Gunshot Detection from Audio Streams in Portable Devices

ELLEN GRANE & LINNEA BOKELUND
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Gunshot Detection from Audio Streams in Portable Devices

Ellen Grane
el6626gr-s@student.lu.se

Linnea Bokelund
li5147bo-s@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisor: Pierre Nugues

Axis supervisors: Robin Olofsson and Housam Abbas

Examiner: Maria Kihl

June 17, 2022

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

Machine learning and artificial neural networks can be used to classify or detect specific sound events in audio signals. Gunshot detection is one use case for such networks and can be used to help law enforcement by alerting officers or triggering camera recordings.

However, artificial neural networks with a high performance usually require large amounts of computational power, meaning that they do not work on smaller portable devices. This thesis shows that a small convolutional neural network (CNN) can be used for real-time gunshot detection on a portable camera without requiring too much memory, battery consumption, or CPU power.

We implemented a CNN with four layers and 100k trainable parameters to detect gunshots. We could reach an average precision of 0.98 and an F1 score of 0.95. We benchmarked the runtime performance of this architecture on the Axis Body Worn Cameras (BWCs). For real-time gunshot detection, our system uses 11.9 MB RAM and requires 4.9 MB persistent memory; it decreases the battery time by only 8.4% and uses approximately 11.5% of the CPU. With our configuration, the real-time detection has a latency of 3.6 seconds on the BWC.

The results of our Master's thesis show that audio-based gunshot detection on portable devices is indeed viable. We hope it will encourage the research on simpler features for audio classification.

Acknowledgments

We would like to thank our supervisors Pierre Nugues, Robin Olofsson and Housam Abbas for all of their encouragement and guidance throughout our thesis work. Pierre has been a great support with the academic writing and the machine learning work. Robin and Housam have been exceptionally helpful with everything regarding the BWC.

This work would not have been possible without the help from everyone at Axis Communications, and we would therefore like to express our gratitude to all whom we have come in contact with who have provided useful input within their respective areas of expertise. The existing knowledge within the organization has been an invaluable resource during the course of our project. We would especially like to thank Pontus Nelderup, Peter Eneroth, Fredrik Hugosson and Stefan Andersson for all of their valuable feedback and guidance, and for presenting the opportunity for us to do this master's thesis in the first place.

Furthermore, we want to express our gratitude to Lunds Pistolklubb, who were very accommodating when allowing us to record gunshots in order to extend our dataset. Thank you also to Ivan Kruzela and Gustav Strömberg, who provided helpful feedback on our report already in early the stages.

Popular Science Summary

When a threatening situation suddenly arises, it may be difficult to quickly get an opportunity to alert the authorities. During such circumstances, it would be beneficial if a nearby electronic device could be set to automatically detect the threat, and trigger appropriate actions to get help.

In this master thesis, we focus on detecting gunshot sounds on the Axis Body Worn Camera (BWC) that is used by police officers and guards. It is not hard to imagine that in a situation where shots are fired, the police officer does not have the possibility to start a recording manually. If the camera could identify gunshots, it could trigger a recording automatically and thus ensure that evidence is captured. The BWC has a prebuffer functionality that allows the last 90 seconds before a recording is started to be included in the resulting video, meaning that the events leading up to the gunshot will also be included.

Gunshot detection has been done before, but often the detection is performed on a powerful device such as a server. Our target device is a small, portable camera with limited battery life, CPU and memory. Therefore, our main challenge was to find a solution that uses as little energy and memory as possible, while at the same time identifies gunshots with a high accuracy.

Within machine learning, image recognition has been researched and developed widely over the last decades. Recognition of particular sounds in audio recordings has not been explored to the same extent. However, by transforming the audio signals into spectrograms that can be viewed as image representations of the signals, the advances in image recognition can be applied to sound identification as well. A common technique used for image recognition is convolutional neural networks (CNNs), and this is what we use for our gunshot detection.

A very important aspect of machine learning is the data. Regardless of the chosen technique, the model will need a lot of data to train on. In our case, we needed data in the form of sound recordings containing gunshots. We also needed other sounds, for the model to learn what a gunshot does not sound like. We used a public dataset called FSD50K containing tens of thousands of clips with sounds of people, traffic, animals, and much more. However, this dataset did not have enough gunshot sounds. Therefore, we recorded our own dataset in cooperation with a local pistol club.

By implementing a small CNN that we trained on the FSD50K dataset and

our gunshot sounds, we were able to create a prototype that works on the BWC. With a smaller CNN, the computations needed to analyze one audio clip is reduced and thereby also the energy consumption. We found that to detect gunshot sounds in audio signals, a small CNN is sufficient, and thus our conclusion is that it is indeed viable to implement a machine learning gunshot detection on a small device such as the BWC.

Acronyms

ANN	Artificial Neural Network
AP	Average Precision
BNN	Binarized Neural Network
BWC	Body Worn Camera
CC	Creative Commons
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DFT	Discrete Fourier Transform
EDA	Exploratory Data Analysis
FFT	Fast Fourier Transform
FN	False Negative
FP	False Positive
GAN	Generative Adversarial Network
GDPR	General Data Protection Regulation
GTCC	Gammatone Cepstral Coefficients
KD	Knowledge Distillation
mAP	mean Average Precision
MFCC	Mel-Frequency Cepstral Coefficient
MFNN	Multilayer Feedforward Neural Network
ML	Machine Learning
PCM	Pulse Code Modulation
PSS	Proportional Set Size
RAM	Random Access Memory
RNN	Recurrent Neural Network
SVM	Support Vector Machine
TF	TensorFlow
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate
USS	Unique Set Size

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

When a threatening situation suddenly arises, it may be difficult to quickly get an opportunity to alert the authorities. During such circumstances, it would be beneficial if a nearby electronic device could be set to automatically detect the threat, and trigger appropriate actions to get help.

Today, we are surrounded by electronic devices with the ability to capture audio; which possibly could be used for such threat detection. The cellphone, which almost everyone constantly carries in their pocket, is one example. For this kind of device, it would be useful, in some situations, if the captured audio signal could be automatically analyzed to predict if the current situation is threatening. This information could then be used to trigger actions to protect, alert, or something else. For example, while walking home at night, the cellphone could automatically call an emergency number if it detects that a threatening situation has emerged.

The detection of threats in an audio signal is a 'sound event detection' problem. This field is increasingly popular in machine learning. Many state-of-the-art solutions in this area achieve great results in terms of accuracy. However, most of the models have high complexity and require a lot of memory and CPU usage to execute the classification. This means that despite their high performance, they are not well suited for less powerful devices.

In this work, we designed a system to detect threatening situations by identifying gunshots from an audio signal. We implemented this real-time system on a body worn camera from the Axis company. When detecting a gunshot, our system triggers a video and sound recording on this camera. However, portable devices usually have a limited battery, memory, and processing power. Therefore, we investigated the effect that such detection had on the devices' energy, CPU, and memory consumption. Furthermore, we measured the latency from audio signal uptake to the finished prediction of a gunshot.

From our measurements, we concluded that using such a functionality on a portable device such as the BWC indeed is possible. Whilst the real-time detection had some negative impact on the battery time and CPU usage, it was not unreasonable. For future research, we suggest investigating other features for audio analysis that are easier to compute, but still allow small networks to achieve high performance.

1.1 Problem Definition

The goal of our thesis was to develop a method using machine learning (ML) that could accurately detect and classify gunshots from audio signals. In addition, it had to be simple enough to work on a portable device, such as the BWC. To evaluate our solution, we plan to use the following metrics:

- Accuracy of predictions;
- Latency from signal detection and prediction of a gunshot;
- Power consumption and memory usage when performing the prediction.

1.2 Delimitations

For a device such as the Axis BWC, it could be useful to create a system with the ability to detect threats in general. For example, many times heated situations start off with an argument. If a system were able to detect signs of aggressiveness in speech or yelling, a threatening situation might identified at an early stage, and countermeasures can be implemented to prevent it from escalating. An in-depth threat detection like this would require a well-annotated dataset for emotion recognition, and would furthermore entail an overall higher complexity. For this reason, we have limited our research to regard only the detection of gunshot sounds.

To make predictions accurate, the data used to train the machine-learning model should reflect real-usage situations as much as possible. In this work, the detection is carried out on an Axis BWC. This means that the dataset used to train the machine learning model would optimally be recorded on this device. We then recorded gunshots using BWCs, and we used the resulting audio in both the training and testing of our model.

Regarding other types of sounds, we did not have the time and resources to create a sufficiently sizable and varied dataset with enough coverage. Instead, we decided to use a publicly available dataset.

1.3 Thesis Outline

In the next chapter, previous research related to this thesis will be presented. Chapter 3 describes the dataset that was used, and furthermore includes a description of how the collection of gunshot audio was done. In Chapter 4, relevant concepts regarding machine learning, audio processing and common metrics that are used to evaluate machine learning models are introduced. Chapter 5 contains information about how the implementation and evaluation of performance were executed.

The results of our work are presented in Chapter 6, and these are discussed and analyzed in the following chapter. Finally, conclusions that may be drawn from our study are summarized in Chapter 8.

1.4 Contributions

Most parts of the report have been written by both Ellen and Linnea. In this section, we will specify the parts where one of us has contributed a majority of the writing. Both of us contributed an even amount to the implementation and measurements.

Ellen has written Sections 3.2 describing how we collected data, 2.3 about low complexity machine learning models, 2.2 about previous research on threat detection, as well as 4.1 about artificial neural networks.

Linnea has written Sections 4.2 about audio processing, 3.3 about the exploratory data analysis, 2.1 about state-of-the-art solutions on FSD50K and 2.4 about previous work on gunshot detection. She also composed the abstract of the report.

Chapter 2
Related Work

In this chapter, we will describe previous research related to our project. First, we will present two solutions for sound event recognition; the current and previous state-of-the-art networks on the dataset we have used (FSD50K). The section will give a brief introduction to present-day machine learning techniques for audio processing in general. In addition, it will form a reference point for our experiments on the dataset.

The second section will explore research on threat detection, and the third describes earlier research in the field of implementing machine learning on less powerful devices. Finally, we will close in on our task and look at previous work on gunshot detection.

2.1 FSD50K

FSD50K [11] is a dataset for sound recognition with 200 labels and over 50,000 audio clips. Each clip is annotated with one or more labels describing what sounds it contains. For a more thorough description, see Chapter 3. To compare different networks' performances on the dataset, the mean average precision (mAP) which gives a summarized measure for the precision of the different labels. The mAP-score is between 0 and 1, where 1 is a perfect score.

2.1.1 PaSST-S

At the time of writing, the *Patchout faSt Spectrogram Transformer with Structured patchout (PaSST-S)* by Koutini et al. [22] is considered to be the current state-of-the-art model for FSD50K. PaSST-S is a transformer-based model with 87M parameters that uses ImageNet pretraining and Patchout for regularization. The network takes mel-frequency cepstral coefficients (MFCCs) as input. MFCCs are features which can be extracted from audio signals, specifically designed to mimic the way humans perceive sound. PaSST-S achieves a mAP score of 0.653.

The transformer structure was first introduced by Vaswani et al. [35] to reduce the computational complexity of machine learning algorithms for sequential problems such as language modeling. Prior methods to solve such problems have been to use recurrent neural networks (RNN) or convolutional neural networks (CNN).

However, to retain and use the sequential information of the input, the computational complexity increase with longer sequences. Instead, the transformer model use a set of attention layers. Attention layers allow the model to focus only on selected parts of the input data, and the layers learn to choose what parts to focus on wisely [20].

In PaSST, Koutini et al. use these attention layers together with a novel method they call Patchout. Patchout forces the model to train on incomplete data by randomly dropping parts of the input sequence. The appended S in PaSST-S, means that structured Patchout is used. Structured Patchout means that a random frequency bin or time frame is picked, and the respective row or column is removed from the MFCCs. Koutini et al. state that Patchout reduces the computational complexity by reducing the input sequence length during training and at the same time regularizes the model.

Transformer models need more data than CNNs to reach high performance [9] but by using pretrained networks the amount of data needed can be reduced [15]. Therefore, PaSST use ImageNet pretraining, meaning that they incorporate a network that is already trained on a large dataset of images (ImageNet) and then fine-tune the network for a specific task [22]. The pretraining extracts information from images that are useful for classification and recognition in general, while the fine-tuning targets the information that is specifically interesting for the task.

2.1.2 PSLA

Before PaSST-S, the previous state-of-the-art was *PSLA* by Gong et al. [16] with a mAP score of 0.567. However, PSLA is considerably smaller than PaSST-S with 13.6M parameters. This model is CNN-based, takes a log mel filterbank feature vector as input, and utilizes ImageNet pretraining as well as an ensemble method.

PSLA stands for Pretraining, Sampling, Labeling and Augmentation. The pretraining is used in the same way as described for PaSST-S in the previous section. Sampling is used to increase the balance between classes in the dataset by randomly sampling the data with distributions related to how often a class occurs in the dataset. In short, classes that are less frequent in the training data will be upsampled while classes that are more frequent will be downsampled.

Labeling refers to the process of *label enhancement*, which addresses the issue of errors in the dataset's annotations. Human-made annotations naturally contain mistakes that can impact the performance. Gong et al. apply a label enhancement algorithm to the dataset to identify and adjust such mistakes.

Finally, augmentation is also a way to reduce the imbalance between classes by creating new samples for the less frequently occurring classes. The augmentation is done by adding frequency and time masks to the audio clips, as well as randomly mixing clips from the dataset into one clip.

In addition to the techniques described above, Gong et al. test different ensemble methods. Ensemble means that a set of networks are trained on the same task, and then they are used together. A common way is to use the average output from all networks as prediction, see a simple example in Figure 2.1. It is also possible to use weighted average or intermediate results from the networks.

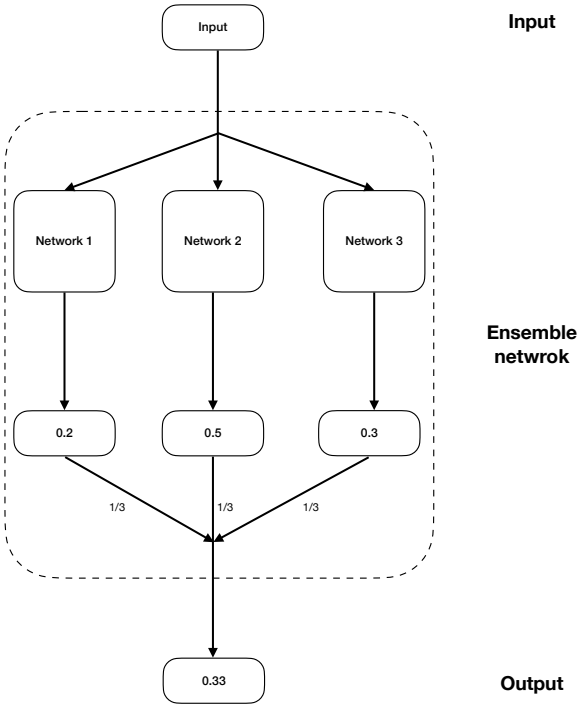


Figure 2.1: An example of an ensemble of three networks using the average output as result.

2.2 Threat Detection

In 2012, Glowacz and Altman [13] proposed a method to detect and classify threatening sounds such as gunshots, screams, and acts of vandalism, by using a support vector machine (SVM). The intended purpose of this audio-based threat classification was to combine it with video-based analysis in order to implement a system of intelligent monitoring. The feature extraction that was used produced MFCCs, MFCC derivatives, signal frame energy and an energy derivative, which was then used as input to the model. With this approach, they managed to reach an efficiency of 79.17% in their threat classification.

A more specific type of sound which may indicate a threatening situation is aggressive tendencies in human speech. In their report from 2007, Hengel and Andringa [34] have presented an audio-based verbal aggression detection system which is able to detect aggressive shouting. The system is intended to work in situations with noisy backgrounds, and thus the first step includes the separation of background and foreground sounds. The pitch of the foreground sounds is then

analyzed as to conclude whether they contain a human voice. If a sound signal is classified as speech, the distortion of the voice is examined and compared to cues that are known to indicate stress and aggravation. This then generates an overall probability of whether the sound signal contains verbal aggression. After optimization, the system managed to reach a near-human performance.

Kooji et al. [21] expand upon this research by designing and implementing a system, called CASSANDRA, which combines video and audio analysis to detect human aggression. For the audio analysis part, they improve the foreground sound processing by utilizing tonal signal components. This makes it possible to identify and separately analyze foreground sounds that stem from different sources.

They tested their system by using three video cameras and a microphone set up at a train station. To create the dataset, professional actors were ordered to play out plausible scenarios which included exhibition of behavior ranging from normal, to critically aggressive. The CASSANDRA system showed promising results, considering the high complexity of the task. Their evaluation results indicate that combining audio and video has the potential to improve the estimation of aggression. CASSANDRA ran on two PCs and had not been optimized for real-time processing.

Jaafar and Lachiri [19] investigated the possibility of using deep neural networks to analyze complementary audio and video in order to classify the level of human aggression in different situations. In addition to combining audio and video predictions to make a final decision, they also take previous classifications (history) into consideration. In cases where the audio and video predictions do not agree, the history may provide context which helps in making a decision. With this approach, they managed to receive improved accuracy for all classes compared to previous studies. This shows that usage of deep neural networks in this area is very promising. It further indicates that using meta-features such as history to impact the fusion of predictions from audio and video may serve to improve the classification performance.

2.3 Low Complexity Machine Learning Models

Many machine learning models require large amounts of memory and computation power to work. While most of the work is required during training, the classification itself can also be quite demanding for very small devices. When such resources are limited, it will be difficult to reach state-of-the-art performance, as more advanced deep learning techniques often correlate to higher usage of memory and power [7]. For such devices, it is instead important to find a way of lowering the complexity whilst preserving as much capability as possible.

Unfortunately, there is no standardized way of measuring the complexity of a network. Instead, a variety of methods to estimate it has been used in different articles. One reason why this is the case may be that the actual inference time, memory usage, and CPU load highly depend on the target device. Additionally, many articles focus on time and memory consumption during training rather than during the actual classification.

During training, thousands of predictions have to be made, preferably at a

fast pace. When the model is used, it is more common to execute one prediction at a time. So for a model which is intended to run on a personal computer or a larger server, the requirements for running the model should not be a huge issue. However, for models meant to run on smaller devices, it is more critical. Usually, the model is trained on a more powerful computer and then transferred to the smaller device. Thus, the computational complexity of a single prediction is more important. One way of estimating complexity that is used in literature is the number of parameters in the model. Furthermore, the number of operations required to compute a prediction may be used to compare the complexity. Additionally, the execution time and memory usage on a specific device can be used to evaluate models.

Cerutti et al. [7] explain that one common strategy when approaching this issue is to design and implement smaller networks, specifically created to fit the particular hardware capacity. These are then trained and can be implemented on a low-power microcontroller. Another possible approach is to compress a trained model in order to create a network with lower complexity that produces similar results as the original one. One way of doing this is with network pruning, meaning finding and removing negligible weights from a trained model until a specified condition is met.

It is furthermore possible to reduce the network's persistent memory size by using matrix/tensor factorization, which exploits the fact that networks have a linear structure. These methods require much fine-tuning, and do not change the RAM usage.

Another way of compressing a neural network is by *quantization*, i.e. to reduce the number of bits needed to represent the weights and activations. An extreme case of quantization is *binarized neural networks* (BNNs) which have weights and activations that are constrained to be 1 or -1 [8]. BNNs drastically reduce the memory size of the network and allow for bitwise operations, which lead to an increased power efficiency [24].

2.3.1 Knowledge distillation

Cerutti et al. [7] further state that knowledge distillation (KD) can be used for compressing. KD is done by using a complex model to train a simpler model. It is also called Student-Teacher, as the smaller model (student) learns to mimic the larger one (teacher). This is based on the idea that the output of the complex model (soft labels) contains more information than the input labels (hard labels), which facilitates the training.

Cerutti et al. [6] show that by applying KD, a CNN-based sound event detection classification model can be compressed significantly without a notable loss in accuracy. The large pre-trained network called VGGish that Cerutti et al. use as teacher has approximately $7 \cdot 10^7$ parameters and requires $1.7 \cdot 10^9$ operations to perform classification and has an average accuracy of 74.7% on the ten classes in the UrbanSound8K dataset. Something to note is that VGGish has been trained on a different dataset, and this may affect the results. When distilling the knowledge to a student network (M_{20k}) with $3.06 \cdot 10^4$ parameters that require $2.11 \cdot 10^6$ operations, the accuracy is decreased to 69.7%, while only using 0.0424%

of the number of parameters and 0.12% of the amount of operations compared to VGGish.

In a later report, Cerutti et al. [7] divide the knowledge distillation into two parts, predominantly to divide the domain adaptation and the parameter reduction. Thus, they first train a network of similar size to VGGish, using VGGish as teacher and the UrbanSound8k dataset for the training. This is done to fit the domain. Then they use the student network in the previous step as a teacher and train a small network of the same size as the previously described M_{20k} to compress it. By repeating the knowledge distillation to include domain adaptation, the accuracy score was further improved to 72.62% while retaining the same amount of compression.

2.3.2 Binary neural networks

By using a partly binarized convolutional neural network, Cerutti et al. [5] achieve an accuracy of 77.9% on a dataset with 28 classes consisting of audio clips from Freesound with a model that requires 262 kB memory in total for execution and storage. Compared to a full-precision neural network, this reduces the memory by a factor of 2.4 and decreases the accuracy by 7.3 percentage points. The network Cerutti et al. propose uses binary values in all layers except the first and the last ones, as these contribute with minimal computational cost.

2.4 Gunshot Detection

To the best of our knowledge, there is no task or dataset commonly used to measure and compare gunshot detection models. However, Bajzik et al. [3] have evaluated the performance on acoustic gunshot detection for three CNNs, well-established in the image recognition field. The three networks are *ResNet18* [17], *VGG16* [29], and *InceptionV3* [31], which Bajzik et al. set up to take MFCCs as input and give as output whether the underlying audio signal contains a gunshot. The models were trained and tested on the non-gunshot sounds from the dataset UrbanSound8K [28], as well as on gunshot sounds from a dataset called *The Free Firearm Sound Library - Expanded Edition* which, unfortunately, is no longer publicly available.

Finally, we have included two smaller networks that are specifically intended for acoustic gunshot detection. The first network is proposed by Baliram et al. [4] and consists of four convolutional layers, takes MFCCs of up to 0.6 seconds of recordings as input, and has only 44,000 trainable parameters. This network is trained and evaluated on the UrbanSound8K dataset. The second network proposed by Morehead et al. [27] is also a CNN, but takes a plain audio signal as input. The advantage of this network is that it does not need any feature extraction, which all the other networks do. This network is trained and evaluated on a dataset that the authors have constructed partly from free internet databases such as Freesound, and partly from audio clips generated by a so-called Generative Adversarial Network (GAN), which is a machine learning network that can be used to generate new data similar to some already existing data.

Table 2.1 shows a summary of all the available information about the networks' performances on gunshot detection. As these measurements are not made on the same datasets, the comparison is not entirely fair. However, it gives some sort of indication of the performance of the different networks.

Table 2.1: Summary of different networks' performances on acoustic gunshot detection. Data for VGG16, InceptionV3, and ResNet18 from [3].

Model	**Params.**	**Acc.**	**Prec.**	**TPR**	**TNR**	**F1**
VGG16	138 M	0.977	0.996	0.959	-	-
InceptionV3	23 M	0.957	0.995	0.919	-	-
ResNet18	11 M	0.991	0.995	0.988	-	-
Baliram [4]	44 K	-	0.951	0.907	0.965	0.929
Morehead [27]	320 K	0.994	0.98	0.966	-	0.973

Chapter 3
Dataset

3.1 Choice of Dataset

To get an accurate representation of the sounds the machine learning model is intended to analyze, it is important to use an appropriate dataset. Virtanen et al. [36, p. 149-150] have specified three properties that can be used to assess the suitability of a given dataset:

Coverage – The dataset should preferably include all categories that may be relevant.

Variability – The samples in each category should cover varied conditions (different voices, surroundings, etc.).

Size – Each category should consist of a sufficient number of samples.

To get a robust model that generates satisfying predictions, a combination of all these properties is necessary. If the size of the dataset is too small, machine learning algorithms tend to overfit the training data and perform worse on unseen data [36, p.164].

Baliram et al. [4] show that the choice of dataset highly affects the network's ability to distinguish between similar sounds from different sources. They collected a dataset with audio recordings of plastic bag pops and tested a gunshot detection model trained on the UrbanSound8K dataset [28]. The model wrongly classified 75% of the plastic bag pops as gunshots. Finally, Baliram et al. proved that the same type of model could be trained to distinguish between plastic bag pops and gunshots by training a binary classification model on the two types of sounds and reaching a sensitivity of 90% and specificity of 96%.

When training an ML model, it is common to divide the dataset into three parts: one for training, one for validation, and another for testing. The training and validation sets are both used during the training phase. In this stage, the training set is the dataset the model trains on, while the validation set is periodically used to evaluate the model's current performance, and the result is used to determine how the training should be tuned. When the model is finished with its training, the test set, which by then is unseen by the model, is used to evaluate the final performance.

3.1.1 Existing datasets

While choosing the main dataset, we analyzed three candidates: AudioSet, UrbanSound8K, and FSD50K.

1. *AudioSet* [12] was brought forth by Google and currently consists of 2 million classified 10-second clips originating from YouTube. Due to the clips being video rather than audio and thus requiring more preprocessing and more considerations regarding GDPR, we decided against using AudioSet.

2. *UrbanSound8k* [28] consists of 8,732 labeled audio streams originating from *Freesound*, and it contains 10 classes that can be found in an urban environment. As UrbanSound8k is a smaller version of FSD50K, we chose to go with the latter.

3. *Freesound Dataset 50K (FSD50K)* [11] was created to serve as the largest open annotated audio dataset for the recognition of events from sounds. The dataset includes over 100 hours of audio distributed over 51,197 human-labeled audio clips. The dataset is *multilabel* meaning that each clip can be labeled with one or more labels and there are in total 200 unique labels including different human and animal sounds, noise, instruments, and more. The dataset is open in the sense that the audio is collected from *Freesound*, where sound clips are released under the Creative Commons (CC) licenses. This is the dataset we chose to use, however, we have removed all samples with CC-BY-NC and CC Sampling+ licenses, which leaves a total of 43,379 audio clips.

3.2 Data Collection

In addition to the FSD50K dataset, we created our own dataset of gunshot recordings. The recordings were done in two environments: indoors and outdoors. More details around each of these recording sessions will be described in the next sections. In total, we collected more than 3000 audio clips, all containing at least one gunshot. The BWC records in stereo, but for classification it is sufficient to use mono. For this reason, we combined the channels of the audio clips afterwards, to get audio clips with mono sound.

3.2.1 Indoor recordings

The indoor recordings were produced at a shooting range located in a basement with concrete walls. The room had dimensions of 32.5x13m, and a ceiling height of 2.95 m. While recording, ten shooters stood beside each other and shot simultaneously, meaning that at times several gunshots overlapped. Figure 3.1 shows where the shooters were standing, and Figure 3.2 shows the rest of the room. Drawings of the different setups can be found in Appendix A.

The resulting recordings were automatically divided into clips of five or fewer seconds containing one or several gunshots. A majority of the gunshots in these clips are produced by the same kind of weapon, namely .22 caliber pistols. Most

of the clips have little background noise, apart from echoes from gunshots and the loading of weapons. Some clips have a background sound from an electrical fan.

Figure 3.1: During the indoor recordings, ten shooters stood in shooting stalls beside each other. The picture shows four of the shooting stalls.

3.2.2 Outdoor recordings

During the outdoor recording, several types of weapons and ammunition were used to increase the diversity of the audio clips. The shooting range was located in an area with little background noise, apart from the wind. One of the cameras used for recording was worn by a person moving around within 10 meters of the shooter, to mimic real-life usage. Two other cameras were placed at different distances from the shooter.

3.2.3 Mixed gunshot clips

To increase the variance among the gunshot audio clips, we decided to create an augmented dataset by layering the gunshot clips we collected with other sounds from FSD50K. The original sounds were not modified in terms of signal strength or likewise. Background sounds to overlap with gunshots were chosen at random from the categories ‘Animal’, ‘Music’, ‘Human sounds’, ‘Sounds of things’, and ‘Source-ambiguous sounds’.

We used background and gunshot sounds from the training set to create the mixed audio clips for the training set, and did the same for the validation and

Figure 3.2: The picture shows the room in front of the shooting stalls. The room is 13 meters wide and the distance from the shooting stalls to the target wall is 25 meters.

testing sets. In this way, the training, validation, and testing sets were kept separate.

3.3 Exploratory Data Analysis

To get a better understanding of the dataset, we used, we made an *exploratory data analysis* (EDA), examining its composition and trying to detect eventual problems with regard to coverage and variability. In the following section, we will outline the most important outcomes of it.

3.3.1 FSD50K

FSD50K is a large and diverse dataset with audio clips of length 0.3 to 30 seconds, each annotated with one or more labels out of the 200 labels defined for the dataset [11]. As already mentioned, we excluded all clips with CC-BY-NC and CC Sampling+ licenses, leaving a total of 43,379 audio clips. Furthermore, we chose to limit the clip length to 5 seconds and exclude all clips longer than that. We primarily did this to decrease the computational requirements. After this, 22,062 audio clips remained. Figure 3.3 shows the clip length distribution before and after removing clips longer than 5 seconds.

The amount of labels and the multilabel nature of FSD50K makes it difficult to get an overview of the content. However, the annotations follow the AudioSet ontology, which organizes sound event classes into a hierarchy. The AudioSet ontology superclasses which are represented in FSD50K are *Human sounds*, *Animal*, *Source-ambiguous sounds*, *Sounds of things*, *Natural sounds*, and *Music*. To make the data more comprehensible, we collapsed all annotations into these superclasses

when performing the EDA, except the label *Gunshot and gunfire* as it is of most interest in this project. For the interested reader, more information about the 200 classes and the hierarchy between classes can be found in Appendix B.

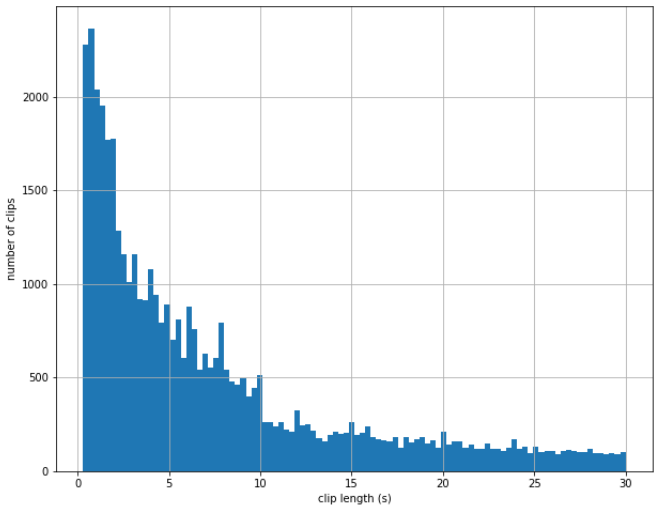


Figure 3.3: Audio clip length distribution in FSD50K before restricting the clip length to less than 5 s.

Figure 3.4 gives an overview of the distribution between the superclass labels and the gunshot label. Each clip that has at least one label from a certain superclass is counted for that superclass except *Gunshot and gunfire*. Sounds of gunshots are underrepresented in the dataset and are only present in 1.45% of the 22,062 audio clips. On the other side of the spectrum, *Music* is the most represented superclass sound, with 35.6% of the clips.

3.3.2 Gunshot audio clips

We combined the subset of FSD50K described in the previous section with the audio recordings we collected ourselves. In this section, we will deep dive into the characteristics of the gunshot audio clips in the resulting dataset. As can be seen in Table 3.1, the audio clips from the indoor recordings and the mixed gunshot clips make out the vast majority of the gunshot clips. This makes the gunshot sounds in our dataset heavily biased towards the settings of the indoor recordings, as many of the gunshot sounds in the mixed clips are taken from the indoor recordings. In particular, all the clips from the indoor recordings are recorded in the same room with the same acoustic characteristics, with a relatively small distance to the weapon, and most of the shots were fired from the same kind of weapon.

The audio clips containing gunshot sounds range from under 1 second to 5 seconds. However, most of them are between 1 and 2 seconds long (see Figure 3.5). Some clips contain only one gunshot, while others contain two or more gunshots fired after each other. Figure 3.6 shows the audio wave and mel spectrogram of one of the clips we recorded indoors. In this clip, three shots were fired shortly

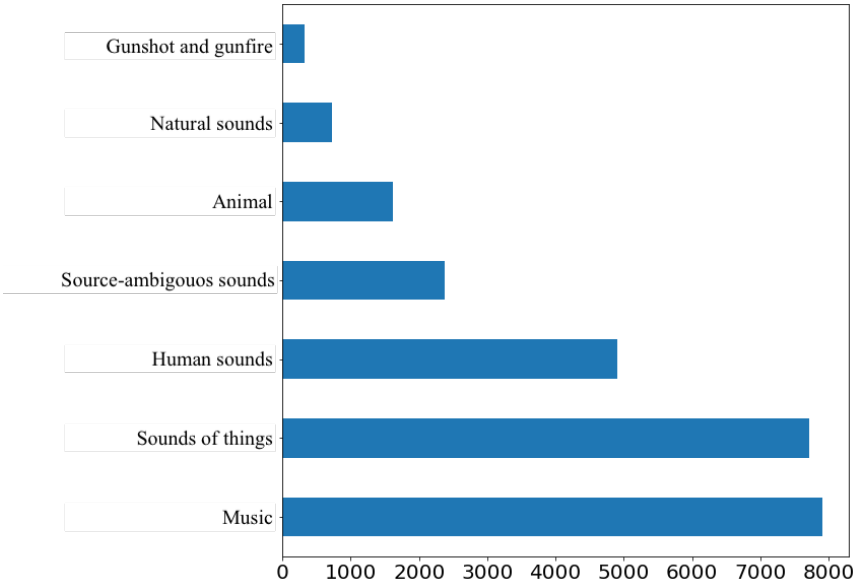


Figure 3.4: Label distribution for the audio clips used from FSD50K. Labels are combined into superclasses according to the Audioset ontology.

after each other, and it is very clear in both the wave and the spectrogram at what time it happens.

Table 3.1: Number of audio clips with gunshots from FSD50K, our in- and outdoor recordings, and the clips created by mixing gunshot sounds with other types of sounds from FSD50K.

Source	Number of clips
FSD50K	322
Indoor	2,754
Outdoor	315
Mixed	2,388
Total	**5,779**

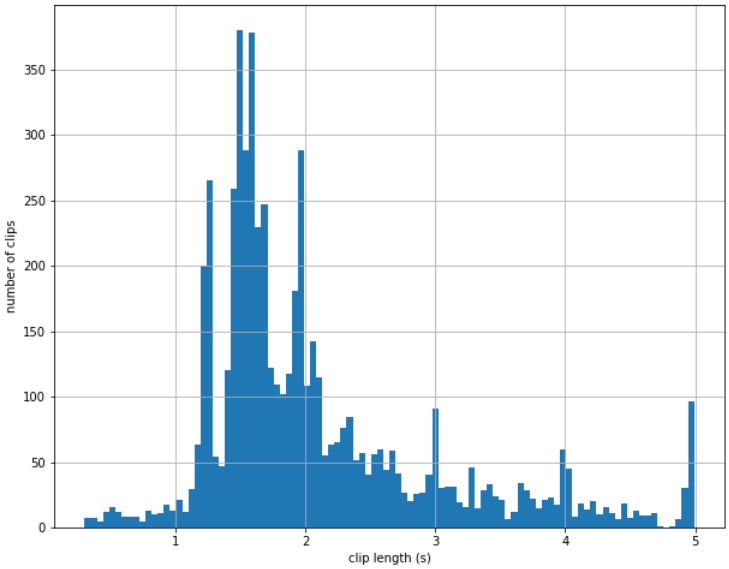


Figure 3.5: Audio clip length distribution for gunshot sounds in the dataset.

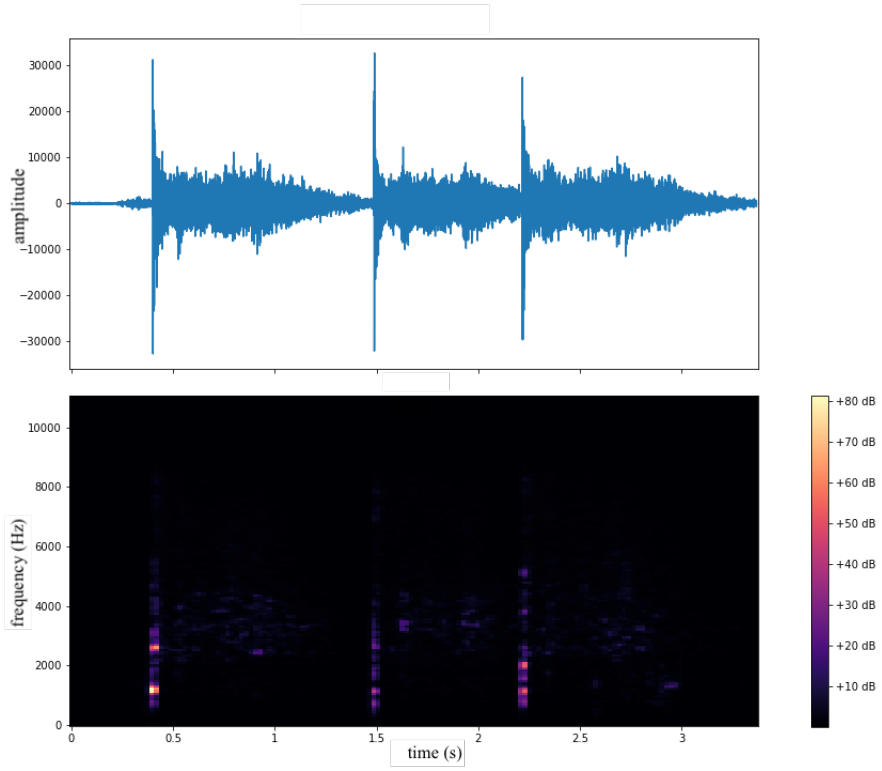


Figure 3.6: Audio wave and mel spectrogram of one of our indoor recordings with three shots fired after each other. The colors in the mel spectrogram indicate the amplitude of a certain frequency at a specific time.

Chapter 4
Approach

Previous research in the area of sound classification has shown that convolutional neural networks (CNNs) which take MFCCs as input can be used to achieve high scores in performance. In addition to this, Cerutti et al. [6] have shown that the complexity of a CNN can potentially be reduced by applying knowledge distillation, which was previously discussed in Section 2.3.

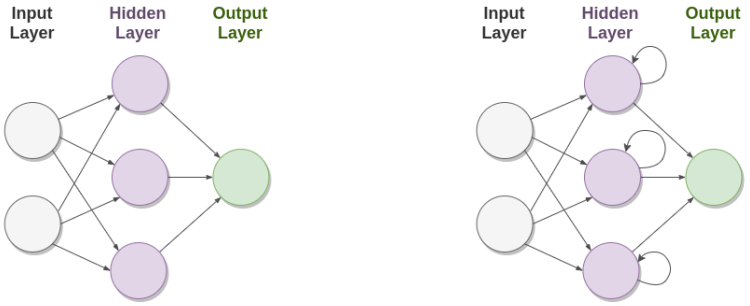
Furthermore, Cerutti et al. [7] state that one common method to create low complexity models is to use smaller networks specifically designed for the task and hardware. Following this strategy, we decided to start off by implementing a rather small CNN which would serve as a baseline. More specifically, we decided to use the design suggested by Baliram et al. [4] as our baseline, as they have proved that this network works well for identifying gunshots. In the following chapter, different tools, techniques, and methods of assessment that were used in our approach will be described in further detail. The last section of the chapter gives an overview of the technical details of our target device, the BWC.

4.1 Artificial Neural Networks

One way of tackling complex machine learning tasks that require processing of substantial amounts of data is by using deep learning, or more specifically, *artificial neural networks* (ANNs). Zhang [37] explains that ANNs are inspired by the human nervous system, and consist of a large number of nodes (also called artificial neurons, or simply neurons) with the ability to receive input signals, process them and generate an output signal. The inputs to an ANN are some measurements or properties, and the output is usually a classification or prediction. As an example, assume the network's task is to predict if it is going to rain today. Then the input could be today's date, whether it was raining yesterday and if it is cloudy. The output could be a number indicating if the network predicts that it is going to rain.

There exist two rather distinct categories of network architectures, and these are *multilayer feedforward neural networks* (MFNN) and *recurrent neural networks* (RNN) [37]. The MFNN consists of nodes arranged in different layers. More specifically, it consists of one input layer of source nodes, one or more hidden layers, and lastly an output layer of nodes. A node in a layer has connections (also called edges) to the nodes in the previous layer, and the different connections may

have different weights or strengths, which affects the way the node processes the input data. During training, the weights are modified to fit the training data, and this is how the network "learns". As the name implies, data is only sent forward, meaning that there are no edges between nodes in the same layer [37]. This can be seen in Figure 4.1.



(a) A feedforward neural network (MFNN). **(b)** A recurrent neural network (RNN).

Figure 4.1: Comparison of an MFNN and an RNN.

Recurrent neural networks are constructed in the same way as feedforward neural networks, with the difference that RNNs contain at least one self-feedback loop [37]. The feedback loops consist of connections that span over adjacent time steps, allowing the model to consider the notion of time. According to Lipton et al. [25], the feedback loops make RNNs powerful since they allow them to process sequences of data. The feedback loops make it possible to take the order of, and dependencies between, the samples into consideration. This makes RNNs especially suitable for processing data samples that may depend on each other, such as video frames, audio snippets, and words [25].

4.1.1 Convolutional neural networks

A special type of feedforward neural network is the *convolutional neural network* (CNN). Commonly used for image recognition, CNNs divide an image into smaller areas and subsequently extract features from every frame [2]. Examining frames rather than separate pixels allows for analysis that retains spatial information. Albawi et al. [2] further describe that to reduce the risk of missing something, there should be an overlap between the extracted frames. This way, the placement of interesting objects in an image does not matter. Various filters (or convolution matrices) can then be used on the frames in different layers to extract features that will be used in the learning process. Figure 4.2 shows an overview of the typical architecture of a CNN. According to Dörfler et al. [10], CNNs can be applied to audio, if the audio is preprocessed properly. One of the most common preprocessing techniques is to convert the audio files into mel spectrograms that are then fed as input to the network. In the report, they examine an alternative to the mel spectrograms and show that the mel-frequency cepstral coefficients (MFCCs) can also be collected by using adaptive filters for particular window lengths.

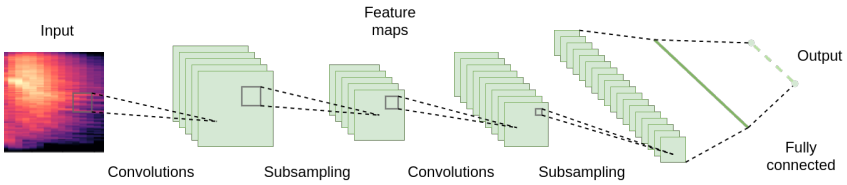


Figure 4.2: The general structure of a convolutional neural network (CNN).

4.1.2 Machine learning tool: TensorFlow

TensorFlow is a reliable open-source platform which can be used to implement machine learning models. TensorFlow provides an abundance of different tools, with one high-level abstraction alternative for constructing ML models being the Keras API [1]. Furthermore, TensorFlow offers a tool (TF Lite converter) that compresses trained models into more lightweight versions, better suitable for mobile devices. The generated TF Lite model consists of an optimized FlatBuffer. TF Lite includes tools for further reducing the size and execution time of models, but this may reduce the accuracy as well.

4.2 Feature Extraction from Audio

Virtanen et al. [36, p. 20] describe feature extraction as a way to convert an acoustic signal into a numerical representation more suitable for machine learning algorithms. Gold et al. [14, p. 107] emphasize the importance of a good feature extraction process by stating that in some cases "the pattern classification can be solved trivially if the features are good enough". A good set of features are such that the samples from the same class have a low variability, while the distinction between samples from different classes is high [14].

Feature extraction from audio can be done in numerous ways, but often the same steps are taken: frame blocking, windowing, spectrum calculation, and subsequent analysis [36, p. 20]. This process is pictured in Figure 4.3 and each step will be further explained in the following sections.

In the *frame blocking* step, the signal is divided into segments of fixed size, called *analysis frames* [36, p. 20]. Virtanen et al. state that the length of the analysis frames is commonly between 20 and 60ms and can be overlapping by at least 50%.

When the signals are cut, abrupt changes can occur at the frame boundaries, which can disturb the classification process. To reduce the risk of this, the signal is smoothed in the *windowing* step [36, p. 21].

Before extracting the final features, each windowed analysis frame is transformed into a *spectrum*. The most common practice is to transform the signal into frequency domain using the discrete Fourier transform (DFT) [36, p. 20]. The DFT representation $X[f]$ of a signal $x[n]$ is calculated as

$$X[f] = \sum_{n=-\infty}^{\infty} x[n]e^{i2\pi fn},$$

where $f = \frac{f_s}{2}$ with f_s being the sampling frequency [36, p. 74]. To increase the efficiency of the calculations, this can be implemented using the *fast Fourier transform (FFT)* [36, p. 75].

Finally, features are extracted from each of the spectrum representations of the frames. Humans perceive sound based on the magnitudes of frequency components in a nonlinear way, which is why feature extraction is typically done with *mel-band energies* and *mel-frequency cepstral coefficients (MFCCs)* [36, p. 21]. These techniques provide compact, smooth representations of the spectrum, mimicking the human auditory perception by using non-linear frequency scaling (mel-frequency scaling) and non-linear representations of magnitude [36]. Virtanen et al. [36, p. 81] define MFCCs as

$$\text{mfcc}(t, c) = \sqrt{\frac{2}{M_{\text{mfcc}}}} \sum_{m=1}^{M_{\text{mfcc}}} \log\left(\tilde{X}_m(t)\right) \cos\left(\frac{c(m-\frac{1}{2})\pi}{M_{\text{mfcc}}}\right).$$

The formula gives the inverse discrete transform of the log energy in mel frequency bands, where M_{mfcc} is the number of mel frequency bands, $\tilde{X}_m(t)$ the energy in the mth frequency band and $c \in \{1, 2, ..., M_{\text{mfcc}}\}$ is the index of the coefficient [36, p. 81].

Although MFCC is the overall most common technique for feature extraction from audio spectra, it was first designed for speech recognition. Valero and Alias [33] show that using *Gammatone cepstral coefficients (GTCCs)* can improve the performance of non-speech audio classification models without increasing the computational complexity.

One advantage of using MFCCs as features is that they can be treated as images. Image detection using machine learning is arguably more researched and developed than audio detection and, with MFCCs, we can take advantage of the techniques developed for image detection.

4.3 Classification Assessment Methods

4.3.1 Confusion matrix

Tharwat [32] describes that for a binary classification system, there are four possible prediction outputs that can be represented in a *confusion matrix*, see Figure 4.4.

Suppose the binary classes are called P, for positive class, and N, for negative class. If a sample is correctly classified as positive, the output would be a *True Positive (TP)*. If a positive sample is classified as negative, it would be a *False Negative (FN)*. For a negative sample, accurately classifying it as negative gives a *True Negative (TN)*, while a positive classification would be considered a *False Positive (FP)* [32].

4.3.2 Performance metrics

To evaluate the performance of a classification algorithm, some commonly used metrics are accuracy, sensitivity, specificity, and precision.

Using the terms described in the previous section, Sokolova et al. [30] define the measurements; accuracy, sensitivity, specificity, and precision as follows:

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}. \tag{4.1}$$

The accuracy measures how effective an algorithm is by giving the ratio between correctly classified samples and the total number of samples [30]. Equation 4.1 shows how to calculate the accuracy.

$$sensitivity = \frac{TP}{TP + FN} \qquad specificity = \frac{TN}{TN + FP}. \tag{4.2}$$

Tharwat [32] explains that sensitivity (also recall, true positive rate (TPR) or hit rate) describes the accuracy for positive samples, while the specificity (also true negative rate (TNR), or inverse recall) describes the accuracy for negative samples. They represent the proportion of positive (sensitivity) and negative (specificity) samples that have been correctly classified. These can be calculated as specified in 4.2.

$$precision = \frac{TP}{TP + FP}. \tag{4.3}$$

Precision is used to measure the performance of the algorithm's prediction, for computation see Equation 4.3. This is done by calculating the proportion between true positives and the total amount of positive predictions [30].

4.3.3 F1-score

$$F_1\text{-}score = \frac{2TP}{2TP + FP + FN}. \tag{4.4}$$

The F_1-score (also F-measure) is the harmonic mean between the sensitivity and the precision and ranges from 0 to 1 [32] and is calculated as in Equation 4.4. An F_1-score close to 1 indicates a high performance.

To gain an understanding of the impact of frequency for a label in the test set, one can plot the relation between label frequency and the label's F_1-score. This is done by Lewis et al. [23]. They use a local linear regression to approximate the relationship between the two aspects and use the resulting plots to compare different classifier methods.

4.3.4 Mean average precision

Following Fonseca et al. [11], we will also use the mean average precision, mAP, to measure the performance of our model. mAP is calculated as the mean across classes of the average precision (AP) [18]. The AP is a common way to summarize the precision-recall curve [11] which is achieved by plotting the precision versus the recall for a varying threshold. Usually, multilabel and binary models output

a score between 0 and 1 for a given class, and the threshold is the decision point for positive or negative classification. As an example, if the threshold is set to 0.5 and the model outputs 0.7 for the label 'Gunshot and gunfire', it is interpreted that the clip includes a gunshot.

4.4 Axis Body Worn Camera

The gunshot detection prototype produced in this project was primarily designed for the Axis Body Worn Camera (BWC), model W100, and tested on it. It is a wireless, wearable camera, primarily intended to increase security by offering portable surveillance to the user. Depending on the use case, it may serve to provide evidence, documentation, protection, and more. Figure 4.5 shows a picture of the BWC.

The BWC has the following technical specifications:

Processor type: ARM Cortex-A53 (1.0 GHz)

Number of cores: 4

Memory card size: 64 GB

DRAM capacity: 512 MB

Flash memory: 256 MB

Audio encoding: AAC-LC 48kHz 64/128 kbps (mono/stereo)

Typical power consumption: 1w

Battery: Li-ion, 3600 mAh

Operating system: Linux

The user can activate an audio and video recording. In addition, s/he can optionally enable a prebuffer functionality, where the buffer may have a size of up to 90 seconds. In this case, this means that the camera is always recording, and keeps the last 90 seconds of recording. Thus, when a recording is started, the content in the prebuffer will be added to the beginning of the video.

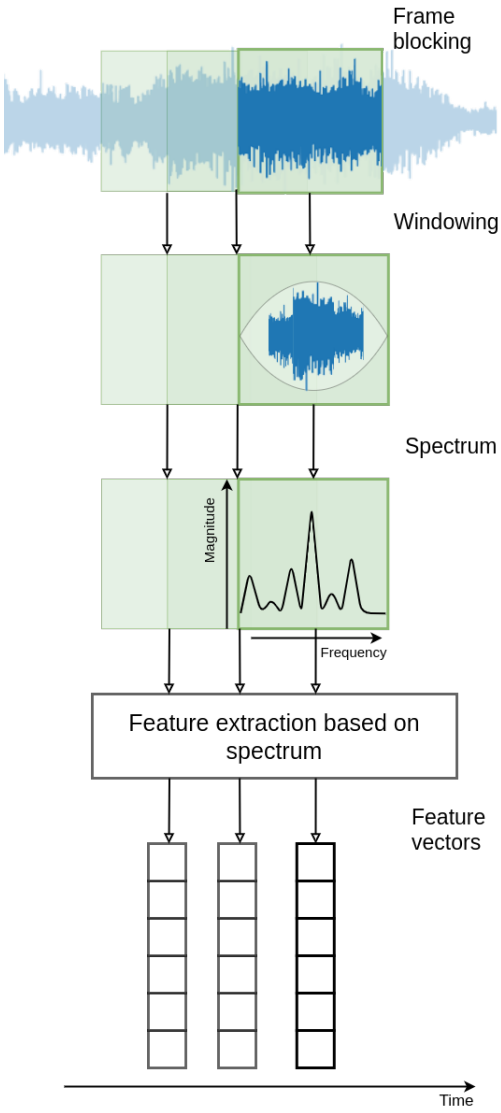


Figure 4.3: The pipeline commonly used for feature extraction from audio.

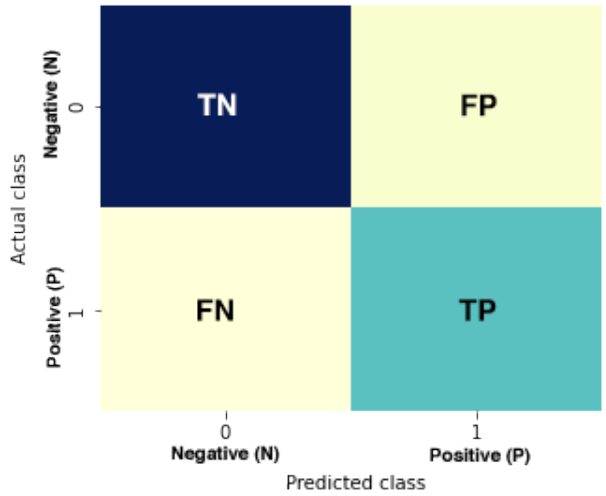


Figure 4.4: Figure showcasing the 4 possible prediction outcomes in a so-called confusion matrix.

Figure 4.5: Axis Body Worn Camera (BWC).

Chapter 5
Experimental Setup

5.1 Networks

We implemented a CNN similar to the one Baliram et al. [4] describe, and this network was intended to serve as the baseline for our project. The network uses MFCCs as features and consists of four convolutional layers. We chose this architecture as it is simple to implement, has relatively few parameters, and Baliram et al. [4] show that it performs well on gunshot detection. Furthermore, we assumed that using a CNN is suitable for the task, as we found that many state-of-the-art solutions for audio classification use this technique.

Each of the convolutional layers consists of a 2D convolution, a max-pooling and a drop-out layer to decrease overfitting. The convolutional layers have 16, 32, 64, and 128 filters respectively of size 3x3, and all drop-out layers use a drop-out rate of 0.2. Figure 5.1 shows a schematic overview of the network structure. We implemented the model in Keras and later converted the model to TensorFlow Lite.

In addition to the baseline model, we implemented a network similar to the one Morehead et al. [27] designed. This is a CNN that takes pulse-code modulation (PCM) data as input, structured as a 240,000x1 vector. The network has four convolutional layers as well, but each layer consists of two 1D convolutions followed by a max-pooling and a drop-out layer, see Figure 5.1. We will refer to this network as the 1D-network.

During training, we used the official division of training, validation, and evaluation sets provided for the audio clips in FSD50K. For the audio clips, we collected on our own, we randomly divided the audio into three parts; 50% for training, 25% for validation, and 25% for testing and used the same division throughout the project. All clips shorter than 5 seconds were padded with zeroes such that all input data had the same length.

5.2 Multilabel Classification

To compare our baseline model with the existing state-of-the-art solutions for FSD50K, we set up the network to be a multilabel classifier. The classifier consists of 200 binary classifiers, one for each label, that output a score between 0 and 1.

Figure 5.1: Overview of the two networks we implemented; the baseline model to the left, and the 1D-network to the right.

A high score for a specific class means that the input is considered likely to include sounds from this class.

We trained the model on the data from FSD50K described in Section 3.3 for 40 epochs, with a learning rate of 10^{-4}, and a batch size of 12. We extracted MFCCs from the audio clips with window length 2048 and hop length 512, i.e. 75 % overlap. For each window, we extract 40 MFCCs. This results in an MFCC matrix of dimension 214x40 and a network with 98,313 trainable parameters. We applied a threshold of 0.3, meaning that clips get tagged with all labels that receive a score of 0.3 or higher. If none of the labels has a score above 0.3, we tag the clip with the label with the highest score.

To investigate what sounds the model easily confused with gunshots, we collapsed some classes into larger groups. To begin with, we collapsed all subclasses of Music into their superclass (Music). The FSD50K has a wide variety of labels

for different music instruments, and by collapsing this group into one, the remaining amount of labels was reduced to 163. We trained the model on this data in the same way as previously. We continued to collapse different classes with their subclasses such as 'Vehicle', 'Liquid', and 'Bird', trained and analyzed the models' performance. Lastly, we collapsed all superclasses described in Section 3.3 except 'Gunshot and gunfire' and trained the model on the resulting dataset.

5.3 Binary Classification

After experimenting with multilabel classification, we modified the code to provide binary classification; with the two labels indicating whether a given audio clip contains a gunshot or not. We trained the model on the FSD50K dataset combined with our gunshot dataset using batches of 12, a learning rate of 10^{-4}, and 40 epochs. For the binary classification, we used a window length of 1024 and hop length of 512, i.e. 50 % overlap. For each window, we extracted 40 MFCCs resulting in an MFCC matrix of the dimension 467x40. This network has 97,281 trainable parameters.

To test out this baseline model in a real-time manner, we wrote a program that collected audio from the microphone using PyAudio and sent the PCM data to the model for classification. We then used a speaker to play audio clips from the unseen part of our dataset to test the accuracy of the prediction.

To simplify the implementation of the real-time detection on the BWC, we implemented the feature extraction as a part of the model. Thus, the model takes raw audio signals (PCM) as input, takes care of the feature extraction, and performs the classification. Next, we converted the entire model into a TF Lite model. Using a TensorFlow Lite Interpreter created specifically for Axis' cameras, we were able to run the TF Lite model and make predictions from the PCM input on the BWC.

In the next step, we implemented a program in C/GLib that collected audio in real-time and ran the TF Lite model to generate a prediction, meaning the BWC could be tested in a real-time manner as well. The program was multithreaded, with one thread designated to collect audio and another that runs it through the model. As Figure 5.2 shows, the program continuously collects audio from the microphones, and every 2.5 seconds sends the audio data from the last 5 seconds to the model for prediction.

For the 1D-network, we trained on the same dataset in batches of 32 and with a learning rate of 10^{-4}. We used early stopping during training using a patience of 15, meaning that if the performance on the validation data does not improve for 15 epochs, the training stops. The network had 319,353 trainable parameters.

5.4 Hardware Performance Evaluation Setup

Finally, we measured how the real-time predictions affected the BWC with regard to battery time, memory consumption, and CPU usage as well as the latency from signal to prediction.

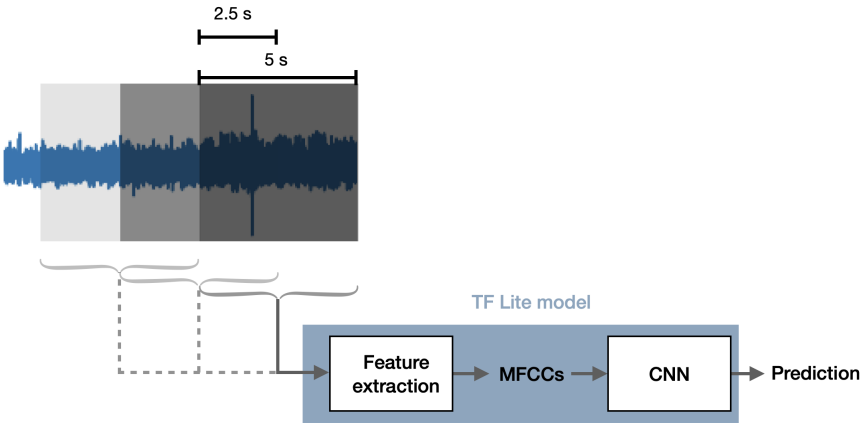


Figure 5.2: Setup of the live detection program, which was implemented on the BWC.

5.4.1 Battery time

To test the real-time prediction's impact on the device's battery time, a BWC was fully charged, started, and then set in a position that would provide similar conditions (darkness and silence) in every test. We then measured the time before the battery ran out. By doing this once with the gunshot detection and once without it, we could calculate how the detection affected the battery time. To get a more generalized result, we did this with three different BWCs and repeated it three times for each mode.

5.4.2 CPU usage

To measure the CPU usage of the real-time predictions, we let the program run for ten hours and then calculated how much time the CPU spent on it. The measurement was done once on three different BWCs in the same environment.

5.4.3 Memory usage

To measure the RAM usage, we let the program run for ten hours. We took note of the memory usage of the audio collection and the TF Lite model interpreter at startup, and then once every hour throughout the test. The memory usage was measured using built-in Linux tools, and the properties considered were USS (Unique Set Size) and PSS (Proportional Set Size). The USS reports all memory that is unique to a specific process [26]. In our case, the sum of the USS from both processes gives how much extra memory usage the gunshot detection is contributing with. The PSS is expected to be equal to or larger than USS, as it also includes a proportional part of any resource that is shared with other processes [26]. Furthermore, we measured the size of the TF Lite models. This gives how much persistent memory the models require, but of course also affects the RAM usage as it is loaded during runtime.

5.4.4 Latency

The latency between a gunshot audio signal reaching the camera, and the triggering of a recording depends on a few different things:

1. First off, preprocessing of the audio signal is required to extract the MFCCs which will be sent as input to the gunshot prediction model.
2. Next, the actual prediction will also take some time.
3. Lastly, the frequency of predictions also needs to be taken into consideration. As an example: if a prediction is performed every 2 seconds, and a gunshot is heard at the beginning of this time slot, this will potentially add latency of at most 2 seconds.

To measure the maximum latency of our prototype, we let the BWC run 100 inferences and measured the execution time. We did this for the entire model as well as for the feature extraction and the classifier separately. We then added the potential maximum delay caused by the prediction frequency.

Chapter 6
Results

In this chapter, we present the results we obtained with the two types of classification, as well as with the implementation on the BWC.

6.1 Multilabel Classification

Our multilabel classifier using all the 200 labels reached a macro F1 score (mean F1 between all labels) of 0.197 and mAP of 0.183. Figure 6.1 shows a plot of the number of clips with a certain label in the training set, versus the F1 score for this label for the classifiers with 200, 163, 144, 131, and 120 labels. As mentioned before, the classifier with 163 labels has the same labels as the one with 200 labels, except for all subclasses to 'Music'. In the classifier with 144 labels, we have also collapsed all subclasses to 'Vehicle'. In the classifier with 131 labels, subclasses to 'Water' and 'Liquid' are also collapsed into their respective superclasses and in the one with 120 we have additionally collapsed 'Fowl', 'Thunderstorm', 'Engine', 'Glass', and 'Bird'. What we can see in the plot of these classifiers is that a higher amount of training samples generally gives better performance.

Figure 6.2 shows the results from the classifiers with 7, 9, and 6 labels. The classifiers with 6 labels have only the superclasses; 'Music', 'Human sounds', 'Sounds of things', 'Natural sounds', 'Animal', and 'Source-ambiguous sounds'. This model with only superclasses, i.e. 6 labels, had a mAP of 0.518. In '7' we have separated 'Gunshot and gunfire' from its superclass and in '9' we have divided 'Natural sounds' into 'Wind', 'Fire', 'Thunderstorm', and 'Water', but included 'Water' in the class 'Sounds of things'.

Although our goal with the investigations of different arrangements of the labels was to see if the model was prone to mix up gunshots with some particular sound, we could not find any such patterns. However, one pattern that we did identify was that the classifiers with 6 and 7 labels, often classified 'Natural sounds' as 'Sounds of things'. When looking into this, we realized that there was at least one obvious reason for this. The majority of clips with 'Natural sounds', have some sort of water sound such as 'Rain' or 'Stream' while 'Sounds of things' includes labels such as 'Liquid', 'Bathtub (filling or washing)', and 'Water tap and faucet'.

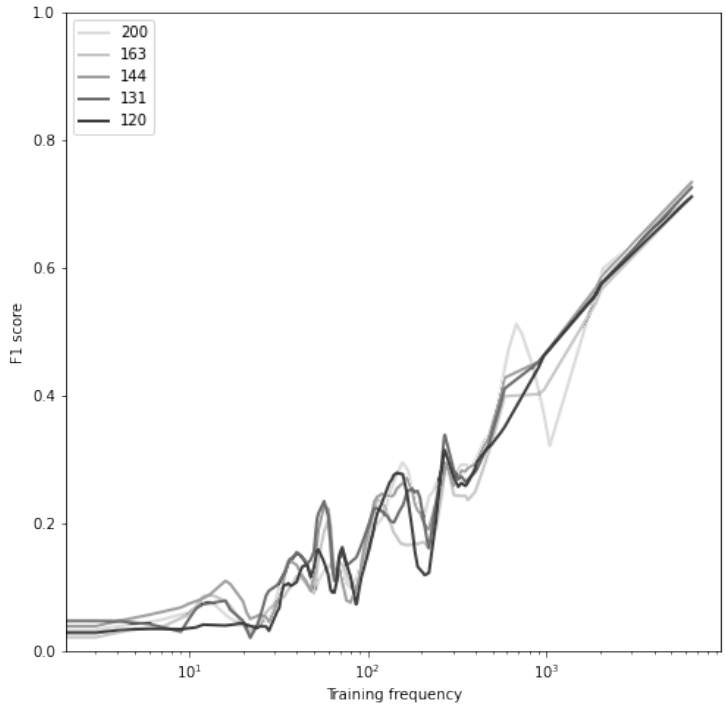


Figure 6.1: Amount of training data versus F1-score per label for the classifiers with 200, 163, 144, 131, and 120 labels.

6.2 Binary Classification

We tested the model on a dataset that included unseen data from FSD50K and the gunshots recordings made by us. Figure 6.3 shows the result after training our baseline model with different subsets of the available data. The best result was achieved when including all available gunshot audio clips. Table 6.1 shows the performance measures of the model when using different thresholds, and Figure 6.5 shows the corresponding confusion matrices. Independent of threshold choice, the average precision of the model is 0.984.

We also tested the model on the different gunshot datasets (FSD50K, indoor, outdoor and mixed) separately. The results from this experiment can be seen in Figure 6.4. The y-axis shows the performance score, while the x-axis gives the training frequency, i.e. the amount of clips in the training set for the given category. As can be seen, the model performs the worst on the gunshots from FSD50K. The clips from FSD50K include a wide variety of gunshots, and due to the multilabel nature of the dataset, most of the clips furthermore contain

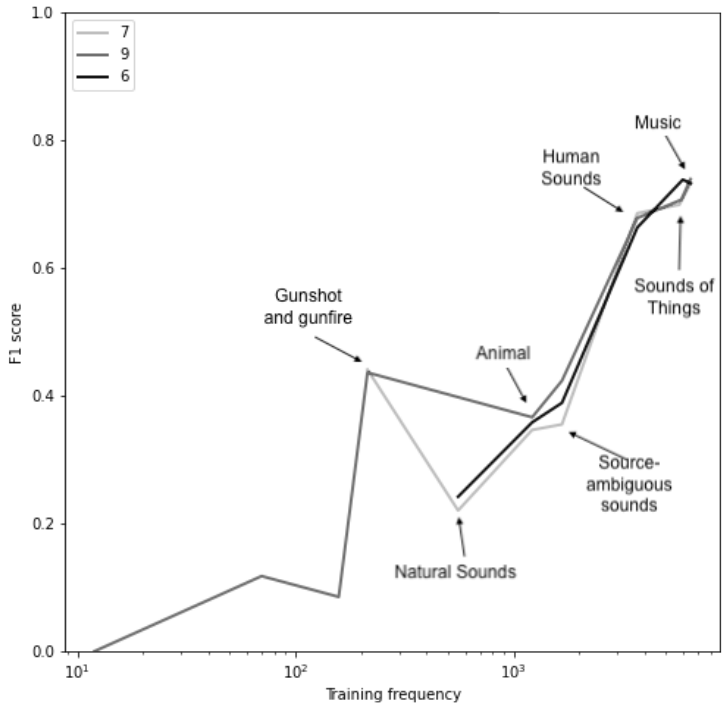


Figure 6.2: Amount of training data versus F1-score per label for the classifiers with 7, 9, and 6 labels.

background sound. The diversity, in combination with the comparatively low training frequency, makes it more difficult for the model to learn to identify this type of gunshots.

The outdoor gunshots do not seem to follow the assumption that a lower training frequency should yield a lower score. This is however most likely due to the outdoor gunshots having similar acoustic features to the indoor gunshots; they were recorded using the same device, with little background sound apart from the wind. Due to this similarity, and the fact that the indoor clips have a very high training frequency, the outdoor clips have an advantage. Apart from this outlier, there seems to be a strong correlation between training frequency and performance.

When testing the model in real-time on the computer, a few different overlaps and audio uptakes frequencies were tested and compared. The combination which produced the best result was using an overlap of 67%, and classifying audio every 0.8 seconds. In other words, every 0.8 seconds, the last 2.4 seconds of audio data was sent to the model for feature extraction and classification.

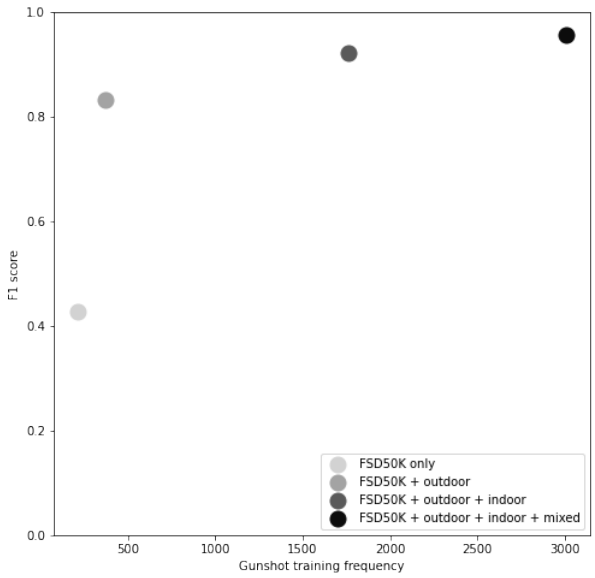


Figure 6.3: Variance in F1 score depending on what training dataset was used.

On the BWC, using the same classification frequency was not feasible due to the feature extraction and prediction needing more than 0.8 seconds to be computed. For this reason, we instead used an overlap of 50% and a classification frequency of 2.5 seconds. This gave the program enough time to compute the gunshot prediction before the next prediction would occur, allowing the gunshot prediction service to run without introducing a steadily increasing delay.

During live testing, the BWC could detect gunshots from the indoors-test set with high accuracy. It did however have a harder time detecting gunshots when there were also other background sounds. We believe this to be due to sending 5-second clips to the model, as we had seen 2.4 second clips produce the best results on mixed gunshot audio while testing different lengths and overlap rates in

Table 6.1: Performance scores for the final model using different thresholds. The highest performance per category is marked in bold.

Threshold	Acc.	Prec.	TPR	TNR	F1
0.3	0.957	0.899	**0.967**	0.952	0.931
0.5	0.968	0.939	0.958	0.972	0.948
0.7	**0.970**	**0.957**	0.947	**0.981**	**0.952**

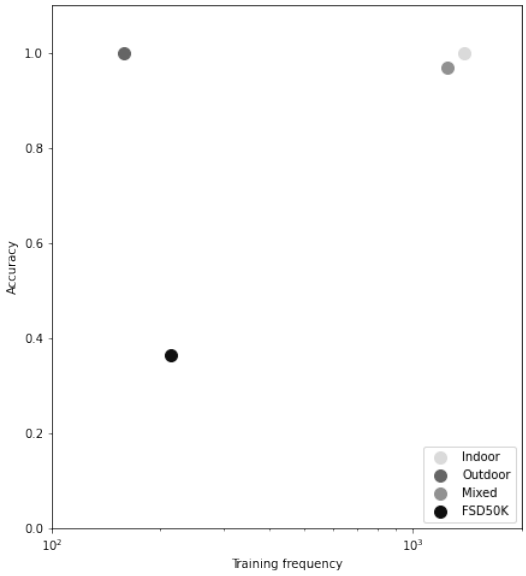


Figure 6.4: Accuracy of gunshot prediction on the different types of gunshots, compared to their training frequency.

the live detection on the computer.

For the 1D-network, we received an average precision of 0.978, and an F1-score of 0.818 when using a threshold of 0.5. The resulting confusion matrix can be seen in Figure 6.6. Due to a long execution time (2.9 s) and lower performance scores, we did not proceed with any further assessments of this model.

6.3 Hardware Performance Evaluation

6.3.1 Battery time

As can be seen in Table 6.2, the gunshot detection lowered the BWCs' battery time by an average of 8.41%.

6.3.2 CPU usage

Table 6.2 shows how much of the CPU capacity the different parts of the live gunshot detection use. In total, the live detection uses 11.5 % of the CPU's four cores' capacity, whereof 10.5 percentage points are used for the feature extraction and classification.

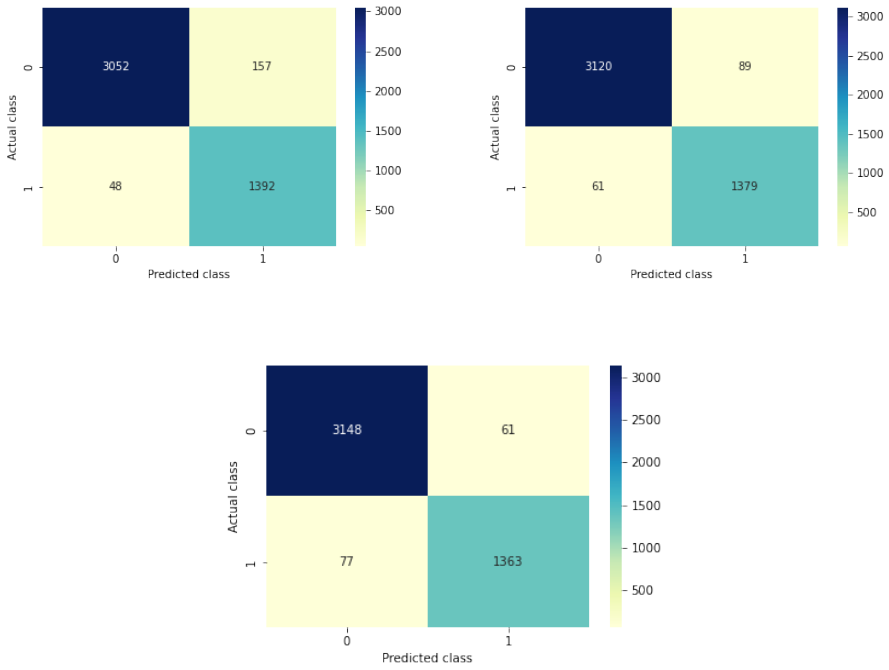


Figure 6.5: The resulting confusion matrices for a threshold of 0.3, 0.5, and 0.7, in that respective order.

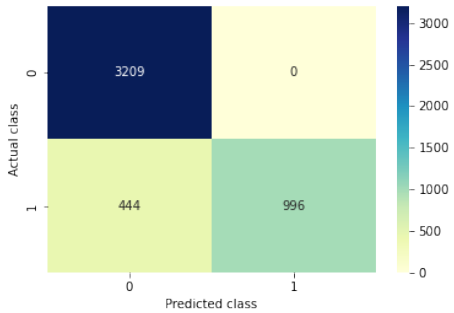


Figure 6.6: Confusion matrix for the 1D-network.

Table 6.2: This table shows the percentage of the CPU capacity the different parts of the gunshot detection uses, as well as its impact on the devices' battery time.

Device	Audio coll.	Preproc. and prediction	Total CPU usage	Difference in battery time
BWC 1	0.97 %	10.54 %	11.51%	-8.94 %
BWC 2	0.98 %	10.50 %	11.48 %	-8.51 %
BWC 3	0.98 %	10.52 %	11.50 %	-7.79 %
Mean:	**0.98 %**	**10.52 %**	**11.50 %**	**-8.41 %**

6.3.3 Memory usage

Figure 6.7 shows the RAM usage of the audio collection and the preprocessing and prediction on each camera at setup and during the preceding ten hours. An average between the cameras for setup and normal run time is shown in Table 6.3b. On average, the real-time detection uses an additional 19.3 MB during setup and 11.9 MB otherwise.

Regarding persistent memory, the size of the model is shown in Table 6.4. The CNN makes up less than 10 % of the total size, while the feature extraction constitutes a clear majority.

Table 6.3: Average RAM memory usage from the 10-hour tests, reported as PSS in (a) and USS in (b). The averages were calculated from measurements during the program startup and then once every hour, starting 1 hour after startup.

(a) PSS

Process	During setup	After 1h
Preproc. and prediction	13.9 MB	10.1 MB
Audio collection	5.4 MB	3.0 MB
Total	19.3 MB	13.2 MB

(b) USS

Process	During setup	After 1h
Preproc. and prediction	13.1 MB	9.6 MB
Audio collection	3.8 MB	2.3 MB
Total	16.9 MB	11.9 MB

6.3.4 Latency

The feature extraction takes up a clear majority of both time and space. By using the classification frequency of 2.5 seconds and adding the mean execution time of the full model (see Table 6.4), the maximum latency is approximately $2.5 + 1.14 = 3.64s$.

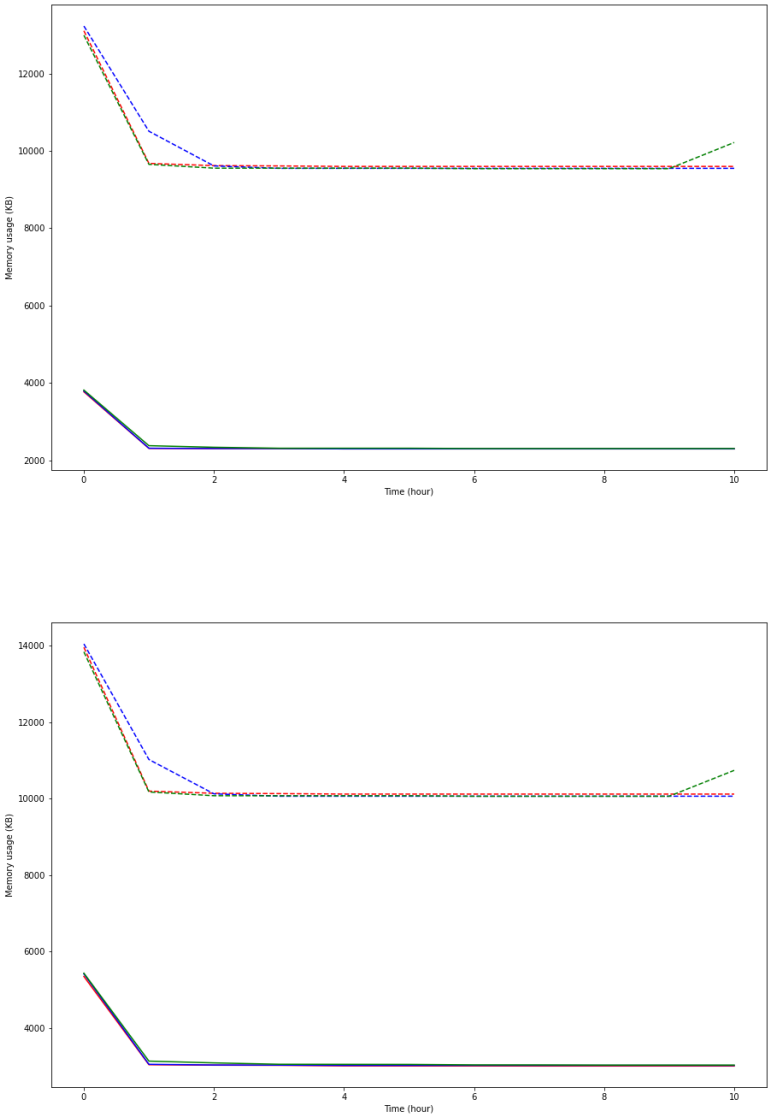


Figure 6.7: RAM memory usage of the gunshot detection program. The top plot shows the USS and the bottom PSS. The dashed lines show the preprocessing and the prediction's memory usage, while the solid lines represent the audio collection. The data was collected once per hour from three different BWCs.

Table 6.4: Size and mean execution time of the TF Lite models.

Model	**Size**	**Mean exec. time**
Baseline (feature extraction)	4.5 MB	958 ms
Baseline (CNN classifier)	0.4 MB	180 ms
Baseline (combined)	4.9 MB	1140 ms
1D-network	1.3 MB	2859 ms

Chapter 7
Discussion

7.1 Classification

The final version of the baseline model received great scores. Depending on what feature is considered to be most important, different thresholds for when to classify an audio clip as a gunshot may be chosen. In Table 6.1, we can see that the higher threshold of 0.7 has the best results in all but one category, so at first sight, it may be the obvious choice.

However, given the potential graveness of the situation where this feature may be used, it might be deemed more important that no gunshot is missed rather than to decrease the number of false alarms. In such a case, it may be more desirable to maximize the sensitivity (TPR), which measures the ratio of correctly guessed actual gunshots. This score received the best result for lower thresholds, such as 0.3.

7.2 Hardware Performance

The results on how the live gunshot detection program affects the BWC's battery time, CPU, and memory usage indicate that such a feature is possible to add without limiting other functionalities.

The battery time is decreased by less than 10% which would still allow usage for a complete workday. Using 11.5% of the BWC's CPU is within reasonable limits, but could potentially cause problems if other features would also require much from the CPU.

The memory usage is increased by 11.9 MB, i.e. approximately 2% of the BWC's RAM. The persistent memory needed for the model, 4.9 MB, is negligible on the BWC's 64 GB memory card. Decreasing the size of the model would of course also decrease the required RAM during execution. If the RAM usage is critical and considered too high, this is one possibility to improve the prototype. However, decreasing the size of the model generally means that the performance will also decrease.

An interesting aspect to examine further, regarding the hardware, is to what extent the increased CPU usage affects the camera's product lifetime.

7.3 Challenges of the Limited Capacity

One of the most limiting capabilities seems to be the long execution time of the feature extraction, which limits the possible classification frequency and eliminates the possibility of using the rate we found to be most efficient at detecting gunshots. Possible solutions to this would be to use a faster CPU or execute the predictions in more than one thread, allowing multiple predictions to be run concurrently. It might also be possible to decrease the execution time of the feature extraction by implementing it hardware-optimized, instead of implementing it in TensorFlow and then converting it to TF Lite.

These changes would furthermore positively affect the maximum latency, as it is bounded by the classification frequency and the prediction speed. If one or both of these variables were changed, the latency could decrease. Since the current goal of the classification is to make the BWC trigger a recording when a gunshot is detected, the calculated maximum delay of 3.64 seconds is not a huge issue due to the BWC's built-in prebuffer. If the classification should be used for something else, such as to alarm or to send a request for reinforcement, decreasing the latency would have a much higher priority.

The 1D-network did not require any preprocessing at all but in return, it had lower performance and the network in itself was larger resulting in a longer execution time. It would be interesting to explore other features to find something that is less complex to compute than MFCCs but still allow small networks to achieve good results.

7.4 Dataset

Adding more gunshot audio clips to the dataset greatly improved the F1-score. Something to keep in mind is however possible bias; as the large majority of the gunshot clips are comprised of either indoor gunshots, or the automatically mixed gunshot/background clips, the model is bound to perform better on audio clips with similar acoustics. This was highlighted when testing the model on gunshot sounds from the different parts of the dataset – the model performed the worst on the gunshot sounds from FSD50K.

A solution to this problem would be to acquire or find more diverse data to add to the training of the model. As described, the dataset currently contains a lot of gunshots from pistols with .22 mm ammunition in a relatively small room. It would be beneficial to add audio clips of gunshots from other types of guns and ammunition, as well as in more varied environments.

Another suggestion could be to use augmentation and sampling, as described for the PSLA network by Gong et al. [16]. The less frequent types of gunshots, such as the outdoor recordings, could be up sampled and augmented such that the indoor recordings are not as dominating. Furthermore, inspired by Morehead et al. [27], Generative Adversarial Networks (GANs) could be used to generate more audio clips.

7.5 Real-life Usage

Although the model performs well on the given dataset, a more comprehensive evaluation would be required to guarantee that the gunshot detection is reliable in a real-life situation. A suggestion for further testing of the real-time detection is to expose it to environments similar to the expected use case.

Should such a test not give sufficiently good results, then we suggest improving the dataset by increasing the diversity of gunshot sounds. As repeated throughout the report, a large and diverse dataset is essential for good performance. For the real-life usage this is even more important as the more diverse the training dataset is, the more likely it will contain sounds that are similar to what will occur in reality.

In addition, adjusting the classification frequency and the length of the signals sent to the gunshot detection could improve the results. When experimenting with different configurations of this on the live detection on a computer, we found that it influenced how well the detection identified gunshots when there was a lot of background noise. We believe this is because a gunshot generates a relatively short sound, a small portion of the 5-second sound signals we send to the classifier. If there are a lot of other sounds during those 5 seconds, a majority of the clip will not be a gunshot and thus over weigh towards a negative prediction.

One aspect that might affect the real-time performance negatively is that the large majority of the audio clips which were used in the model training have not been recorded by BWCs. Although the clips in FSD50K come from a variety of sources, it would presumably be beneficial if the model trained on recordings with similar acoustic properties to the BWC's microphones. A possible solution for this would be to record an entire dataset with the BWC. Alternatively, the sounds from FSD50K could be altered to resemble BWC recordings.

7.6 Multilabel Classification

The small network implemented in this Master's thesis did not compare well to the state-of-the-art solutions for the multilabel task of FSD50K. This was however expected as it is less than 1% the size of PSLA and almost 0.1% the size of PaSST-S. When simplifying the task to only regard the superclasses, the network performed relatively well, especially on the labels with larger training frequencies.

The exploration with the different variations of label combinations underlined the fact that performance is highly dependent on the training frequency. In addition, we saw that well-separated classes are beneficial, with the example of 'Water' and 'Liquid' belonging to different superclasses.

Chapter 8
Conclusion and Future Work

8.1 Conclusion

According to our measurements, it does indeed seem viable to use gunshot detection with a simple CNN on a portable device such as the BWC. While the battery time and memory usage were somewhat affected, it was not considered to be so by an excessive amount. The live detection program's CPU usage was also, depending on what other things need to be executed, within reasonable bounds.

We show that a CNN with four layers and 100,000 parameters is able to identify gunshots in audio signals. Our network achieved an accuracy of 0.970 and F1-score of 0.952. Our conclusion is that larger, more complex networks are not necessary for this specific task, especially not if the target device has limited resources.

In addition, we found that real-time detection of gunshots in small, portable devices is applicable, although it may be difficult to avoid having some latency between the audio signal and the finished detection. We did our implementation and tests on the BWC, but the results can be generalized to devices with similar hardware properties.

Our implementation increased the RAM usage by 11.9 MB, decreased the battery time with 8.4% and uses 11.5 % of the CPU. The latency is approximately 3.6 seconds. We could see that the most demanding part of the detection is to extract the MFCCs from the audio signal, which uses 10.5% of the CPU and takes more than 5 times longer than the prediction.

8.2 Future Work

As the preprocessing, i.e. extracting MFCCs, was the most demanding part of our gunshot detection, we think this is the most interesting area to investigate further. Both by looking at how the extraction can be optimized for the hardware and by exploring other features.

If the MFCCs are extracted separately, instead of as a part of the TF Lite model, we believe that there is a potential improvement in efficiency. Some portable devices have specific hardware particularly fast for calculations related to audio processing. In these cases, it would most certainly be beneficial to do the extraction separately. However, the BWC does not have such a chip, but it might still be possible to increase the computational cost of the extraction.

By decreasing the number of mel bands, the frame size, and the overlap used when extracting the MFCCs, it might be possible to decrease the execution time without decreasing classification performance. Additionally, it would be interesting to investigate other, less complex, features that could be used.

Another area that is interesting to examine further is the dataset. Is it beneficial to use recordings that are recorded on the target device? Would the classification performance improve if the mixed clips were created manually? Is there an optimal division between pure gunshot clips and mixed clips? Would it be better to have fewer gunshots from the same environment (the indoor recordings), i.e. to use less data in order to decrease the bias? It would also be interesting to see how our model performs if we had a greater variety of gunshot sounds.

We hope that our work will encourage research on sound event detection on portable devices, where the hardware performance is evaluated and disclosed. Moreover, we hope that our results can inspire the development of efficient machine learning programs for devices with limited resources to make them accessible for embedded usage, as this study proves it can be done.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[3] Jakub Bajzik, Jiri Prinosil, and Dusan Koniar. Gunshot detection using convolutional neural networks. In *2020 24th International Conference Electronics*, pages 1–5, 2020.

[4] Rajesh Baliram Singh, Hanqi Zhuang, and Jeet Kiran Pawani. Data collection, modeling, and classification for gunshot and gunshot-like audio events: a case study. *Sensors*, 21(21):7320, 2021.

[5] Gianmarco Cerutti, Renzo Andri, Lukas Cavigelli, Elisabetta Farella, Michele Magno, and Luca Benini. Sound event detection with binary neural networks on tightly power-constrained iot devices. *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, Aug 2020.

[6] Gianmarco Cerutti, Rahul Prasad, Alessio Brutti, and Elisabetta Farella. Neural network distillation on iot platforms for sound event detection. In *Interspeech*, pages 3609–3613, 2019.

[7] Gianmarco Cerutti, Rahul Prasad, Alessio Brutti, and Elisabetta Farella. Compact recurrent neural networks for acoustic event detection on low-energy low-complexity platforms. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):654–664, 2020.

[8] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1, 2016.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[10] Monika Dörfler, Roswitha Bammer, and Thomas Grill. Inside the spectrogram: Convolutional neural networks in audio processing. In *2017 International Conference on Sampling Theory and Applications (SampTA)*, pages 152–155, 2017.

[11] Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, and Xavier Serra. Fsd50k: an open dataset of human-labeled sound events. *arXiv preprint arXiv:2010.00475*, 2020.

[12] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. IEEE, 2017.

[13] Andrzej Glowacz and Grzegorz Altman. Automatic threat classification using multiclass svm from audio signals. In *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA 2012)*, pages 1–6, 2012.

[14] Ben Gold, Nelson Morgan, and Dan Ellis. *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*. John Wiley & Sons Inc., United States, 1st edition, 1999.

[15] Yuan Gong, Yu-An Chung, and James Glass. Ast: Audio spectrogram transformer, 2021.

[16] Yuan Gong, Yu-An Chung, and James Glass. PSLA: Improving audio tagging with pretraining, sampling, labeling, and aggregation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3292–3306, 2021.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[18] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson. Cnn architectures for large-scale audio classification, 2017.

[19] Noussaiba Jaafar and Zied Lachiri. Audio-visual fusion for aggression detection using deep neural networks. In *2019 International Conference on Control, Automation and Diagnosis (ICCAD)*, pages 1–5, 2019.

[20] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks, 2017.

[21] J.F.P. Kooij, M.C. Liem, J.D. Krijnders, T.C. Andringa, and D.M. Gavrila. Multi-modal human aggression detection. *Computer Vision and Image Understanding*, 144:106–120, 2016. Individual and Group Activities in Video Event Analysis.

[22] Khaled Koutini, Jan Schlüter, Hamid Eghbal-zadeh, and Gerhard Widmer. Efficient training of audio transformers with patchout, 2021.

[23] David D Lewis, Yiming Yang, Tony Russell-Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

[24] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network, 2017.

[25] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[26] Matt Mackall. Smem(8) – linux manual page, 2008.

[27] Alex Morehead, Lauren Ogden, Gabe Magee, Ryan Hosler, Bruce White, and George Mohler. Low cost gunshot detection using deep learning on the raspberry pi. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 3038–3044, 2019.

[28] Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. A dataset and taxonomy for urban sound research. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1041–1044, 2014.

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[30] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence*, pages 1015–1021. Springer, 2006.

[31] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[32] Alaa Tharwat. Classification assessment methods. *Applied Computing and Informatics*, 2020.

[33] Xavier Valero and Francesc Alias. Gammatone cepstral coefficients: Biologically inspired features for non-speech audio classification. *IEEE Transactions on Multimedia*, 14(6):1684–1689, 2012.

[34] P.W.J. van Hengel and T.C. Andringa. Verbal aggression detection in complex social environments. In *2007 IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 15–20, 2007.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

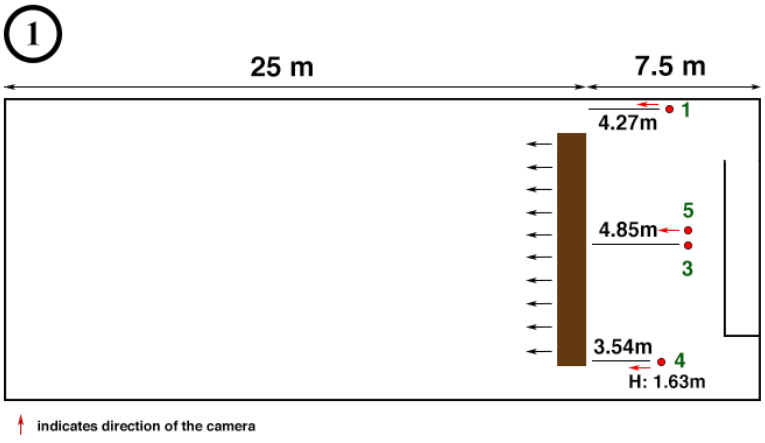
[36] Tuomas Virtanen, Mark D Plumbley, and Dan Ellis. *Computational analysis of sound scenes and events*. Springer, 2018.

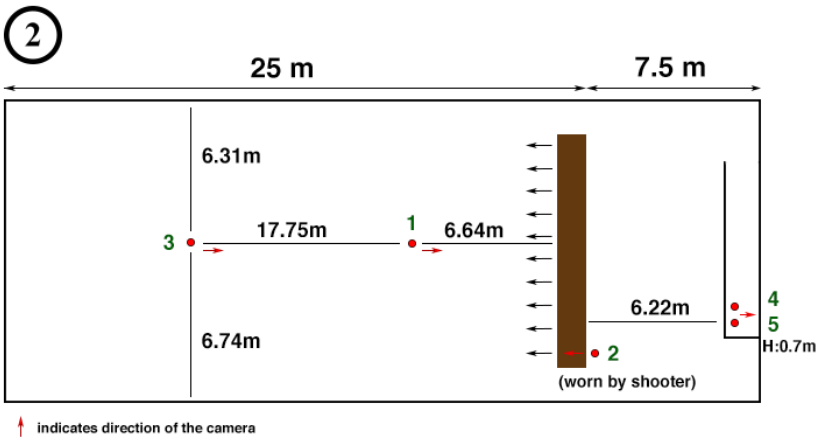
[37] Zhihua Zhang. Artificial neural network. In *Multivariate time series analysis in climate and environmental research*, pages 1–35. Springer, 2018.

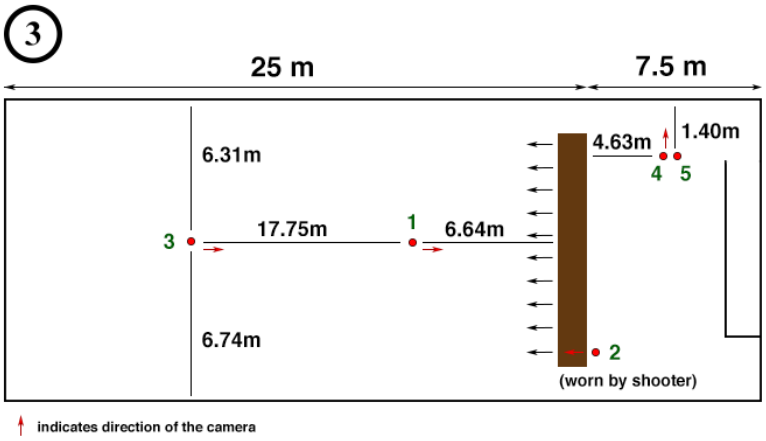
Appendix A

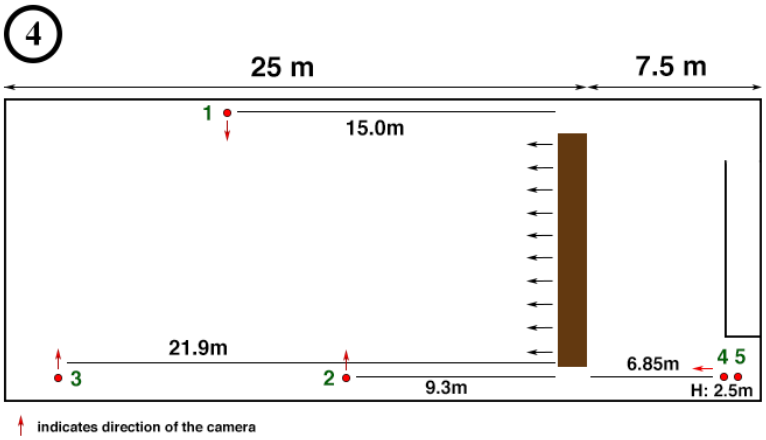
Indoor Recording Setups

This section includes drawings of the different setups used when recording gunshot audio indoors, as described in Section 3.2.1. The red dots listed as 1, 2, 3 and 4 are BWCs, while 5 is a cellphone with a microphone attached to it.






3
25 m
7.5 m
6.31m
4.63m
1.40m
4
5
3
17.75m
1
6.64m
6.74m
2
(worn by shooter)
indicates direction of the camera


4
25 m
7.5 m
1
15.0m
21.9m
3
2
9.3m
6.85m
4 5
H: 2.5m
indicates direction of the camera

Appendix B

More Information About FSD50K

Table B.1: The table shows all labels in the FSD50K dataset with the number of audio clips in the subset we used containing the sound. The table also lists which AudioSet superclasses each label belongs to. The superclasses are *Sounds of things* (**SOT**), *Music* (**M**), *Animal* (**A**), *Human sounds* (**HS**), *Source-ambiguous sounds* (**SAS**), and *Natural sounds* (**NS**)

Label	**Count**	**Superclass**
Accelerating and revving and vroom	275	SOT
Accordion	102	M
Acoustic guitar	672	M
Aircraft	272	SOT
Alarm	1864	SOT
Animal	4357	A
Applause	550	HS
Bark	536	A
Bass drum	339	M
Bass guitar	400	M
Bathtub (filling or washing)	204	SOT
Bell	1184	SOT, M
Bicycle	311	SOT
Bicycle bell	129	SOT, M
Bird	1729	A
Bird vocalization and bird call and bird song	961	A
Boat and Water vehicle	139	SOT
Boiling	102	SOT
Boom	204	SOT
Bowed string instrument	2064	M
Brass instrument	1035	M
Breathing	658	HS
Burping and eructation	311	HS
Bus	285	SOT
Buzz	156	SAS, A
Camera	376	SOT

Car	999	SOT
Car passing by	232	SOT
Cat	435	A
Chatter	454	HS
Cheering	326	HS
Chewing and mastication	256	HS
Chicken and rooster	173	A
Child speech and kid speaking	255	HS
Chime	268	SOT, M
Chink and clink	433	SOT
Chirp and tweet	548	SAS, A
Chuckle and chortle	111	HS
Church bell	141	SOT, M
Clapping	566	HS
Clock	349	SOT
Coin (dropping)	529	SOT
Computer keyboard	223	SOT
Conversation	126	HS
Cough	385	HS
Cowbell	235	SOT, M, A
Crack	187	SOT, SAS
Crackle	327	SAS, NS
Crash cymbal	285	M
Cricket	276	A
Crow	111	A
Crowd	435	HS
Crumpling and crinkling	291	SAS
Crushing	276	SAS
Crying and sobbing	151	HS
Cupboard open or close	153	SOT
Cutlery and silverware	400	SOT
Cymbal	903	M
Dishes and pots and pans	479	SOT
Dog	930	A
Domestic animals and pets	1365	A
Domestic sounds and home sounds	6658	SOT
Door	1522	SOT
Doorbell	144	SOT
Drawer open or close	220	SOT
Drill	211	SOT
Drip	331	SOT
Drum	1514	M
Drum kit	401	M
Electric guitar	687	M
Engine	1408	SOT
Engine starting	220	SOT
Explosion	1388	SOT

Fart	630	HS
Female singing	207	HS
Female speech and woman speaking	640	HS
Fill (with liquid)	148	SOT
Finger snapping	199	HS
Fire	509	NS
Fireworks	469	SOT
Fixed-wing aircraft and airplane	146	SOT
Fowl	281	A
Frog	110	A
Frying (food)	123	SOT
Gasp	110	HS
Giggle	188	HS
Glass	1241	SOT
Glockenspiel	143	M
Gong	278	M
Growling	123	A
Guitar	2232	M
Gull and seagull	104	A
Gunshot and gunfire	482	SOT
Gurgling	234	NS
Hammer	234	SOT
Hands	870	HS
Harmonica	223	M
Harp	234	M
Hi-hat	604	M
Hiss	289	SAS, A, NS
Human group actions	1423	HS
Human voice	5653	HS
Idling	213	SOT
Insect	598	A
Keyboard (musical)	1804	M
Keys jangling	251	SOT
Knock	373	SOT, SAS
Laughter	1186	HS
Liquid	1601	SOT
Livestock and farm animals and working animals	635	A
Male singing	150	HS
Male speech and man speaking	858	HS
Mallet percussion	427	M
Marimba and xylophone	210	M
Mechanical fan	119	SOT
Mechanisms	1534	SOT
Meow	250	A
Microwave oven	206	SOT
Motor vehicle (road)	1986	SOT
Motorcycle	250	SOT

Music	14739	M
Musical instrument	14703	M
Ocean	332	NS
Organ	348	M
Packing tape and duct tape	130	SOT
Percussion	3977	M
Piano	855	M
Plucked string instrument	2350	M
Pour	323	SOT
Power tool	214	SOT
Printer	163	SOT
Purr	128	A
Race car and auto racing	123	SOT
Rail transport	836	SOT
Rain	708	NS
Raindrop	194	NS
Ratchet and pawl	111	SOT
Rattle	300	SAS, A
Rattle (instrument)	317	M
Respiratory sounds	1202	HS
Ringtone	169	SOT
Run	332	HS
Sawing	173	SOT
Scissors	165	SOT
Scratching (performance technique)	309	M
Screaming	377	HS
Screech	154	SAS
Shatter	510	SOT
Shout	393	HS
Sigh	136	HS
Singing	717	HS
Sink (filling or washing)	407	SOT
Siren	132	SOT
Skateboard	135	SOT
Slam	552	SOT
Sliding door	295	SOT
Snare drum	886	M
Sneeze	125	HS
Speech	2254	HS
Speech synthesizer	120	HS
Splash and splatter	518	SOT
Squeak	606	SOT, SAS
Stream	317	NS
Strum	233	M
Subway and metro and underground	387	SOT
Tabla	97	M
Tambourine	300	M

Tap	384	SOT, SAS
Tearing	386	SAS
Telephone	685	SOT
Thump and thud	470	SOT, SAS
Thunder	583	NS
Thunderstorm	601	NS
Tick	109	SOT, SAS
Tick-tock	170	SOT
Toilet flush	279	SOT
Tools	1004	SOT
Traffic noise and roadway noise	279	SOT
Train	449	SOT
Trickle and dribble	304	SOT
Truck	154	SOT
Trumpet	637	M
Typewriter	102	SOT
Typing	430	SOT
Vehicle	3710	SOT
Vehicle horn and car horn and honking	183	SOT
Walk and footsteps	580	HS
Water	1925	NS
Water tap and faucet	458	SOT
Waves and surf	245	NS
Whispering	245	HS
Whoosh and swoosh and swish	372	SOT, SAS
Wild animals	2392	A
Wind	428	NS
Wind chime	119	SOT, M
Wind instrument and woodwind instrument	2653	M
Wood	432	SOT
Writing	343	SOT
Yell	199	HS
Zipper (clothing)	395	SOT

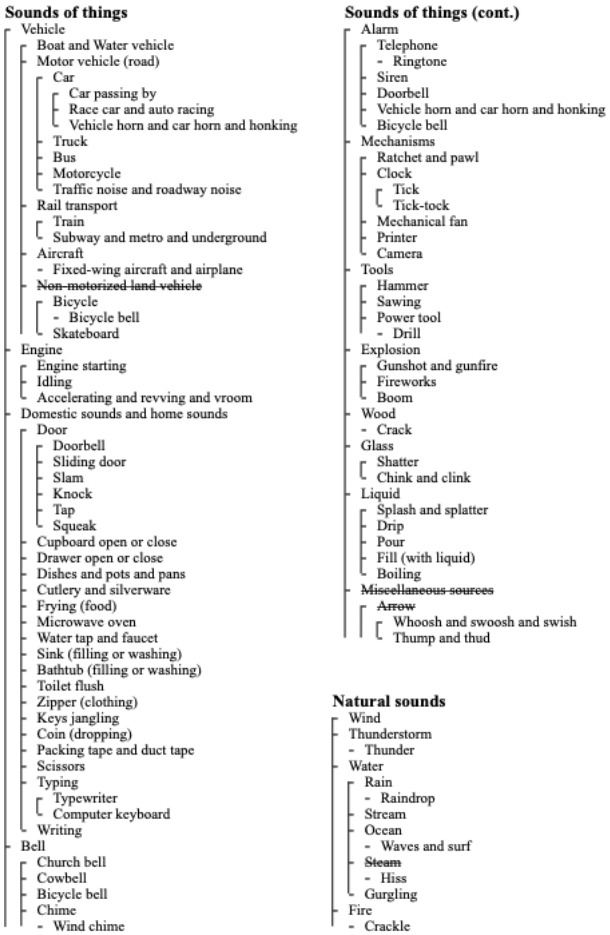


Figure B.1: The labels in FSD50K ordered according to the AudioSet ontology. Labels with strike-through are not included in FSD50K, but have subclasses that are. This is shown in Figure B.1-3.

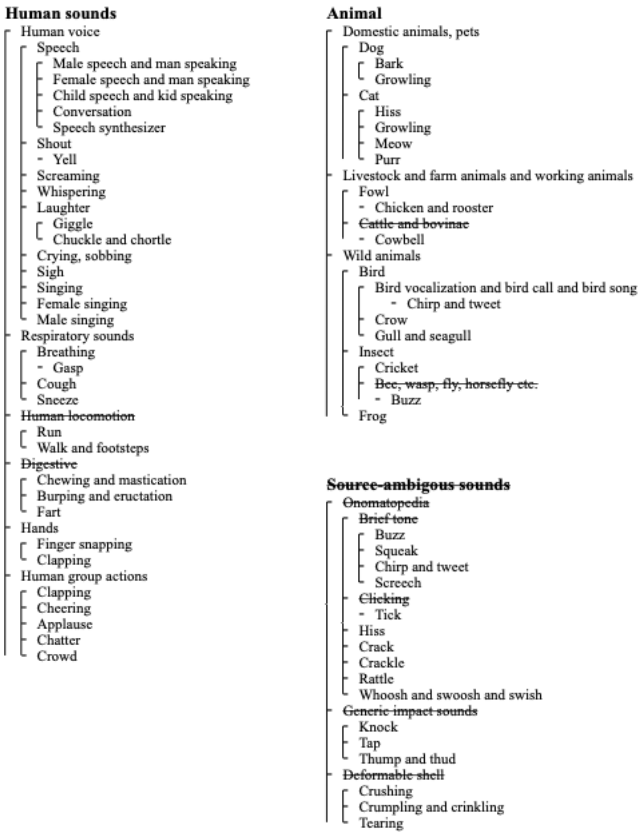

Human sounds
Human voice
Speech
Male speech and man speaking
Female speech and man speaking
Child speech and kid speaking
Conversation
Speech synthesizer
Shout
Yell
Screaming
Whispering
Laughter
Giggle
Chuckle and chortle
Crying, sobbing
Sigh
Singing
Female singing
Male singing
Respiratory sounds
Breathing
Gasp
Cough
Sneeze
Human locomotion
Run
Walk and footsteps
Digestive
Chewing and mastication
Burping and eructation
Fart
Hands
Finger snapping
Clapping
Human group actions
Clapping
Cheering
Applause
Chatter
Crowd
Animal
Domestic animals, pets
Dog
Bark
Growling
Cat
Hiss
Growling
Meow
Purr
Livestock and farm animals and working animals
Fowl
Chicken and rooster
Cattle and bovinae
Cowbell
Wild animals
Bird
Bird vocalization and bird call and bird song
Chirp and tweet
Crow
Gull and seagull
Insect
Cricket
Bee, wasp, fly, horsefly etc.
Buzz
Frog
Source-ambigous sounds
Onomatopedia
Brief tone
Buzz
Squeak
Chirp and tweet
Screech
Clicking
Tick
Hiss
Crack
Crackle
Rattle
Whoosh and swoosh and swish
Generic impact sounds
Knock
Tap
Thump and thud
Deformable shell
Crushing
Crumpling and crinkling
Tearing

Figure B.2

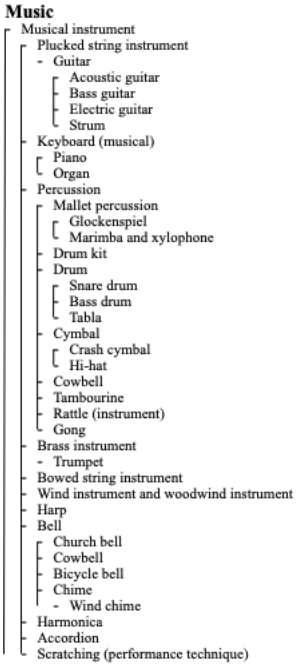

Music
Musical instrument
Plucked string instrument
Guitar
Acoustic guitar
Bass guitar
Electric guitar
Strum
Keyboard (musical)
Piano
Organ
Percussion
Mallet percussion
Glockenspiel
Marimba and xylophone
Drum kit
Drum
Snare drum
Bass drum
Tabla
Cymbal
Crash cymbal
Hi-hat
Cowbell
Tambourine
Rattle (instrument)
Gong
Brass instrument
Trumpet
Bowed string instrument
Wind instrument and woodwind instrument
Harp
Bell
Church bell
Cowbell
Bicycle bell
Chime
Wind chime
Harmonica
Accordion
Scratching (performance technique)

Figure B.3

Printed by Tryckeriet i E-huset, Lund 2022

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2022-873
http://www.eit.lth.se