdcalls gathered from [86, 87, 88]

Figure 5.2

FFT magnitude of gunshot sounds is lower on average than other categories. As explained in Section 3.3.1, gunshot signal has two primary attributes, Muzzle Blast and Supersonic Projectile. One might expect larger magnitudes for higher frequencies due to the shockwave presence in the gunshot sound. However, it is worth reminding that the presence of a shockwave and its magnitude in gunshot signal depends on the distance and angle between the bullet trajectory and the acoustic sensor. Moreover, acoustic sensor characteristics have a non-negligible influence on the reproduced frequencies, which is looked at closely in Section 6.1. Unfortunately, there are no data indicating the shooting distance, angle, and microphone types used in the recordings.

For closer inspection, gunfire audio signals of a hunting rifle and an assault rifle are selected from [90]. Figures 5.3, 5.4 show the time and frequency domain representation of the gunshot sounds in two settings of far and close. Both Muzzle Blast and the shock wave are present within the signals. However, there are no traces of supersonic frequencies in the frequency-domain representation, which again shows the non-negligible influence of the recording microphones.

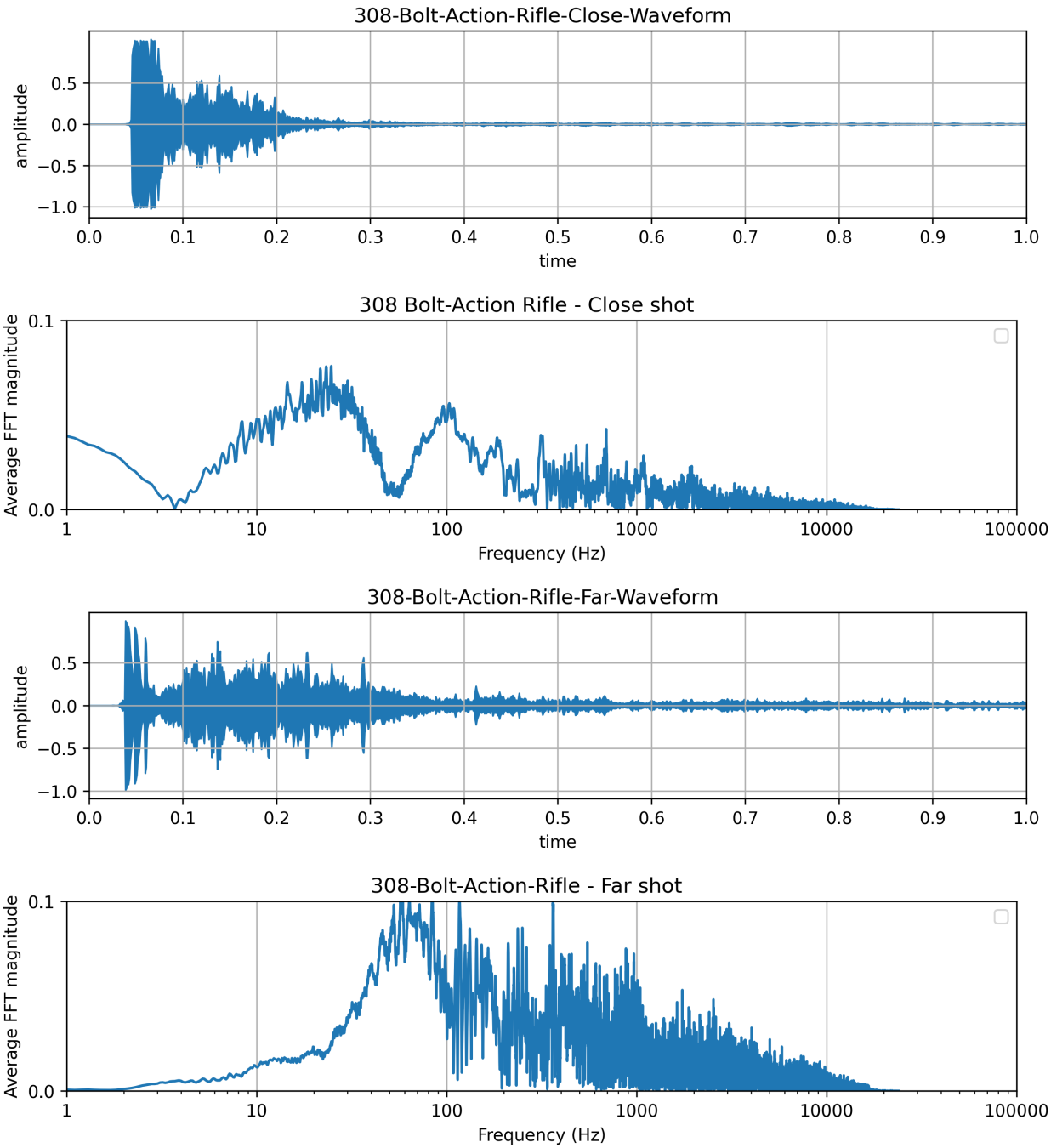


Figure 5.3 – Time and frequency domain representation of the 308-Bolt-Action Rifle in two settings, far, and close.

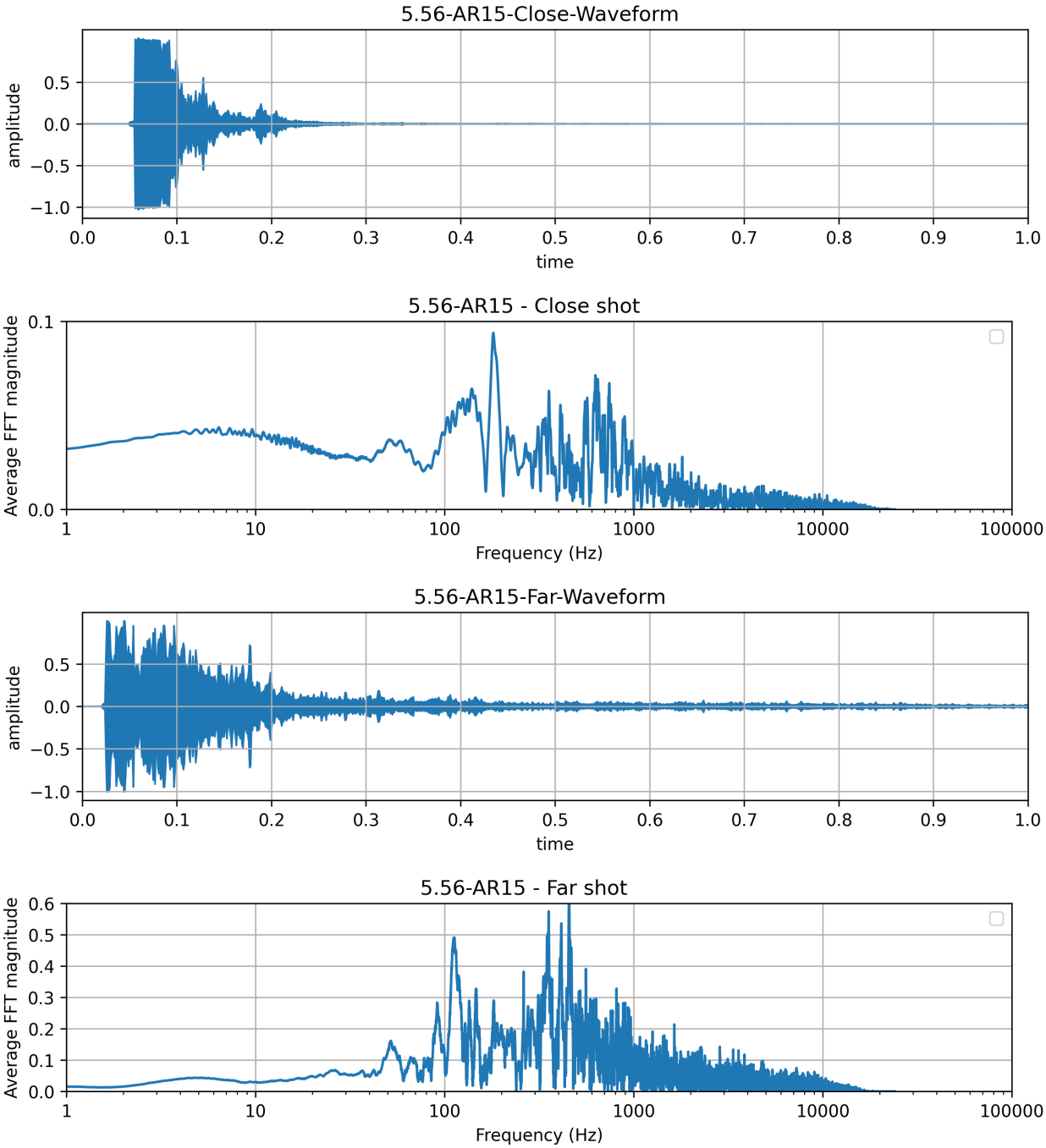


Figure 5.4 – Time and frequency domain representation of the 5.56-AR15 Assault Rifle in two settings, far, and close.

5.1.1 Synthesized Samples

The first step in creating new samples with Algorithm 1 is to define the soundtracks as signal layers. Each signal layer has a few properties as stated in Section 4.1.1. Table 5.2 presents the configuration for all the categories used in the experiments. The signal layers are all fixed at ten seconds. Rain and thunder sounds have random value time shifts with a maximum of five-second. A single gunshot sound happens at an arbitrary moment within the ten-second frames. The amplitude of gunshot sounds is varied depending on the specified SNR. If all four categories are chosen, there would be four signal layers. Following the Algorithm 1, the possible permutations would be sixteen.

Data Category	Fixed Duration	Max Time Shift	Max Amplitude Scale
Savanna Sounds	10s	0	1
Rain	10s	$\pm 5s$	1
Thunder	10s	$\pm 5s$	1
Gunshot Sounds	10s	Arbitrary	SNRs

Table 5.2 – Signal layer configuration for all the sound categories

If chosen, the datasets created with the algorithm have an independent binary probability for each category. The savanna sounds are chosen to always be present in the samples; however, half are selected from the daytime sounds and the other half from nighttime sounds. Equation 5.1 represents the possible permutations for four signal layers. For each permutation, one of the options from the tuples is selected.

$$[savanna_{daytime}, savanna_{nighttime}], [rain, -], [thunder, -], [gunshot, -] \quad (5.1)$$

As explained in Section 4.1.2, separate datasets are created to model each SNR. Each dataset size is calculated with the Equation $n.2^p$ explained in Section 4.1.1.

A visual representation of a random sample generated with the Algorithm 1 is shown in Figures 5.5,5.6. If chosen, with a window size of 50ms, MFCC features are extracted, resulting in a matrix of size (13, 200). As demonstrated in Figures 5.5,5.6, when the SNR is higher, the gunshot sound pattern is more distinct. Figures 5.7,5.8 visualizes the MFCCs of the same sample for the SNRs. Even though MFCCs are less readable for the human eye, the distinction in the lowest bin is still evident.

5.2 Experiments

In order to determine the best acoustic gunshot recognition model, different experiments are performed. This Section presents the results of the experiments described in Section 4.2. The performance measure of the models is based on the confusion matrix and the accuracy metric. The main goal is to achieve the highest possible prediction accuracy on the test set.

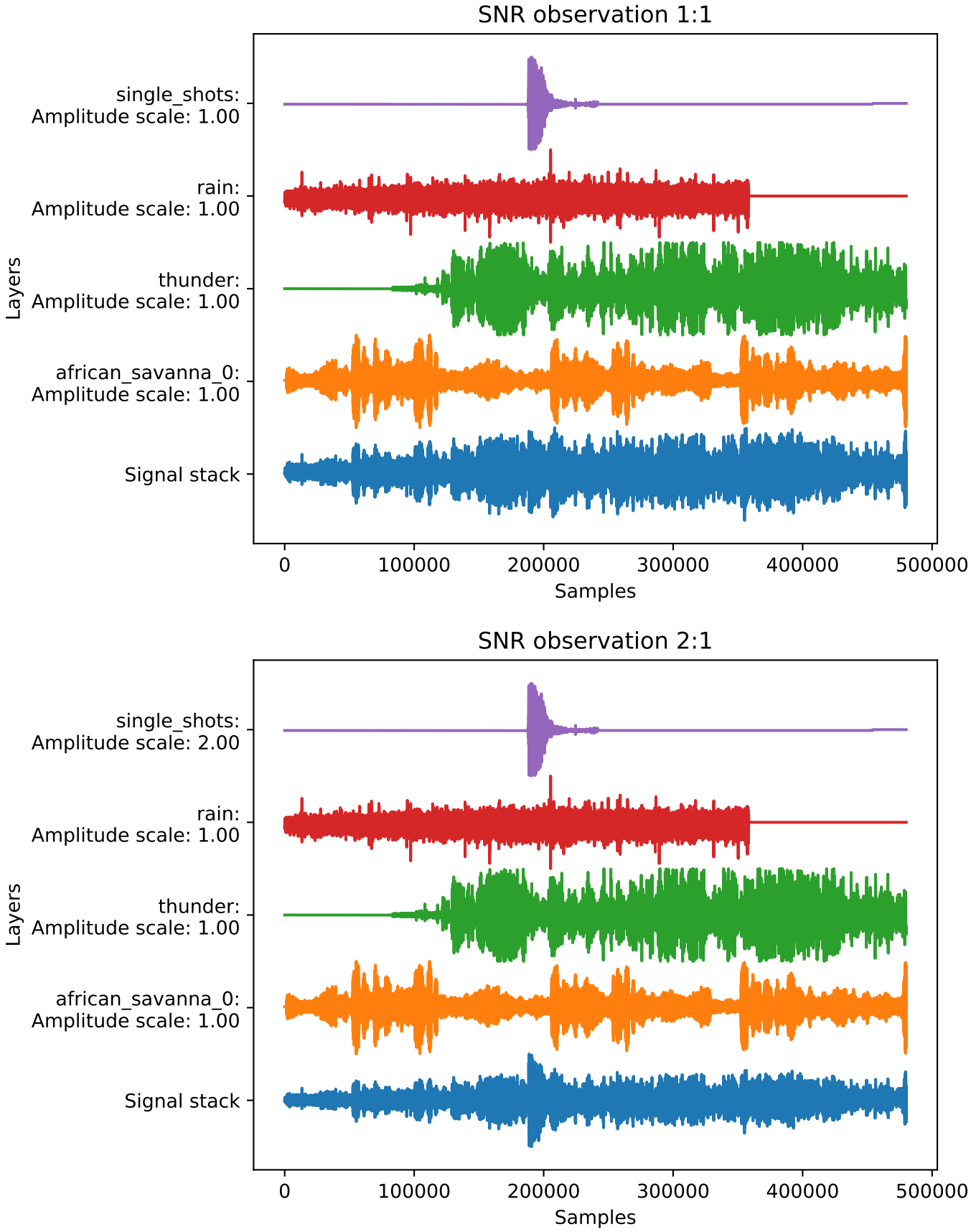


Figure 5.5 – Signal stacks of a random sample for SNR levels of 1 and 2

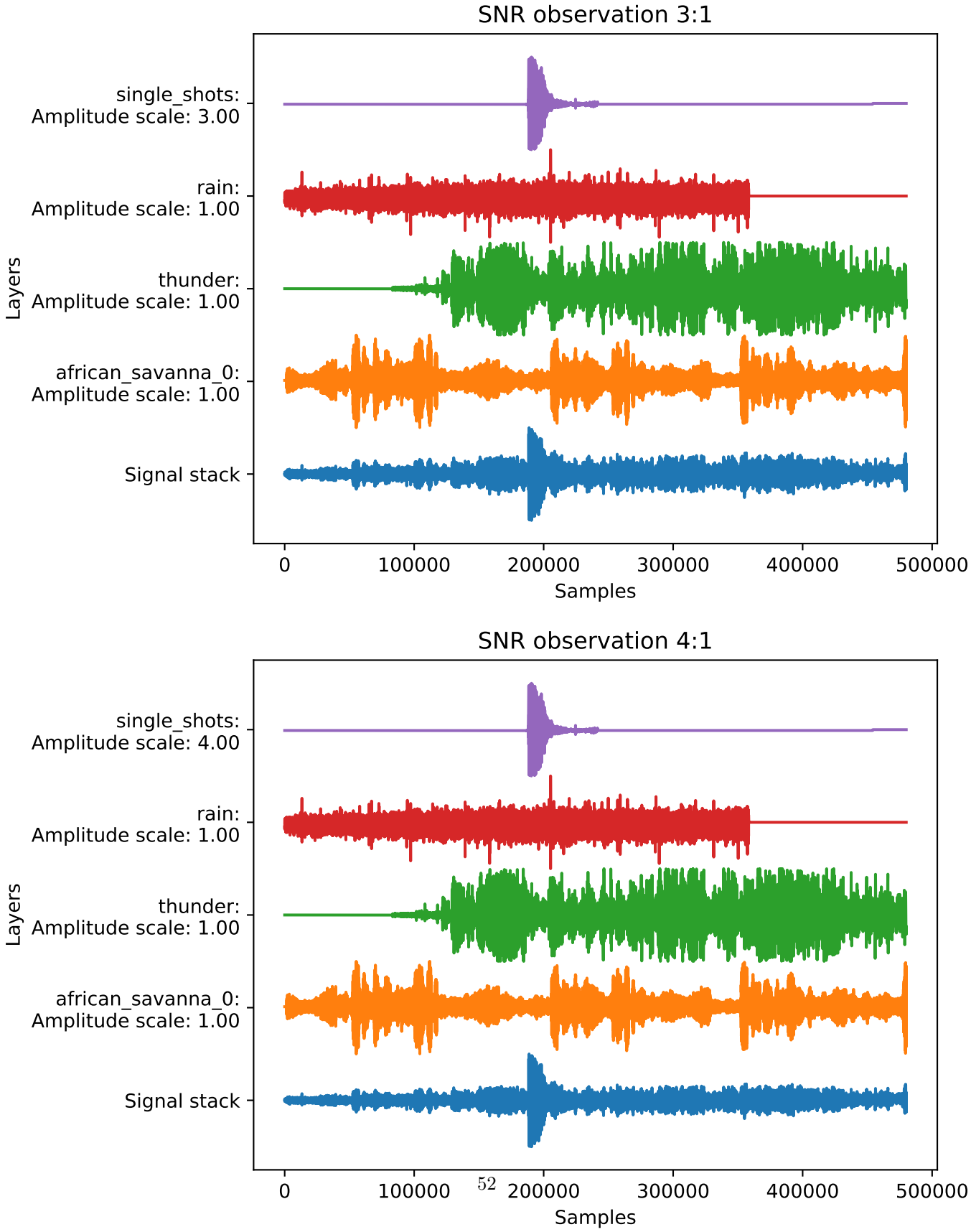


Figure 5.6 – Signal stacks of a random sample for SNR levels of 3 and 4

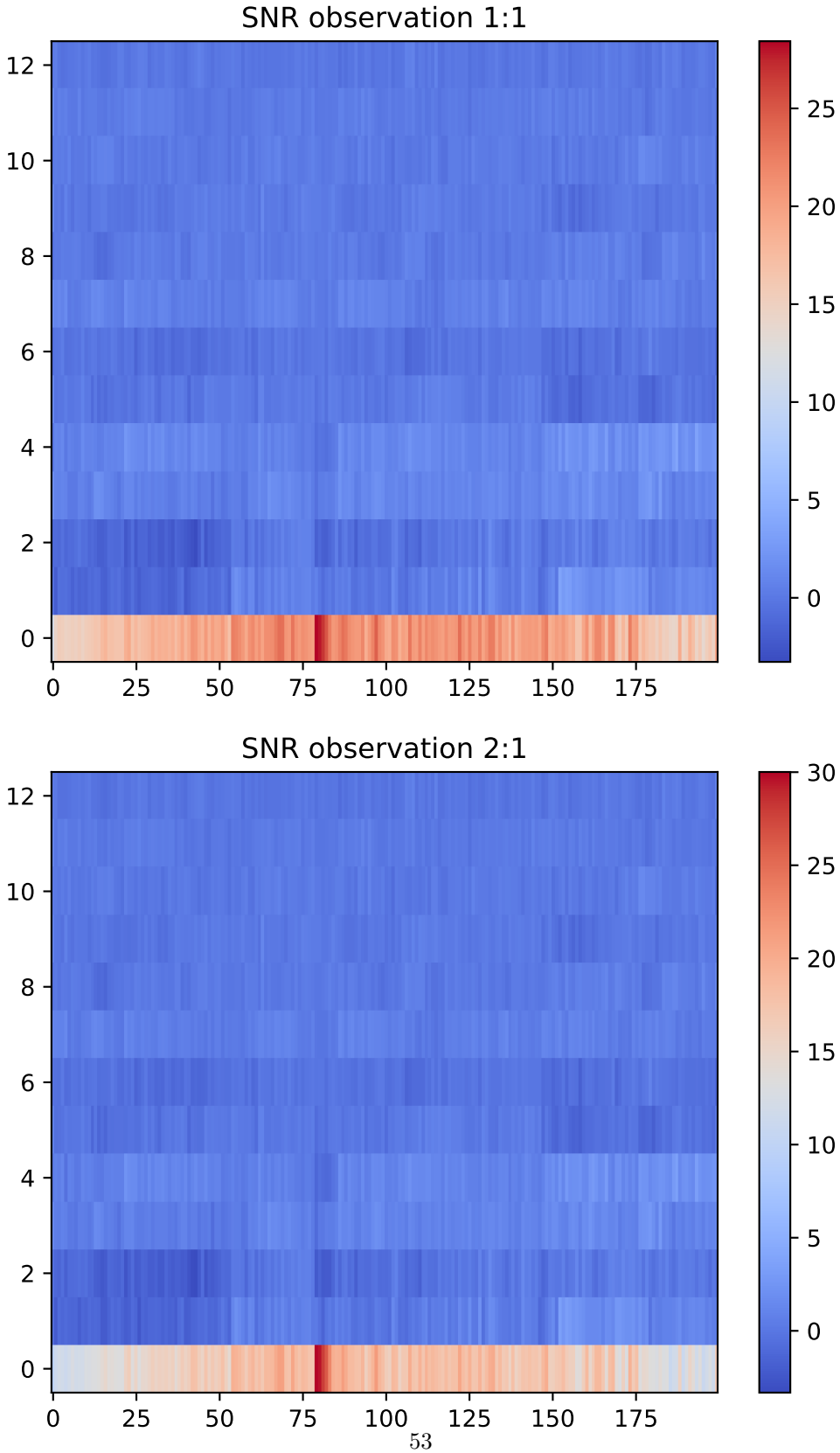


Figure 5.7 – Extracted MFCC features of the sample presented in Figure 5.5 for SNR levels of 1 and 2.

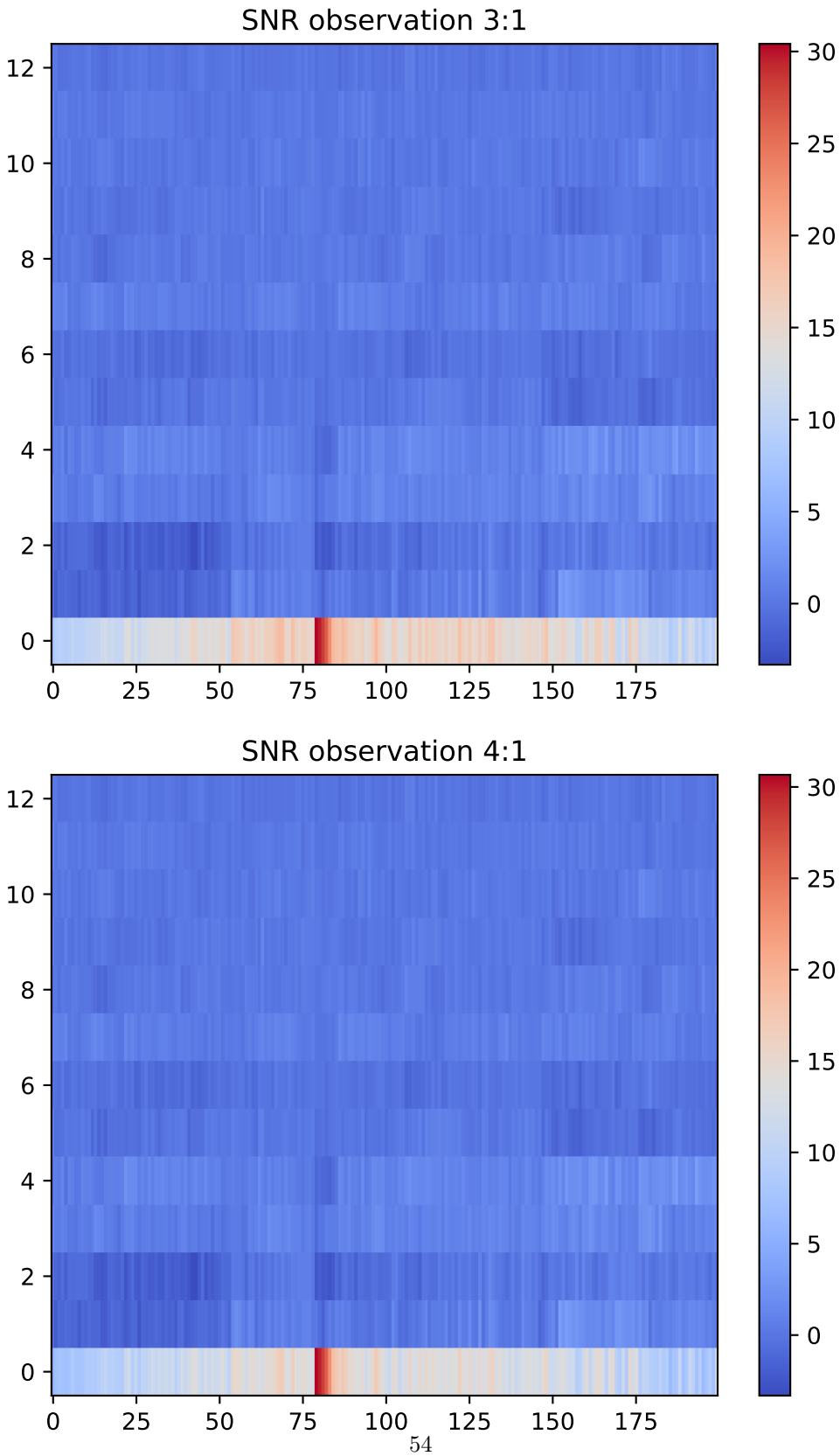


Figure 5.8 – Extracted MFCC features of the sample presented in Figure 5.6 for SNR levels of 3 and 4.

5.2.1 SNR Effect on the Model Accuracy

To explore the effect of SNR on the model accuracy, a 2-dimensional CNN model with fixed initial parameters is used. As mentioned in Section 4.2.1, the data used for this experiment consist of Savanna ambient and gunshot sounds such as in Table 4.1. Two hundred samples are selected from each category, yielding $200 \times 2^2 = 800$ samples per dataset. The model is trained for five separate datasets, each with a sample size of 800.

Figure 4.2 and Table 4.2 present the CNN model architecture and parameters used for the experiment. The model consist of two convolution layers, and two fully connected dense layers. The performance of each trained model is then evaluated on its test set. Table 5.3 shows the results of these evaluation. As predicted, the SNR and the model accuracy are *positively correlated*. The model accuracy in detecting gunshots is better when the SNR is higher, or in other words, the gunshot sound is louder. Considering the correlation between the distance and the SNR investigated in Section 3.2, the obtained results indicate that the model accuracy is also correlated with the shorter distance.

Dataset	**Confusion Matrix**	**Accuracy**
$SNR_{1:1}$	$\begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}$	0.85
$SNR_{2:1}$	$\begin{bmatrix} 0.96 & 0.04 \\ 0.06 & 0.94 \end{bmatrix}$	0.95
$SNR_{3:1}$	$\begin{bmatrix} 0.98 & 0.02 \\ 0.03 & 0.97 \end{bmatrix}$	0.97
$SNR_{4:1}$	$\begin{bmatrix} 0.99 & 0.01 \\ 0.02 & 0.98 \end{bmatrix}$	0.98
$SNR_{5:1}$	$\begin{bmatrix} 0.98 & 0.02 \\ 0 & 1 \end{bmatrix}$	0.99

Table 5.3 – Confusion matrix and accuracy of CNN model trained for five SNRs. There are **two** sound categories present in the samples, Savanna and gunshots

Dataset	**Confusion Matrix**	**Accuracy**
$SNR_{1:1}$	$\begin{bmatrix} 0.45 & 0.55 \\ 0.52 & 0.48 \end{bmatrix}$	0.46
$SNR_{2:1}$	$\begin{bmatrix} 0.83 & 0.17 \\ 0.21 & 0.79 \end{bmatrix}$	0.81
$SNR_{3:1}$	$\begin{bmatrix} 0.93 & 0.07 \\ 0.07 & 0.93 \end{bmatrix}$	0.93
$SNR_{4:1}$	$\begin{bmatrix} 0.99 & 0.01 \\ 0.04 & 0.96 \end{bmatrix}$	0.97
$SNR_{5:1}$	$\begin{bmatrix} 0.99 & 0.01 \\ 0.01 & 0.99 \end{bmatrix}$	0.99

Table 5.4 – Confusion matrix and accuracy of CNN model trained for five SNRs. All **four** sound categories are present in the samples, Savanna and gunshots, rain and thunder.

5.2.2 Rain and Thunder Effect

This experiment investigates thunder and rain effects on gunshot recognition accuracy. The experiment is done in two parts as explained in Section 4.2.2.

Part One

In the first part, the effects of rain and thunder are investigated separately. To investigate the effect of rain, both Savanna and rain sounds are present in all the samples, and there is a 50% chance of gunshot sound occurring. Similarly,

only Savanna, thunder, and gunshot sounds are selected to investigate the effect of thunderstorms.

These configurations are presented in Table 4.4. The sample size of each dataset is 1600. Figure 4.2 and Table 4.2 present the CNN model architecture and parameters.

Table 5.5 shows the accuracy achieved for each datasets. The first row of both Tables are repeated from Table 5.3 for comparison. Comparing the results shows that weather condition significantly affects acoustic gunshot recognition. This indicates that on rainy days or during thunderstorms if the shooter is far from the acoustic sensor, the probability of positively detecting gunshot sound (TP) is drastically lower.

If the rainfall or thunderstorm is louder than the assumed Savanna ambient sounds, the acoustic sensor hearing range is narrowed, and the gunshot sound level must be louder than the background noise. This means that if the shooter distance is far enough that the SNR value drops below 1, the gunshot signal becomes unusable; hence, the model performance will be limited to a closer range.

Additionally, false-positive and false-negative predictions considerably increase during thunderstorms and rainfalls. Therefore, if deployed on the end-nodes, this model would only be reliable for closer ranges during these weather conditions. If this problem is overlooked, many poaching activities might be missed, or the false alarms would result in unwanted trips to the field for the rangers.

Dataset	**Confusion Matrix**	**Accuracy**	**Dataset**	**Confusion Matrix**	**Accuracy**
No rain, thunder	$\begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}$	0.85	*No rain, thunder*	$\begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}$	0.85
$Rain_{1:1}$	$\begin{bmatrix} 0.42 & 0.58 \\ 0.55 & 0.45 \end{bmatrix}$	0.43	$Thunder_{1:1}$	$\begin{bmatrix} 0.47 & 0.53 \\ 0.43 & 0.57 \end{bmatrix}$	0.52
$Rain_{1:2}$	$\begin{bmatrix} 0.74 & 0.26 \\ 0.28 & 0.72 \end{bmatrix}$	0.73	$Thunder_{1:2}$	$\begin{bmatrix} 0.82 & 0.18 \\ 0.25 & 0.75 \end{bmatrix}$	0.78

Table 5.5 – Rain and Thunder Effect on gunshot detection accuracy. in $Rain/Thunder_{i:j}$, i and j are the relative amplitude ratio of rain/thunder and gunshot sound respectively.

Part Two

In the second part of this experiment, both rain and thunder can happen within a sample. The signal level configurations are the same as presented in Table 4.4. The Savanna ambient sounds are always present, and the other three categories have a 50% chance of happening within the samples. The sample size of each dataset is 1600. Same as the former experiment the CNN model architecture and parameters used for this experiment is as Figure 4.2 and Table 4.2.

Table 5.4 shows the confusion matrix and accuracy achieved for each trained model. Comparing the results with Table 5.3, once more prove that weather conditions affect the acoustic gunshot recognition, especially for smaller SNR

values. However, this effect is negligible for higher SNRs, meaning if shooter distance is close enough so that $SNR > 5$, gunshot detection is assured.

Based on the assumptions made for sound levels in Section 3.2, if the background noise level is 60 dB, $SNR \geq 5$ would translate to a gunshot sound pressure more than 74 dB. According to Table 3.2, if the near-gun sound pressure would be 150 dB in an open area, 74 dB sound pressure would happen if the shooter is approximately 2 km from the acoustic sensor. However, this distance might be smaller due to the influence of humidity, temperature, and environmental structures on the sound waves.

Finally, it should be noted that these results are based on our initial CNN model. It might be the case that the model performance increase when hyperparameter optimization is used for the CNN model. This claim is investigated closely in Section 5.2.4.

5.2.3 Sample Size Alteration

This experiment aims to compare the model accuracy for different sample sizes. The data used for the experiment contains all four sound categories, such as in Table 4.4. The CNN model introduced in Figure 4.2 is used for this experiment. The model is trained for Dataset $SNR_{3:1}$ for four sample sizes. Model performance is evaluated on a fixed test set. The results are presented in Table 5.6. As expected, the model accuracy increases with the sample size. On the other hand, training time increases with this growth. This indicates that a trade-off can be achieved between the sample size and the accuracy.

Sample Size	**Confusion Matrix**	**Accuracy**
320	$\begin{bmatrix} 0.83 & 0.17 \\ 0.8 & 0.92 \end{bmatrix}$	0.87
800	$\begin{bmatrix} 0.96 & 0.04 \\ 0.11 & 0.89 \end{bmatrix}$	0.92
1600	$\begin{bmatrix} 0.93 & 0.07 \\ 0.07 & 0.93 \end{bmatrix}$	0.93
3200	$\begin{bmatrix} 0.96 & 0.08 \\ 0.04 & 0.92 \end{bmatrix}$	0.94

Table 5.6 – The effect of sample size on model accuracy for dataset $SNR_{3:1}$

5.2.4 Hyperparameter Tuning for CNN Model

This experiment aims to increase the CNN model accuracy using hyperparameter tuning for the CNN model. The sample size is 1600. The search space for the hyperparameters is presented in Table 5.7, and the hyperparameters optimization setting are presented in Table 4.5. Note that the training is done for each dataset separately, hence the tuning process. As a result, the hyperparameters might vary from one model to another.

The results achieved from this experiment demonstrate a nonlinear increase in the model accuracy. For $SNR = 1$ this increase is 25%, which drops bellow 5% for $SNR \geq 2$. This shows that when the signal-to-noise ratio is higher, the significance of the hyperparameter tuning lessens.

Dataset	Confusion Matrix	Accuracy	Initial CNN Accuracy
$SNR_{1:1}$	$\begin{bmatrix} 0.53 & 0.47 \\ 0.28 & 0.72 \end{bmatrix}$	0.62	0.46
$SNR_{2:1}$	$\begin{bmatrix} 0.93 & 0.07 \\ 0.21 & 0.79 \end{bmatrix}$	0.86	0.81
$SNR_{3:1}$	$\begin{bmatrix} 0.92 & 0.08 \\ 0.02 & 0.98 \end{bmatrix}$	0.95	0.93
$SNR_{4:1}$	$\begin{bmatrix} 0.98 & 0.02 \\ 0.01 & 0.99 \end{bmatrix}$	0.98	0.97
$SNR_{5:1}$	$\begin{bmatrix} 0.99 & 0.01 \\ 0.01 & 0.99 \end{bmatrix}$	0.99	0.99

Table 5.7 – Confusion matrix and accuracy of CNN models with hyperparameter optimization trained for five SNRs.

Additionally, FN and FP predictions are observed for each SNR dataset. Tables 5.8 and 5.9 present the percentage of each category causing these errors. Table 5.8 shows that for lower SNRs, all three categories can cause false-negative predictions. This error, however, is mostly due to the presence of rain. For higher SNRs, though, only two categories of rain and thunder are responsible for the wrong predictions.

Table 5.9, on the other hand, reveals that most of the false-positive predictions are due to the presence of thunder. Another interesting fact that this analysis reveals is that certain bird calls and insect sounds cause wrong predictions. Figures 5.10 and 5.9 present some of the falsely predicted samples that include the mentioned animal sounds.

SNR datasets	Total number of FN predictions	only Savanna	Rain	Thunder
$SNR_{1:1}$	55	30%	51%	37%
$SNR_{2:1}$	41	18%	57%	51%
$SNR_{3:1}$	4	0	100%	100%
$SNR_{4:1}$	2	0	100%	100%
$SNR_{5:1}$	3	0	66%	100%

Table 5.8 – Total number of FN predictions from test sets of each SNR dataset. Additionally, the wrong predictions are further investigated to observe which categories were present in the samples.

GPU Vs. CPU

This experiment compares the effect of the used hardware on the accuracy of the CNN model. The data used for this experiment contains all four sound

SNR datasets	Total number of FP predictions	only Savanna	Rain	Thunder
$SNR_{1:1}$	95	24%	42%	55%
$SNR_{2:1}$	15	33%	14%	54%
$SNR_{3:1}$	17	30%	18%	60%
$SNR_{4:1}$	4	25%	0	75%
$SNR_{5:1}$	2	100%	0	0

Table 5.9 – Total number of FP predictions from test sets of each SNR dataset. Additionally, the wrong predictions are further investigated to observe which categories were present in the samples.

categories, such as in Table 4.4. One hundred files are selected from each sound category, so the sample size is 1600. The CNN model is trained for five SNR datasets once using CPU and GPU. Hyperparameter tuning is used for CNN model optimization in both scenarios.

Table 5.10 shows the accuracy achieved using both hardware for SNR datasets. The model accuracy remains almost the same. The slight accuracy deviation observed is suspected to be due to the indeterministic nature of GPU when used for model training.

The advantage of using GPU for model training is the shortened training time. This process, though, does not affect the evaluation phase. This means that if deployed on the end-nodes, the model accuracy will be independent of the end-node hardware.

Dataset	Accuracy	
	GPU	CPU
$SNR_{1:1}$	0.62	0.65
$SNR_{2:1}$	0.86	0.85
$SNR_{3:1}$	0.95	0.95
$SNR_{4:1}$	0.98	0.98
$SNR_{5:1}$	0.99	0.99

Table 5.10 – Comparing the accuracy achieved for CNN model using CPU vs. GPU for five SNR datasets

5.2.5 Feature Extraction Effect: Raw Waveform Vs. MFCC

This experiment focuses on the input type of the model. Raw waveforms are used instead of extracting and using MFCCs. Feature extraction is, therefore, entirely left to the convolution layers of the model. This type of data only contains amplitude values in the time domain and is called one-dimensional data. Therefore, the CNN model chosen for the training is of type one-dimensional. The datasets used for this experiment contain all four sound categories. The sample size is 1600. Hyperparameter tuning is used to find the best fitting. The results of the experiment are presented in Table 5.11.

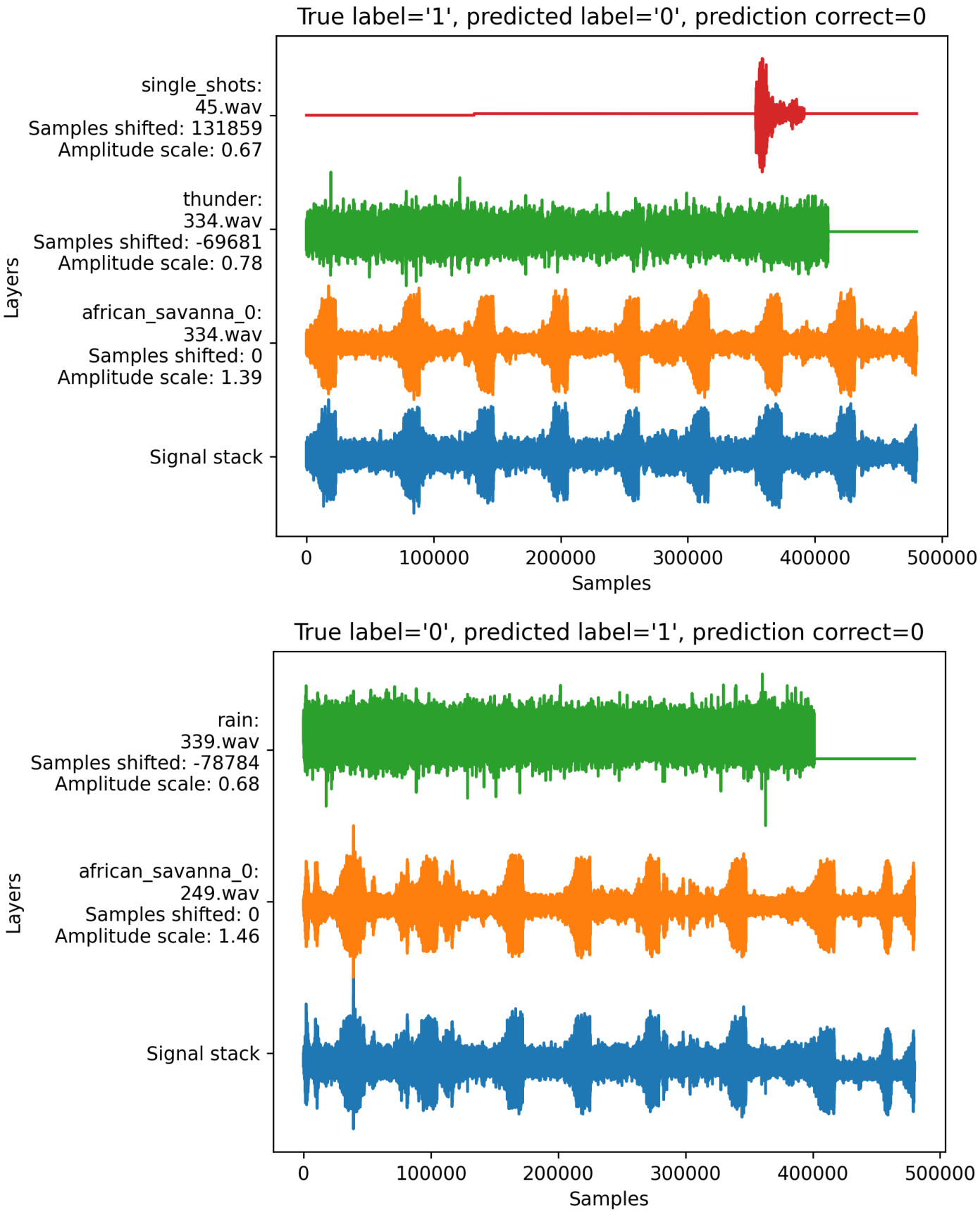

True label='1', predicted label='0', prediction correct=0
single_shots:
45.wav
Samples shifted: 131859
Amplitude scale: 0.67
thunder:
334.wav
Samples shifted: -69681
Amplitude scale: 0.78
african_savanna_0:
334.wav
Samples shifted: 0
Amplitude scale: 1.39
Signal stack
Layers
0
100000
200000
300000
400000
500000
Samples
True label='0', predicted label='1', prediction correct=0
rain:
339.wav
Samples shifted: -78784
Amplitude scale: 0.68
african_savanna_0:
249.wav
Samples shifted: 0
Amplitude scale: 1.46
Signal stack
Layers
0
100000
200000
300000
400000
500000
Samples

Figure 5.9

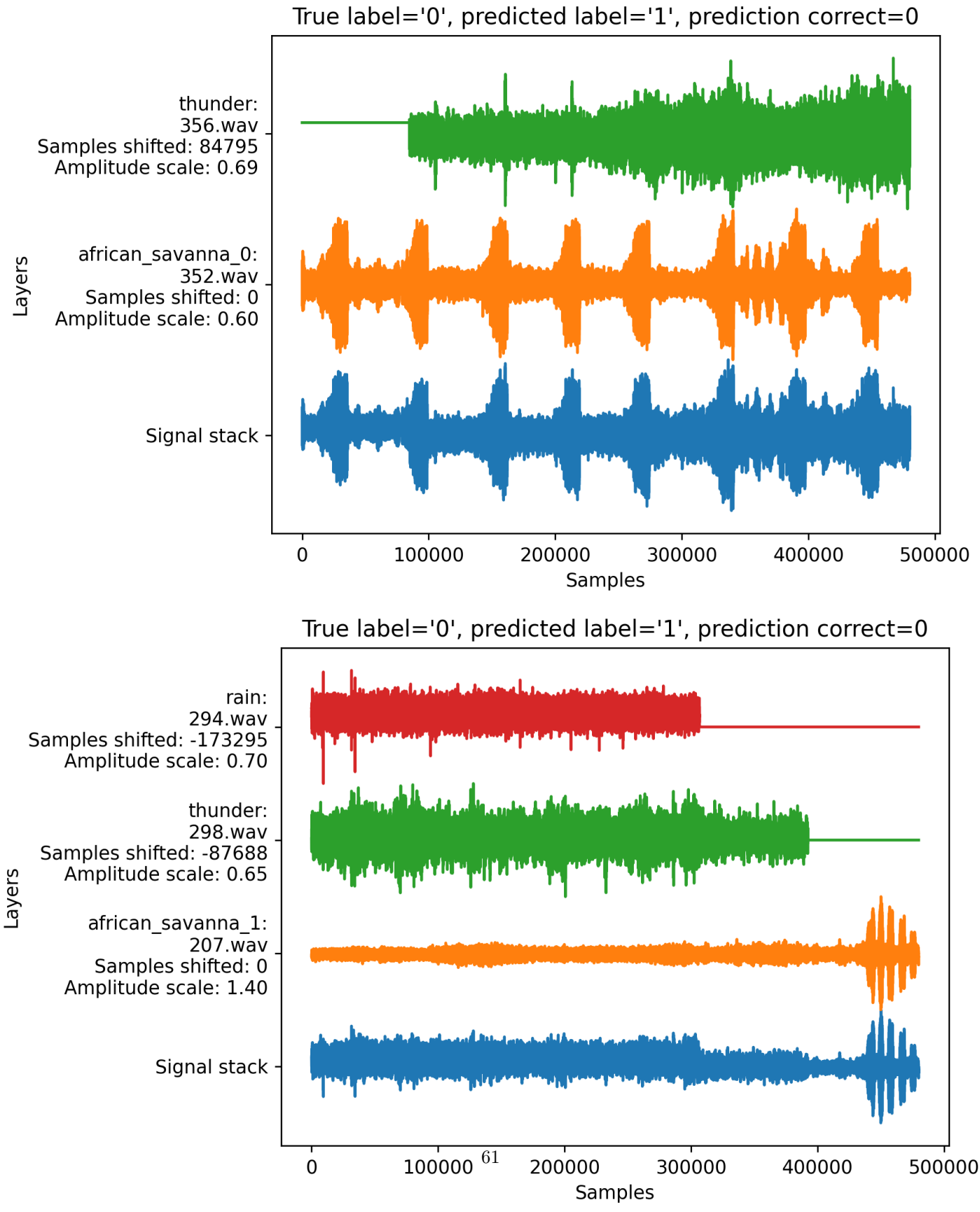

True label='0', predicted label='1', prediction correct=0
thunder:
356.wav
Samples shifted: 84795
Amplitude scale: 0.69
african_savanna_0:
352.wav
Samples shifted: 0
Amplitude scale: 0.60
Signal stack
Layers
0
100000
200000
300000
400000
500000
Samples
True label='0', predicted label='1', prediction correct=0
rain:
294.wav
Samples shifted: -173295
Amplitude scale: 0.70
thunder:
298.wav
Samples shifted: -87688
Amplitude scale: 0.65
african_savanna_1:
207.wav
Samples shifted: 0
Amplitude scale: 1.40
Signal stack
Layers
0
100000
200000
300000
400000
500000
Samples

Figure 5.10

Comparing the results presented in Table 5.11 to the results of Experiment 5.2.4 presented in Table 5.7 reveals that the accuracy is almost the same for both feature extraction methods. More over, other than the longer training time when using raw waveforms, loading and classifying samples are four times slower in the model evaluation phase. This shows that when the computational power is limited and working in real-time is a crucial factor, MFCC feature extraction is a better judgment.

However, note that if the signals were not downsampled as explained in Section 4.2.5, the one-dimensional CNN model might have been more accurate. It might be the case that crucial information about the signal is lost when the sample rate decreases.

Dataset	Confusion Matrix	Accuracy	MFCC Accuracy
$SNR_{1:1}$	$\begin{bmatrix} 0.52 & 0.48 \\ 0.29 & 0.71 \end{bmatrix}$	0.61	0.62
$SNR_{2:1}$	$\begin{bmatrix} 0.85 & 0.15 \\ 0.19 & 0.81 \end{bmatrix}$	0.83	0.86
$SNR_{3:1}$	$\begin{bmatrix} 0.87 & 0.13 \\ 0.03 & 0.97 \end{bmatrix}$	0.92	0.95
$SNR_{4:1}$	$\begin{bmatrix} 0.94 & 0.06 \\ 0.03 & 0.97 \end{bmatrix}$	0.95	0.98
$SNR_{5:1}$	$\begin{bmatrix} 0.95 & 0.05 \\ 0.01 & 0.99 \end{bmatrix}$	0.97	0.99

Table 5.11 – Confusion matrix and accuracy of 1D-CNN models trained for five SNRs. The input data is raw waveforms.

5.2.6 Combining SNR datasets

This experiment is designed to extend the training data to all the SNR datasets. For simplicity, the initial CNN model is used. The results are then compared to the Table 5.4 from experiment 5.2.2. The SNR datasets used for this experiment are presented in Table 4.4. The data consist of all four sound categories. Savanna sounds are always present within the samples, with half of them being the daytime sounds and the other half nighttime sounds. The other three categories, rain, thunder, and gunshot sounds, each have an independent 50% chance of occurring.

SNR datasets are inserted into the training data in four steps. Starting from the dataset $SNR_{5:1}$, dataset $SNR_{4:1}$ is added on top of the training data. The accuracy of the newly fit model is then compared to the accuracy of the model trained only for dataset $SNR_{4:1}$. This process is repeated for datasets $SNR_{3:1}$, $SNR_{2:1}$, and $SNR_{1:1}$ as explained in Section 4.2.

This experiment is done in two parts:

Part One

First, the total sample size is fixed at 1600. For instance, 800 samples from both datasets $SNR_{5:1}$, and $SNR_{4:1}$, are selected for model training in the first step, so the total size is 1600. Table 5.12 shows the sample size per SNR dataset in every step. As presented in this Table, the model accuracy for the targeted SNR decreases in every step, except for $SNR_{1:1}$, which is a modest improvement. This indicates that introducing SNR diversity at the cost of keeping the training size constant has no positive influence on classification accuracy.

Sample Size per Dataset	**Present SNR Datasets**	**Target SNR**	**Model Accuracy**	**Initial Accuracy**
800	5:1, 4:1	4:1	0.93	0.97
533	5:1, 4:1, 3:1	3:1	0.69	0.93
400	5:1, 4:1, 3:1, 2:1	2:1	0.57	0.81
320	5:1, 4:1, 3:1, 2:1, 1:1	1:1	0.52	0.46

Table 5.12 – CNN model accuracy for a target SNR. Total sample size is fixed at 1600. SNR datasets are combined together in four steps.

Part Two

In the second part, however, the sample size per SNR dataset is constant; hence, the total sample size increases in every step. For instance, in the first step, 1600 samples from both datasets $SNR_{5:1}$, and $SNR_{4:1}$, are selected for model training, so the total size is 3200. Table 5.13 shows the total sample size in every step. As presented in this Table, the model accuracy for the targeted SNR increases in every step. This improvement is more significant for smaller SNRs. Unlike the first part of this experiment, this indicates that introducing SNR diversity and increasing the sample size enhances the gunshot classification accuracy remarkably.

Total Sample Size	**Present SNR Datasets**	**Target SNR**	**Model Accuracy**	**Initial Accuracy**
3200	5:1, 4:1	$SNR_{4:1}$	0.98	0.97
4800	5:1, 4:1, 3:1	$SNR_{3:1}$	0.98	0.93
6400	5:1, 4:1, 3:1, 2:1	$SNR_{2:1}$	0.91	0.81
8000	5:1, 4:1, 3:1, 2:1, 1:1	$SNR_{1:1}$	0.80	0.46

Table 5.13 – CNN model accuracy for a target SNR. The sample size of every dataset is fixed at 1600. Hence, total training data size increases in every step.

5.2.7 Final Proposition

After conducting various experiments and observing the effects of different parameters on the gunshot classification accuracy, we propose a single model for

gunshot detection in African wildlife.

Since the model should perform in different noise ratios, all the SNR datasets are combined in the training data. Additionally, experiment 5.2.2 showed that the weather conditions affect the model accuracy; hence, the same configuration used for the SNR datasets is repeated for this part. Table 5.14 presents this configuration again. The Savanna ambient sounds are always present, and the other three categories have a 50% chance of happening within the samples.

Dataset	**Relative Amplitude Ratio**			
	Gunshot	Savanna ambient	Rain	Thunder
$SNR_{1:1}$	1	1	1	1
$SNR_{2:1}$	2	1	1	1
$SNR_{3:1}$	3	1	1	1
$SNR_{4:1}$	4	1	1	1
$SNR_{5:1}$	5	1	1	1

Table 5.14 – Five SNR datasets that are combined together for training the final model.

Based on the results from experiment 5.2.3, 200 files are selected from each category. This yields $200 \times 2^4 = 3200$ samples per SNR dataset and a total sample size of $5 \times 3200 = 160k$. Subsequently, based on the promising results obtained in experiment 5.2.4, hyperparameter optimization is used for CNN model training. The model is trained on GPU to accelerate the training time. Finally, the MFCC features are chosen for model input, based on the comparison done in experiment 5.2.5.

SNR	**Model Accuracy in Different Weathers**			
	Clear Sky	Rain	Thunderstorm	Rain/Thunder
1:1	0.85	0.73	0.75	0.67
2:1	0.97	0.96	0.93	0.89
3:1	0.98	0.97	0.98	0.97
4:1	0.99	0.99	0.99	0.98
5:1	0.99	0.99	0.99	0.98

Table 5.15 – The accuracy of the final model proposed for this study for five SNR values and four weather condition

Table 5.16 presents the CNN model parameters chosen by the oracle. The model has three convolutional and five dense layers. To evaluate the model, a new test set is built. As stated in table 5.1 from Section 5.1, the data gathered for this study resulted in approximately 260 files per sound category. Since 200 files from each category are selected for training, 60 files per category are remained to synthesize a test set. To evaluate the model accuracy for each SNR and each weather condition separately, the data is synthesized in 20 different configurations, five SNR, four different weather conditions.

Once again, the Savanna sounds are present in all the test set samples. In *Clear Sky*, the background noise is only Savanna ambient sounds. In the *Rain* set, all samples have the rain sounds as well. Similarly, the samples from *Thunderstorm* set have Savanna ambient and thunder sounds in the background.

Model Layers	Parameters	Model Layers	Parameters
Convolution layer 1	Convolutional filters: 40 Kernel size: [3, 3] Strides: [1, 1] Activation function: ReLU	Pooling layer 1	MaxPooling2D Pool size: [2, 2] Strides: [2, 2]
Convolution layer 2	Convolutional filters: 28 Kernel size: [4, 3] Strides: [1, 1] Activation function: Sigmoid	Pooling layer 2	MaxPooling2D Pool size: [2, 2] Strides: [2, 2]
Convolution layer 3	Convolutional filters: 24 Kernel size: [4, 3] Strides: [1, 1] Activation function: ReLU	Pooling layer 3	MaxPooling2D Pool size: [2, 2] Strides: [2, 2]
Dense layer 1	Units: 200 Activation: Sigmoid		
Dense layer 2	Units: 100 Activation: tanh		
Dense layer 3	Units: 100 Activation: tanh		
Dense layer 4	Units: 300 Activation: ReLU		
Dense layer 5	Units: 1 Activation: Sigmoid		

Table 5.16 – Hyperparameters chosen for CNN model by the oracle

Finally, the *Rain/Thunder* set has all three sound categories in the background. In every set, there is a 50% chance for gunshot sounds. Table 5.15 presents the evaluation results on the test set. The results show that the model is nearly perfect for $SNR > 2$. However, the model accuracy drops significantly when both rain and thunder sounds are present within the samples. To provide more details about the predictions, Table 5.21 presents the confusion matrix of each test category. These Tables show that the highest rate of wrong predictions are for $SNR = 1:1$, and these predictions are mostly false-negatives. This outcome, however, is not unexpected; $SNR = 1$ means that the gunshot signal amplitude is not louder than the background noise. Additionally, as explained in Section 5.1, the supersonic frequencies are unfortunately absent from the gunshot signals. Hence, it would be more difficult for the model to distinguish the gunshots.

Another insight observed from Tables 5.21 is the low rate of false-positive predictions, which is an advantage itself since it almost eliminates the chance of false alarms. On the other hand, the FN predictions drop to zero for $SNR > 3$, except when rain and thunderstorms happen simultaneously. This means that the model never misses the gunshot sounds for SNR values larger than 3.

Finally, using the SNR-distance mapping from Table 3.3 and the obtained model accuracy for different SNRs, the correlation between the shooter distance and the gunshot detection accuracy is depicted in Figure 5.11. This figure is

based on the assumptions that the background noise level is constantly 60 dB, regardless of the weather condition, and that the Gunshot sound level is 150 dB near the gun ($< 1m$). While the model accuracy looks promising for the shooter distances up to a few kilometers from the acoustic sensor ($< 6km$), it should be noted that the actual values are suspected to be lower when deployed on the end-nodes. This, of course, is due to the parameters left out of this research study, such as geographical conditions, signal degradation, humidity, temperature, and acoustic sensor characteristics.

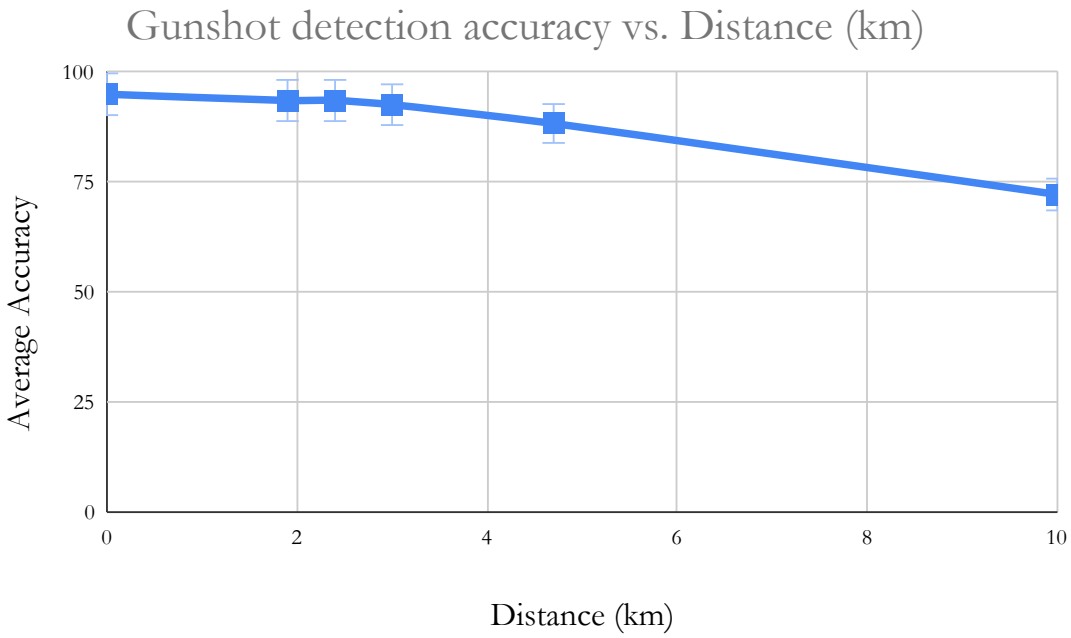


Figure 5.11

SNR	Confusion Matrix	Accuracy
1:1	$\begin{bmatrix} 0.98 & 0.02 \\ 0.28 & 0.72 \end{bmatrix}$	0.85
2:1	$\begin{bmatrix} 0.98 & 0.02 \\ 0.03 & 0.97 \end{bmatrix}$	0.97
3:1	$\begin{bmatrix} 0.98 & 0.02 \\ 0.02 & 0.98 \end{bmatrix}$	0.98
4:1	$\begin{bmatrix} 0.98 & 0.02 \\ 0 & 1 \end{bmatrix}$	0.99
5:1	$\begin{bmatrix} 0.98 & 0.02 \\ 0 & 1 \end{bmatrix}$	0.99

Table 5.17 – Clear Sky

SNR	Confusion Matrix	Accuracy
1:1	$\begin{bmatrix} 0.97 & 0.03 \\ 0.51 & 0.49 \end{bmatrix}$	0.73
2:1	$\begin{bmatrix} 0.97 & 0.03 \\ 0.05 & 0.95 \end{bmatrix}$	0.96
3:1	$\begin{bmatrix} 0.96 & 0.04 \\ 0.02 & 0.98 \end{bmatrix}$	0.97
4:1	$\begin{bmatrix} 0.98 & 0.02 \\ 0 & 1 \end{bmatrix}$	0.99
5:1	$\begin{bmatrix} 0.98 & 0.02 \\ 0 & 1 \end{bmatrix}$	0.99

Table 5.18 – Rain

SNR	Confusion Matrix	Accuracy
1:1	$\begin{bmatrix} 0.99 & 0.01 \\ 0.49 & 0.51 \end{bmatrix}$	0.75
2:1	$\begin{bmatrix} 0.98 & 0.02 \\ 0.11 & 0.89 \end{bmatrix}$	0.93
3:1	$\begin{bmatrix} 0.99 & 0.01 \\ 0.02 & 0.98 \end{bmatrix}$	0.98
4:1	$\begin{bmatrix} 0.98 & 0.02 \\ 0 & 1 \end{bmatrix}$	0.99
5:1	$\begin{bmatrix} 0.98 & 0.02 \\ 0 & 1 \end{bmatrix}$	0.99

Table 5.19 – Thunderstorm

SNR	Confusion Matrix	Accuracy
1:1	$\begin{bmatrix} 0.98 & 0.02 \\ 0.63 & 0.37 \end{bmatrix}$	0.67
2:1	$\begin{bmatrix} 0.97 & 0.03 \\ 0.19 & 0.81 \end{bmatrix}$	0.89
3:1	$\begin{bmatrix} 0.97 & 0.03 \\ 0.03 & 0.97 \end{bmatrix}$	0.97
4:1	$\begin{bmatrix} 0.97 & 0.03 \\ 0.01 & 0.99 \end{bmatrix}$	0.98
5:1	$\begin{bmatrix} 0.97 & 0.03 \\ 0.01 & 0.99 \end{bmatrix}$	0.98

Table 5.20 – Rain/Thunder

Table 5.21 – Confusion matrices for each test configuration presented in 5.15.

Chapter 6

Discussion, Conclusion and Recommendations

In this thesis, a deep learning model is proposed to detect gunshot acoustics in the Savanna wildlife. Existing solutions for intrusion detection mostly do not cover gunshot sounds. On the other hand, existing gunshot detection systems do not cover wildlife scenarios. We have created a dataset for this specific case study, including ambient sounds from African Savanna, rainfall, thunderstorms, and gunshot sounds. A CNN model is then trained to detect gunshot sounds. Two feature extraction methods are evaluated for the model input. We have also investigated the model accuracy deviation in different weather conditions and noise levels. Additionally, we have investigated the correlation between the shooter distance, signal-to-noise ratio, and model accuracy.

The experiments results confirm that the model accuracy is correlated with the SNR and hence, the distance. The model accuracy decreases when SNR is lower. If the shooter is far, the gunshot sound pressure reaching the acoustic sensor would be smaller, ergo, the SNR, which causes the model accuracy to decrease.

It is also revealed that the weather conditions such as rainfalls and thunderstorms have a non-negligible negative influence on gunshot detection accuracy. If the weather conditions result in louder noise than the assumed threshold noise level (in this case, Savanna ambient sounds), the detection range is narrowed. This result was already expected since the higher noise level would translate into smaller SNR, and if $SNR < 1$, the signal is unusable, and the model performance would be meaningless.

Comparing MFCCs and raw waveforms for CNN model input showed that both yield approximately the same accuracy. This comparison implies that if the computation complexity, power consumption, and real-time response are crucial aspects, using MFCCs would be more efficient.

Finding and modeling the correlation between the shooter (sound source) distance, SNR, and the gunshot detection accuracy is beneficial for many reasons, such as system reliability, end-node grid size, and project cost. For instance, to

implement this project on the field, the distance between the end-nodes must be determined beforehand. The acoustic surveillance system must cover the entire protected area. As shown in Figure 6.1, based on the grid-size (end-nodes distance), three conditions may occur. If the end-nodes are too far from one another, there will be uncovered areas, and the system will fail to recognize and report if the shooter is located within the blind spot. On the other hand, if the end-nodes are too close to each other, the whole field will be covered; however, the number of nodes will increase, which aside from extra cost, may introduce more complexity to the system. This is where finding the correlation between model accuracy and the shooter distance becomes useful.

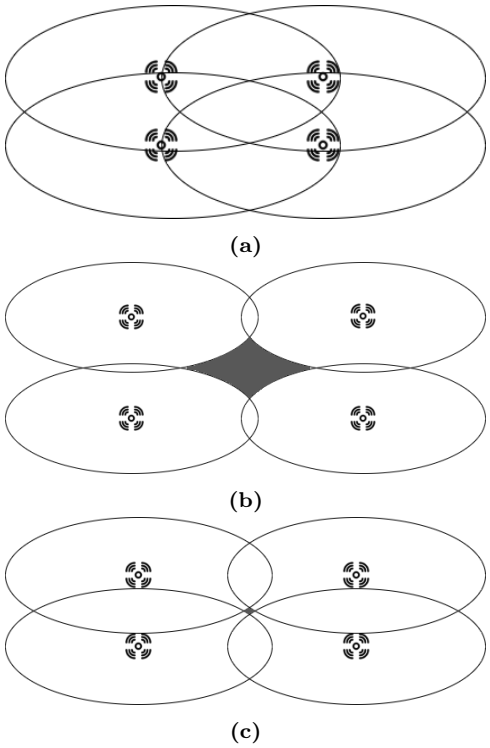


Figure 6.1 – Three conditions for end-node distances. (a) Close distance results in a large overlap area. (b) Far distance results in an uncovered area (c) The optimal distance of the end-nodes

6.1 Limitations

Like most research studies, this thesis has some limitations due to the overall research design. This Section presents these limitations to provide a more accurate picture of what can and cannot be concluded from this work.

As explained repeatedly in this thesis, a suitable dataset for this specific case study was not publicly available. Hence, the raw data is gathered from different sources, and new samples are synthesized with the purpose of a real-life simulation. In other words, created samples and imitated noise levels are done

on the signals using mathematical calculations, with some level of randomness in signal amplitudes and time of occurrence. Hence, it might be the case that the weather conditions and noise level imitations diverge from reality. For instance, in rainy weather, the wildlife activity may retrieve from the usual amount, and hence, the ambient sound level might be lower than the assumed amount. Plus, it might be the case that the gunshot sound disrupts the wildlife, and the ambient sound becomes louder or change abruptly, i.e., birdcall increase.

On the other hand, the frequency analysis of the raw tracks indicates that the magnitude of frequencies above 10 kHz is significantly low. Even though this is acceptable for background sounds, it is lost information in a gunshot signal. As already mentioned in Section 5.1, the presence of a shockwave in gunshot signal, and therefore the higher frequencies, beside the gun type, and the firing angle, depends on the recording microphone characteristics.

A microphone's *Frequency Response Line[91]* shows the sensitivity of a microphone to various frequencies. For instance, Figure 6.2, shows *Shure SM57* microphone's frequency-specific sensitivity. According to this Figure, *Shure SM57* sensitivity is degraded compared to its average line from 40 Hz to 200 HZ, and around 15 kHz. This indicates that the ultrasound frequencies in the gunshot sounds recorded with this microphone will have a negligible magnitude.

On the other hand, the frequency response line of *Earthworks M50* presented in Figure 6.3 looks almost steady for all the frequencies from 5 Hz to 50 kHz. This implies that the higher frequencies will become apparent if gunshot sounds are recorded with this type of microphone.

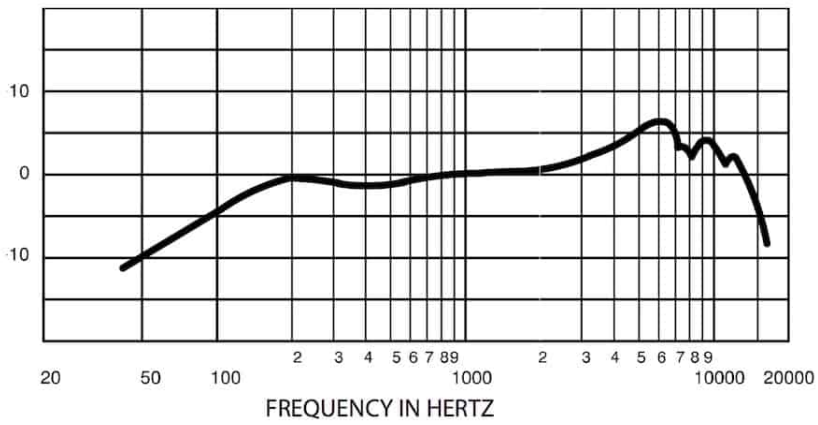


Figure 6.2 – Shure SM57 Frequency Response Graph [91]. The Y-axis is the relative sensitivity in decibels (dB)

Another limitation worth mentioning is the absence of details in the model output. Gunshot detection in wildlife is treated as a binary classification in this study. Hence, the classification algorithm does not provide details about the number of shots or the timestamp of the event within the ten-second frame, which may raise concerns about the true-positive classifications. For instance, if a gunshot and other similar sounds are present within a sample, and the prediction is true, it is unknown if the prediction is true for the right reason.

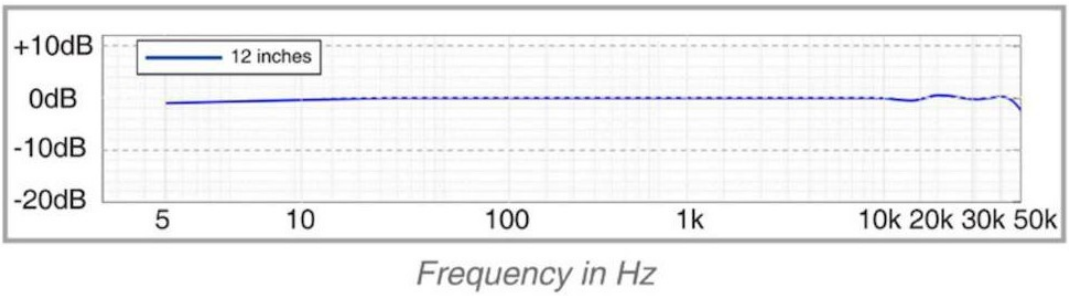


Figure 6.3 – The Earthworks M50 Frequency Response [91]

However, the probability of this case is small since the frequency signature of similar sound events is different. The probability of this event will even decrease more if the supersonic signature of the gunshots is present within the data.

6.2 Conclusions and Future Work

In this research study, we have investigated the problem of gunshot detection in wildlife and the possibility of an acoustic-based solution for poaching prevention. We have simulated the African savanna sounds and proposed a deep learning model to classify gunshots in different weather conditions. We have also investigated the correlation between the model accuracy and shooter distance to help estimate system reliability and the end-node grid size.

Although this work presents promising results for gunshot detection in wildlife, there is still room for improvement and further research. We propose the following topics that we feel will improve the system:

- **Gathering actual data:** Instead of synthesizing the data and simulating the environment, One can create significantly detailed and accurate data by actually recording the sounds in the field. If collected this way, there will be control over parameters that were out of our hands in this research. For instance, if the shooter distance is known at the recording time, it would help verify and even refine the model. Alternatively, choosing suitable acoustic sensors will avoid the loss of signal information such as high-frequency data. Finally, recording the gunshot sounds in different distances and angles would introduce diversity to the dataset known to the developer.

- **Feature extraction methods:** Even though using MFCCs has yielded acceptable results in this study, investigating other features might prove helpful. For instance, different features could lead to better accuracy or less computation complexity. In addition, training a model with multiple inputs (different feature vectors) may increase the model accuracy.

- **Timestamping the events:** To provide details about the events within the time frames and verify the true-positive predictions, one can design a

model that detects the exact time of gunshot within the frame. This may help to know the number of shots within the time frames. It can also clear doubts about the nature of true-positive predictions.

- **Multi-class and multi-label classification:** This work can be extended to detect other events as well. For instance, detecting vehicle sounds or human speech can also help the poaching prevention process. Additionally, having multiple labels would provide more information which might help the decision-making process. In other words, if the model can also recognize the presence of birdcalls, insect sounds, and weather conditions within the time frames, the system can classify the events with more confidence.

Bibliography

[1] *Illegal hunting poaching*. URL: https://www.sensoguard.com/illegal-hunting-poaching-security-2/.

[2] *Smart Parks*. URL: https://www.smartparks.org/work/.

[3] S Mohandass, S Sridevi, and R Sathyabama. "A UNIFIED APPROACH TO ANIMAL INTRUSION DETECTION FOR PREVENTING HUMAN-WILDLIFE CONFLICT AND CROP PROTECTION". In: *Journal of Critical Reviews* (2020). ISSN: 2394-5125. DOI: 10.31838/jcr.07.19.875. eprint: http://www.jcreview.com/?mno=136113. URL: http://www.jcreview.com/?mno=136113.

[4] Namrata Ansari Hardiki Patil. "Automated Wild-Animal Intrusion Detection and Repellent System Using Artificial Intelligence of Things". In: 2021. URL: http://dx.doi.org/10.2139/ssrn.3867275.

[5] Wikipedia. *Internet of things*. URL: https://en.wikipedia.org/wiki/Internet_of_things.

[6] RF Wireless World. *Advantages of motion sensor | Disadvantages of motion sensor*. 2012. URL: https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-motion-Sensor.html.

[7] Chloé Clavel, Thibaut Ehrette, and Gaël Richard. "Events detection for an audio-based surveillance system". In: *2005 IEEE International Conference on Multimedia and Expo*. IEEE. 2005, pp. 1306–1309.

[8] Marco Crocco et al. "Audio Surveillance: A Systematic Review". In: *ACM Comput. Surv.* (2016). ISSN: 0360-0300. DOI: 10.1145/2871183. URL: https://doi.org/10.1145/2871183.

[9] G. Valenzise et al. "Scream and gunshot detection and localization for audio-surveillance systems". In: *2007 IEEE Conference on Advanced Video and Signal Based Surveillance*. 2007. DOI: 10.1109/AVSS.2007.4425280.

[10] *Chengeta Wildlife*. URL: https://chengetawildlife.org/.

[11] Wenling Xue, Ting Jiang, and Jiong Shi. "Animal intrusion detection based on convolutional neural network". In: *2017 17th International Symposium on Communications and Information Technologies (ISCIT)*. 2017. DOI: 10.1109/ISCIT.2017.8261234.

[12] "Scream and gunshot detection in noisy environments". In: *2007 15th European Signal Processing Conference*. 2007, pp. 1216–1220.

[13] Marius Vasile Ghiurcau et al. "Towards an application for detecting intruders in wildlife regions". In: *2010 18th European Signal Processing Conference*. 2010.

[14] Marius Vasile Ghiurcau, Corneliu Rusu, and Radu Ciprian Bilcu. "Wildlife intruder detection using sounds captured by acoustic sensors". In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2010. DOI: 10.1109/ICASSP.2010.5495924.

[15] Marius Vasile Ghiurcau, Comeliu Rusu, and Radu Ciprian Bilcu. "A modified TESPAR algorithm for wildlife sound classification". In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. 2010. DOI: 10.1109/ISCAS.2010.5537179.

[16] "Low Cost using Deep Learning on the Raspberry Pi". In: *2019 IEEE International Conference on Big Data (Big Data)*. 2019, pp. 3038–3044. DOI: 10.1109/BigData47090.2019.9006456.

[17] Jose Apolinario Izabela Freire. "Gunshot detection in noisy environments". In: 2010. URL: https://www.researchgate.net/publication/228741330_Gunshot_detection_in_noisy_environments.

[18] "Gunshot Detection Using Convolutional Neural Networks". In: *2020 24th International Conference Electronics*. 2020, pp. 1–5. DOI: 10.1109/IEEECONF49502.2020.9141621.

[19] R.A. King and Mr T.C. Phipps. "Shannon, TESPAR and approximation strategies". In: *Computers & Security* (1999). ISSN: 0167-4048. DOI: https://doi.org/10.1016/S0167-4048(99)80111-4. URL: https://www.sciencedirect.com/science/article/pii/S0167404899801114.

[20] David Lindgren et al. "Shooter localization in wireless sensor networks". In: *2009 12th International Conference on Information Fusion*. 2009.

[21] T. Ahmed, M. Uppal, and A. Muhammad. "Improving efficiency and reliability of gunshot detection systems". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 513–517. DOI: 10.1109/ICASSP.2013.6637700.

[22] Settha Tangkawanit and Surachet Kanprachar. "Spectral Vector Design for Gunfire Sound Classification System with a Smartphone using ANN". In: *2018 21st International Symposium on Wireless Personal Multimedia Communications (WPMC)*. 2018. DOI: 10.1109/WPMC.2018.8712930.

[23] Hendrik Purwins et al. "Deep Learning for Audio Signal Processing". In: *CoRR* abs/1905.00078 (2019). URL: http://arxiv.org/abs/1905.00078.

[24] "Environmental sound processing and its applications". In: *IEEJ Transactions on Electrical and Electronic Engineering* 14.3 (2019), pp. 340–351. DOI: https://doi.org/10.1002/tee.22868. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/tee.22868. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/tee.22868.

[25] Eduardo Fonseca et al. "General-purpose tagging of Freesound audio with AudioSet labels: Task Description, Dataset and Baseline". In: *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*. 2018. URL: http://dcase.community/documents/workshop2018/proceedings/DCASE2018Workshop_Fonseca_114.pdf.

[26] J. Salamon, C. Jacoby, and J. P. Bello. "A Dataset and Taxonomy for Urban Sound Research". In: *22nd ACM International Conference on Multimedia (ACM-MM'14)*. Orlando, FL, USA, 2014, pp. 1041–1044. URL: https://urbansounddataset.weebly.com/urbansound8k.html.

[27] Tatsuya Harada Yuji Tokozume Yoshitaka Ushiku. "Learning from Between-class Examples for Deep Sound Recognition". In: *CoRR* abs/1711.10282 (2017). URL: http://arxiv.org/abs/1711.10282.

[28] Arthur Fox. *What Is The Difference Between Sound And Audio?* URL: https://mynewmicrophone.com/what-is-the-difference-between-sound-and-audio/.

[29] *Audio signal processing*. URL: https://en.wikipedia.org/wiki/Audio_signal_processing.

[30] *Sound waves*. URL: https://www.pasco.com/products/guides/sound-waves.

[31] *Analog Signals vs. Digital Signals*. URL: https://www.monolithicpower.com/en/analog-vs-digital-signal.

[32] D. Self. *Audio Engineering Explained*. Taylor & Francis, 2012. ISBN: 9781136121265. URL: https://books.google.nl/books?id=aLDpAwAAQBAJ.

[33] Francesc Alías, Joan Claudi Socoró, and Xavier Sevillano. "A Review of Physical and Perceptual Feature Extraction Techniques for Speech, Music and Environmental Sounds". In: *Applied Sciences* 6.5 (2016). ISSN: 2076-3417. DOI: 10.3390/app6050143. URL: https://www.mdpi.com/2076-3417/6/5/143.

[34] Dalya Gartzman. *Getting to Know the Mel Spectrogram*. 2019. URL: https://towardsdatascience.com/getting-to-know-the-mel-spectrogram-31bca3e2d9d0.

[35] *Meaning of negative values in audio waveforms*. URL: https://stackoverflow.com/questions/1380692/meaning-of-negative-values-in-audio-waveforms.

[36] Devopedia. *Audio Feature Extraction*. 2021. URL: https://devopedia.org/audio-feature-extraction.

[37] mlearnere. *Learning from Audio: The Mel Scale, Mel Spectrograms, and Mel Frequency Cepstral Coefficients*. 2021. URL: https://towardsdatascience.com/learning-from-audio-the-mel-scale-mel-spectrograms-and-mel-frequency-cepstral-coefficients-f5752b6324a8.

[38] Leland Roberts. *Understanding the Mel Spectrogram*. 2020. URL: https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53.

[39] A.V. Oppenheim and Ronald Schafer. "From Frequency to Quefrency: A History of the Cepstrum". In: *Signal Processing Magazine, IEEE* 21 (Oct. 2004), pp. 95 –106. DOI: 10.1109/MSP.2004.1328092.

[40] D. H. Johnson. "Signal-to-noise ratio". In: *Scholarpedia* 1.12 (2006). revision #126771, p. 2088. DOI: 10.4249/scholarpedia.2088.

[41] Wikipedia. *Signal-to-noise ratio*. URL: https://en.wikipedia.org/wiki/Signal-to-noise_ratio.

[42] EAR Customized Hearing Protection. *Gunfire Noise Level Reference Chart*. URL: https://earinc.com/gunfire-noise-level-reference-chart/.

[43] Brandon Maddox. *How loud is a gunshot? Gun dB levels compared*. 2019. URL: https://www.silencercentral.com/blog/how-loud-is-a-gunshot-gun-db-levels-compared/.

[44] Noise Monitoring Services. *How loud is a gunshot?* 2020. URL: https://www.noisemonitoringservices.com/how-loud-is-a-gunshot/.

[45] Wikipedia. *Inverse-square law*. URL: https://en.wikipedia.org/wiki/Inverse-square_law#Sound_in_a_gas.

[46] Victor Kiprop. *The World's Loudest Animals*. 2018. URL: https://www.worldatlas.com/articles/the-world-s-loudest-animals.html.

[47] DELTA. *Typically noise levels*. URL: https://eng.mst.dk/media/mst/69037/typically_noise_levels_1175.1.11.pdf.

[48] IAC Acoustics. *Comparative Examples of Noise Levels*. URL: https://www.iacacoustics.com/blog-full/comparative-examples-of-noise-levels.html.

[49] Brian Donohue and John Pearse. *Rain noise*. Universitätsbibliothek der RWTH Aachen, 2019.

[50] *Damping of sound level (decibel dB) vs. distance*. URL: http://www.sengpielaudio.com/calculator-distance.htm.

[51] Arthur Fox. *Units Of Measurement & Prefixes In Sound & Audio Electronics*. URL: https://mynewmicrophone.com/units-of-measurement-prefixes-in-sound-audio-electronics/.

[52] Robert Maher and Steven Shaw. "Deciphering Gunshot Recordings". In: (2008). URL: https://www.montana.edu/rmaher/publications/maher_aesconf_0608_1-8.pdf.

[53] Robert C. Maher. "Acoustical Characterization of Gunshots". In: *2007 IEEE Workshop on Signal Processing Applications for Public Security and Forensics*. 2007, pp. 1–5. DOI: 10.1109/IEEECONF12259.2007.4218954.

[54] David Lindgren et al. "Shooter Localization in Wireless Microphone Networks". In: *EURASIP J. Adv. Sig. Proc.* 2010 (Feb. 2010). DOI: 10.1155/2010/690732.

[55] Judith Hurwitz & Daniel Kirsch. *Machine Learning for Dummies*. John Wiley & Sons, Inc., 2018. ISBN: 978-1-119-45494-6. URL: https://www.ibm.com/downloads/cas/GB8ZMQZ3.

[56] IBM Cloud Education. *Machine Learning*. 2020. URL: https://www.ibm.com/cloud/learn/machine-learning#toc-how-machin-NoVMSZI_.

[57] Jorge Castañón. *10 Machine Learning Methods that Every Data Scientist Should Know*. 2019. URL: https://towardsdatascience.com/10-machine-learning-methods-that-every-data-scientist-should-know-3cc96e0eeee9.

[58] David M. W. Powers. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation". In: *CoRR* abs/2010.16061 (2020). arXiv: 2010.16061. URL: https://arxiv.org/abs/2010.16061.

[59] Jun Wu. *AI, Machine Learning, Deep Learning Explained Simply*. 2019. URL: https://towardsdatascience.com/ai-machine-learning-deep-learning-explained-simply-7b553da5b960.

[60] Dishashree. *Fundamentals of Deep Learning – Activation Functions and When to Use Them?* 2020. URL: https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/.

[61] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[62] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[63] IBM Cloud Education. *Convolutional Neural Networks*. 2020. URL: https://www.ibm.com/cloud/learn/convolutional-neural-networks.

[64] Wikipedia. *Hyperparameter optimization*. URL: https://en.wikipedia.org/wiki/Hyperparameter_optimization.

[65] Shashank Ramesh. *https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-i-hyper-parameter-8129009f131b*. 2018. URL: Aguidetoanefficientwaytobuildneuralnetworkarchitectures-PartI:Hyper-parameterselectionandtuningforDenseNetworksusingHyperasonFashion-MNIST.

[66] Google Cloud. *Overview of hyperparameter tuning*. URL: https://cloud.google.com/ai-platform/training/docs/hyperparameter-tuning-overview.

[67] RichardK15. *5 gunshot sounds*. 2019. URL: https://www.kaggle.com/richardk15/5-datasets.

[68] Emrah AYDEMİR. *Gunshot audio dataset*. 2021. URL: https://www.kaggle.com/emrahaydemr/gunshot-audio-dataset.

[69] Tito Spadini. *Sound Events for Surveillance Applications*. Version 1.0.0. Oct. 2019. DOI: 10.5281/zenodo.3519845. URL: https://doi.org/10.5281/zenodo.3519845.

[70] George Vlad. *Nature and wildlife sounds from the African savanna*. 2018. URL: https://www.youtube.com/watch?v=OcVtCTBTJ-4.

[71] George Vlad. *Nature and wildlife sounds from the African savanna - Night in Tsavo*. 2020. URL: https://www.youtube.com/watch?v=bsuYcwlzr2w\&t.

[72] MeditationRelaxClub. *Nature Sounds: Rain Sounds One Hour for Sleeping, Sleep Aid for Everybody*. 2013. URL: https://www.youtube.com/watch?v=Mr9T-943BnE\&ab.

[73] *SAVANNA*. 2004. URL: http://kids.nceas.ucsb.edu/biomes/savanna.html.

[74] Sounds of Nature and Beyond. *RELAX OR STUDY WITH NATURE SOUNDS: Ultimate Thunderstorm*. 2013. URL: https://www.youtube.com/watch?v=T9IJKwEspI8\&ab.

[75] *Thunder*. URL: https://en.wikipedia.org/wiki/Thunder.

[76] Siddharth Sigtia et al. "Automatic Environmental Sound Recognition: Performance Versus Computational Cost". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.11 (2016), pp. 2096–2107. DOI: 10.1109/TASLP.2016.2592698.

[77] Regunathan Radhakrishnan, Ajay Divakaran, and A Smaragdis. "Audio analysis for surveillance applications". In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.* IEEE. 2005, pp. 158–161.

[78] TIOBE PYPL PopularitY of Programming Language Index. *Top Computer Languages*. 2021. URL: https://statisticstimes.com/tech/top-computer-languages.php.

[79] Claire D. Costa. *Top Programming Languages for AI Engineers in 2021*. 2020. URL: https://towardsdatascience.com/top-programming-languages-for-ai-engineers-in-2020-33a9f16a80b0.

[80] Brian McFee et al. *librosa/librosa: 0.8.0*. Version 0.8.0. July 2020. DOI: 10.5281/zenodo.3955228. URL: https://doi.org/10.5281/zenodo.3955228.

[81] James Robert. *pydub 0.25.1*. URL: https://pypi.org/project/pydub/.

[82] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[83] *About Keras*. URL: https://keras.io/about/.

[84] Tom O'Malley. *Hyperparameter tuning with Keras Tuner*. 2020. URL: https://blog.tensorflow.org/2020/01/hyperparameter-tuning-with-keras-tuner.html.

[85] *Matplotlib: Visualization with Python*. URL: https://matplotlib.org/.

[86] *Savannah sparrow bird bathing, eating | Call sounds*. URL: https://www.youtube.com/watch?v=g66-30Bl6qU\&ab_channel=GoTrails.

[87] *Birds of Africa Part IV - Birds of Savanna*. URL: https://www.youtube.com/watch?v=rRRQkd_GYQk\&list=PLnCc1OlPWlwwkTA47iHNuTmsdQpBSw_RZ\&index=5\&ab_channel=africannature.

[88] *Beautiful birdsong in the African savanna*. URL: https://www.youtube.com/watch?v=tQlC_uHJeQE\&ab_channel=GeorgeVlad.

[89] Robert Beason. "What Can Birds Hear?" In: *U.S. Department of Agriculture National Wildlife Research Center-Staff Publications* 78 (Sept. 2004).

[90] *Gun Shooting Sound Effects*. URL: https://www.fesliyanstudios.com/royalty-free-sound-effects-download/gun-shooting-300.

[91] Arthur Fox. *Complete Guide To Microphone Frequency Response*. URL: https://mynewmicrophone.com/complete-guide-to-microphone-frequency-response-with-mic-examples/#How-Do-Microphones-Pick-Up-Different-Frequencies?.