Degree Project in Mathematical Statistics
Second Cycle, 30 Credits

Detection and Classification of Sparse Traffic Noise Events
A Machine Learning Based Approach Implementing Supervised and Unsupervised Models

KEVIN GOLSHANI
ELIAS EKBERG

Stockholm, Sweden 2023
Detection and Classification of Sparse Traffic Noise Events

A Machine Learning Based Approach
Implementing Supervised and Unsupervised Models

Kevin Golshani, Elias Ekberg
Supervisors: Maurice Duits, Romain Rumpler
Examiner: Maurice Duits

MWL Laboratory for Sound and Vibration Research
Department of Mathematics
KTH, Royal Institute of Technology
Stockholm, Sweden

May 2023
Abstract

Noise pollution is a big health hazard for people living in urban areas, and its effects on humans is a growing field of research. One of the major contributors to urban noise pollution is the noise generated by traffic. Noise simulations can be made in order to build noise maps used for noise management action plans, but in order to test their accuracy real measurements needs to be done, in this case in the form of noise measurements taken adjacent to a road. The aim of this project is to test machine learning based methods in order to develop a robust way of detecting and classifying vehicle noise in sparse traffic conditions. The primary focus is to detect traffic noise events, and the secondary focus is to classify what kind of vehicle is producing the noise.

The data used in this project comes from sensors installed on a testbed at a street in southern Stockholm. The sensors include a microphone that is continuously measuring the local noise environment, a radar that detects each time a vehicle is passing by, and a camera that also detects a vehicle by capturing its license plate. Only sparse traffic noises are considered for this thesis, as such the audio recordings used are those where the radar has only detected one vehicle in a 40 second window. This makes the data gathered weakly labeled.

The resulting detection method is a two-step process: First, the unsupervised learning method k-means is implemented for the generation of strong labels. Second, the supervised learning method random forest or support vector machine uses the strong labels in order to classify audio features. The detection system of sparse traffic noise achieved satisfactory results. However, the unsupervised vehicle classification method produced inadequate results and the clustering could not differentiate different vehicle classes based on the noise data.

Keywords

Sammanfattning

Svensk titel: Detektering och klassificering av bullerhändelser från gles trafik.

Buller är en stor hälsorisk för människor som bor i stadsområden, och dess effekter på människor är ett växande forskningsfält. En av de största bidragen till stadsbuller är oljud som genereras av trafiken. Man kan utföra simuleringar i syfte att skapa bullerkartor som kan användas till planer för att minska dessa ljud. För att testa deras noggrannhet måste verkliga mätningar tas, i detta fall i formen av ljudmätningar tagna intill en väg. Syftet med detta projekt är att testa maskininlärningsmetoder för att utveckla ett robust sätt att detektera och klassificera fordonsljud i glesa trafikförhållanden. Primärt fokus ligger på att detektera bullerhändelser från trafiken, och sekundärt fokus är att försöka klassificera vilken typ av fordon som producerade ljudet.


Nyckelord

Buller, Maskininlärning, Ljudhändelsedetektering, Stödvektormaskin, SVM, Slumpmässiga beslutskogar, RF, K-means klustring, Sfärisk k-means klustring, Trafikljud, Bullerhändelse.
Acknowledgements

We would like to thank our supervisors for making it possible for us to write this thesis: Prof. Maurice Duits, KTH, who has been supervising our thesis from the academic side, and Assoc. Prof. Romain Rumpler, KTH, who is a principle investigator for the project Geometric and provided us with the opportunity to write our thesis with them. We would also like to give special thanks to PhD. student Siddharth Venkataraman, KTH, who has enthusiastically been guiding us throughout this whole project.

A big thank you to you all.

Kevin Golshani & Elias Ekberg
Table of Contents

List of Figures viii
List of Tables x
List of Abbreviations xii

1 Introduction 1
  1.1 Background ............................................. 1
  1.2 Research Questions ................................... 3
  1.3 Process ................................................. 3
  1.4 Outline ................................................ 3
  1.5 Related Work ......................................... 4

2 Theoretical Background on Audio Features 7
  2.1 Digitizing Sound ...................................... 7
  2.2 Spectrograms ........................................... 7
  2.3 Mel Scale ............................................... 11
  2.4 Log-mel Spectrograms .................................. 13
  2.5 Mel-frequency Cepstral Coefficients ................. 13

3 Theoretical Background on Machine Learning 15
  3.1 Unsupervised Learning ................................. 15
    3.1.1 K-means Clustering ................................. 15
    3.1.2 Spherical K-means Clustering ....................... 17
  3.2 Supervised Learning ................................... 18
    3.2.1 Support Vector Machine ............................ 19
    3.2.2 Random Forest ..................................... 20
  3.3 Cross Validation ....................................... 21
  3.4 Evaluation Metrics ..................................... 21
    3.4.1 Accuracy Score ...................................... 21
    3.4.2 F1-score ........................................... 22
List of Figures

2.2.1 FFT plot. ................................................................. 9
2.2.2 Hann window applied to an audio signal. ................. 10
2.2.3 Illustrative example of the spectrogram calculation, highlighting
    the first two frames, their DFT transformation and subsequently,
    the DFT of the framed signal being represented in the spectro-
    gram. Here, the frames are not overlapping for simplicity. Note:
    Figure only meant for visualization and is not entirely correct. . 11
2.2.4 Power spectrogram of audio signal consisting of three guitar notes. 11
2.3.1 Mel filterbanks using 10 Mel-filters. ....................... 12
2.4.1 Log-mel spectrogram of the same audio signal with the three
guitar notes implementing 64 mel filters. The y-axis elements
    of this spectrogram will therefore consist of 64 values (log-mel
    energies) for each frame. These 64 mel filters divide the y-axis
    into 64 bands, often called mel bands. ......................... 13
2.5.1 First 13 MFCCs calculated with 64 Mel-filters of the same three
guitar notes. ............................................................. 14

3.1.1 Initial data points, and cluster centroids assignment. . 16
3.1.2 The nearest centroid classifies each data points. ........... 16
3.1.3 New centroids based on each cluster, boundaries are redrawn
    which reclassifies the data. ...................................... 16
3.1.4 Final state of convergence. Each cluster centroid appear as an X 16
3.1.5 Initial data points, and the unit sphere. ...................... 18
3.1.6 Data points projected onto the unit sphere. Initial centroids cho-
    sen. Marked as X's. .............................................. 18
3.1.7 Centroids shift positions based on the classifications from the
    previous figure. .................................................. 18
3.1.8 Final convergence. A data point that was classified as green in
    k-means is now red. ........................................... 18
3.2.1 Example of how a support vector machine makes a decision bound-
    ary, marked in the figure as a dashed line. The support vectors
    are marked by the parallel orange lines that marks the maximum
    margin. ........................................................... 20
4.2.1 Flow chart of classification process. .............................. 24
5.0.1 Flow chart of the SED process. ................................. 32
5.0.2 An example of a vehicle pass-by recording where log-mel energies are used as features for the ML model. Given a weak label in the neighbourhood of a vehicle pass-by, strong labels are generated by the unsupervised SED system. ........................................ 33
5.0.3 An example of a vehicle pass-by recording where log-mel energies are used as features for the ML models. Strong labels are used for training a supervised classifier. The event as detected by the unsupervised system is used as a reference to compare the predicted event from the supervised system. ...................... 33
5.1.1 Microphone setup at Hornsgatan. Green circle marks the position of the microphone, and rectangles marks the four radar sensors, one for each lane, and where the two cameras records, one for each direction. .................................................. 35
5.4.1 An example of a vehicle pass-by recording where log-mel energies is the base feature used for clustering frames with k-means. Three clusters where used and is denoted by different colors, yellow, blue and magenta. In this case, the magenta event-segment is containing all the frames where a vehicle was considered detected. 37
6.1.1 Comparison of time events based on k-means (top) and spherical k-means (bottom). ............................................... 44
6.1.2 Comparison of time events based on k-means with 2, 3, and 4 clusters respectively from top to bottom. .......................... 45
6.1.3 Comparison of time event before and after outliers are removed from the data set. .................................................. 46
List of Tables

4.2.1 Features extracted using log-mel energies or MFCCs as base.

4.3.1 Evaluation metrics presented for different models on both the training set and test set.

4.3.2 Evaluation metrics presented for different models on both the training set and test set.

4.3.3 Evaluation metrics presented for different models on both the training set and test set.

4.3.4 K-means using different number of clusters for feature set 3.

4.3.5 Random forest using different number of trees for feature set 3.

4.3.6 Support vector machine with different kernels for feature set 3.

4.3.7 Support vector machine with different kernels for feature set 11.

5.7.1 Features extracted using log-mel energies or MFCCs as base.

6.2.1 Evaluation scores for different feature sets and k-means (3 clusters) for the generation of strong labels.

6.2.2 Evaluation scores for different feature sets and spherical k-means (3 clusters) for the generation of strong labels.

6.2.3 Evaluation scores for different feature sets and k-means (3 clusters) for the generation of strong labels.

6.2.4 Evaluation scores for different feature sets and spherical k-means (3 clusters) for the generation of strong labels.

6.2.5 Evaluation scores for RF and SVM on the whole sparse data set (2248 audio recordings) using feature set i and k-means (3 clusters) for the generation of strong labels.

6.3.1 Comparison of clustering methods using feature set 3 and vehicle classifications from radar data. Sorted by cluster size.

6.3.2 Comparison of clustering methods using feature set 4 and vehicle classifications from radar data. Sorted by cluster size.

6.3.3 Comparison of clustering methods using feature set 5 and vehicle classifications from radar data. Sorted by cluster size.

6.3.4 Comparison of clustering methods using feature set 11 and vehicle classifications from radar data. Sorted by cluster size.
List of Abbreviations

**CNN** Convolutional Neural Network.

**CRNN** Convolutional Recurrent Neural Network.

**DFT** Discrete Fourier Transform.

**DNN** Deep Neural Networks.

**ER** Error Rate.

**FFT** Fast Fourier Transforms.

**GMM** Gaussian Mixture Model.

**HMM** Hidden Markov Model.

**MFCC** Mel-Frequency Cepstral Coefficient.

**ML** Machine Learning.

**NMF** Non-negative Matrix Factorization.

**RBF** Radial Basis Function.

**RF** Random Forest.

**SED** Sound Event Detection.

**STFT** Short-Time Fourier Transform.

**SVM** Support Vector Machine.
Chapter 1

Introduction

1.1 Background

The automated monitoring of human activities is more prevalent today than it has ever been, and monitoring systems are being deployed in many areas in our living environment, e.g. urban areas and industrial facilities. Early forms of these automated systems were based on video cameras and that continue to be the most widespread sensor modality today. Automated monitoring systems that only rely on visual data often lacks reliability and robustness in several applications. Cameras are sensitive to shadows, reflections, sudden lightning changes and is negatively affected by bad weather conditions. Also, they perform poorly at low or no light conditions such as during nighttime. Furthermore, video-only systems are unable to detect many important audio events like gunshots and car horns. In contrast, audio sensors have many appealing properties: The audio stream require much less memory storage, bandwidth and computational requirements in comparison to a video stream. Audio sensors provide a spherical field of view, whereas cameras have a limited angular field of view. Also, there is no need for illumination using audio sensors. High-tech cameras are often much more expensive than many low-cost microphones that can be used for monitoring. Thus there is immense motivation to deploy audio sensors for automated monitoring tasks, either stand-alone or in tandem with cameras.

In the artificial intelligence scene, sound event detection (SED) refers to the task of detecting and classifying sound events. The goal is to recognize at what temporal instances different sounds are active within an audio recording. There are multiple challenges that are associated with SED and classification, including how to acquire good quality audio recordings, how distinct different sounds are in a polyphonic environment and how far away the source of the sounds are. Natural environments are polyphonic, i.e. multiple sounds may be present at the same time, which make detection and classification even more
difficult. In contrast, monophonic refers to the presence of only one type of sound. Naturally, polyphonic SED systems are more difficult to build than its monophonic counterpart [2].

Another big hindrance for automated SED is the availability of strongly labeled data sets. Strong labels refers to the annotation of audio recordings where the annotation contains temporal information of the sound events present in a given recording, i.e. the onset and offset times for the events. In contrast, weak labels only informs if a sound event is present in a audio recording, but without any temporal information regarding when the sound is active or not [2]. Manually annotating strong labels is very time consuming, making it expensive to produce large data sets with strong labels. This makes the task of SED even more challenging.

SED is a popular up-and-coming research field that has many potential applications such as: noise monitoring in smart cities [3, 4], surveillance [5] and urban planning [4] to name a few. Relating to SED, environmental sound classification is a topic with significant research interest. A particularly useful application in the context of noise pollution is the determination of the contribution from different sources.

Noise pollution originating from traffic in urban environments negatively impact the quality of life for the population. This type of noise pollution has been found to increase the risk of hypertension and of developing ischaemic heart disease such as myocardial infarction [6, 7]. The annoyance of traffic noise can also disrupt productivity and sleep. Noise pollution has been a growing problem due to urbanization, economic growth and the increased use of motorized transport. In 2002, the European Union enacted the Environmental Noise Directive (END) in order to assess the extent of noise pollution and to act on mitigating it. This is done by monitoring noise levels for producing noise maps, which can be used for developing noise management action plans [7]. To this end, audio sensors have been used in order to monitor traffic noise in urban environments [3, 4].

Bringing everything together, there is clear motivation for audio monitoring of traffic noise and this thesis will explore the use of machine learning models for automated SED and classification of sparse traffic noise.
1.2 Research Questions

The research question is as follows:

*What noise signal characteristics, machine learning models and statistical
methods would be suitable for robust detection and classification of sparse road
traffic noise events?*

This can be broken down into the following goals:

- Analyzing the performance of existing methods for noise detection using
custom data.
- Finding a suitable combination of techniques for pre-processing the data,
and models based on supervised machine learning to detect a traffic
noise event.
- Determining the classifications of traffic noise events that can be made
using unsupervised learning.
- Evaluating the robustness of the chosen techniques and models, and
finding possible explanations for their performances.

1.3 Process

To answer the research questions, this thesis is divided in multiple steps.
Initially, a literature study is performed in order to be familiarized with
existing methods and research. Then, to gain experience and increase the
comprehension of the information learned in the literature study, a practical
case study on the UrbanSound8K [10] data set is performed. With the
knowledge gained, methods are implemented on the data, where each step is
dependent on the results of the previous step.

1.4 Outline

I In the first chapter of this report, a background is given about the subject
at hand and the questions that are intended to be answered throughout
this work. A brief explanation of the relevance of this project and other
work is touched upon.

II The second chapter is about the background of how sound is recorded,
stored, and presented digitally using different audio features. This section
is heavily based on Fourier transforms.

III The third chapter is about the different models of machine learning that
will be used in this report. It includes both unsupervised and supervised
models, as well as different evaluation metrics used to evaluate model performance.

IV The fourth chapter is a case study involving the data set called UrbanSound8K [10]. This section presents a recreation of previous works, and the benefits and disadvantages of different methods.

V The fifth chapter presents the methods used for SED and classification, some parts are based on the findings of the previous chapter, and other parts implement new original methods in order to build complete SED and classification systems.

VI The sixth chapter is the results gained from following the methods of the previous chapter.

VII The seventh chapter includes an analysis of the results from the previous chapter, and also a discussion of how the results can be used in further works.

VIII In the eighth and final chapter the previous three chapters of original work are summarized, and the report is concluded.

1.5 Related Work

A good review paper on different polyphonic SED systems is "A Comprehensive Review of Polyphonic Sound Event Detection", by Chan et al. [11]. It presents many of the different methods used by researchers in the field and describes them in a nice and intelligible way. The review highlights many important works and some of them [12, 13] will be presented here.

Earlier approaches for SED systems were based on Gaussian mixture models (GMMs) and hidden Markov models (HMMs), since they were priorly used for speech recognition and music information retrieval. These methods are not designed to detect multiple sound event classes at once, however, this can be done by a setup involving a binary classifier for each sound event class [2]. Mesaros et al. [12] built a GMM-HMM system for polyphonic SED that was tested on an audio database consisting of 61 event classes. Using Mel-Frequency Cepstral Coefficients (MFCCs), in addition to ∆− and ∆∆− MFCCs as input features, they trained a three-state left-to-right HMM with 16 Gaussians per state for each of the 61 event classes. The 61 models were connected into a network HMM, having equal transition probabilities from one event model to another. The system only achieved a F1-score of 30.1% and a Error rate (ER) of 84.1%.

Another method used for SED is non-negative matrix factorization (NMF). NMF can be used both for supervised SED as well as unsupervised audio source separation [11]. Mesaros et al. [13] used coupled NMF, with two input
matrices, one being the audio frequency spectrum (full spectrum or
mel-spectrum) matrix, and the other one being a frame-level one hot encoding
matrix. The system was tested on a database of real audio recordings and
evaluation was based on event detection in one second blocks. The system
obtained an average F1 score of 57.8%.

Another good paper on the principles and methodologies relating to SED is
written by Ykhlef et al. \[14\]. The authors describe the process of building a
monophonic SED system that can detect events within a given set of
polyphonic recordings. To this end, they propose to train $E$ models, each one
being able to detect if an event $e \in \{1, \ldots, E\}$ is present or not within a given
time frame. Four different classifiers were used and trained on 40 MFCCs:
random forest (RF), support vector machine (SVM), convolutional neural
network (CNN) and AdaBoost (AB). Different hyperparameters where used
for the classifiers, the best accuracy scores obtained were $83.69 \pm 0.07\%$ for
RF, $75.25 \pm 0.26\%$ for SVM, $80.95 \pm 0.06\%$ for CNN and $83.69 \pm 0.07\%$ for
AB. The authors also investigated the influence of adding $\Delta$- and $\Delta\Delta$-
MFCCs in addition to MFCCs as features for the the RF, SVM and AB,
noting that it only had a major impact on the SVMs regarding an increased
performance score.

Neural networks, especially deep neural networks (DNNs) can perform
multi-label classification more easily in contrast to some of the previously
detailed methods where multiple models had to be trained individually to
capture different sound event classes \[2\]. Due to advancements in machine
learning algorithms and computer hardware, deep neural networks can be
trained efficiently \[11\]. DNNs are therefore predominant in the field since they
have such a big advantage in solving multi-label classification problems and
due to them, SED has seen major leaps in state-of-the-art performance \[2\].

DCASE \[15\] hosts annual challenges and workshops relating to SED, acoustic
scene classification and more, inviting researchers from all over the world to
take part in the competition. Each passing year is propelling advancements in
SED and related fields. The baseline system given by DCASE for the SED
task is a mean-teacher model, first proposed by Tarvainen et al. \[16\]. The
current baseline is based on the work of JiaKai \[17\], the winner of 2018 task 4.
Log-mel energies were used as input features, and JiaKai’s system achieved an
F1-score of 34.42% and ER of 1.12% in comparison to the former baseline
system used that year that only achieved an F1-score of 14.06% and an ER of
1.54%. The baseline system is composed of two networks that are both the
same convolutional recurrent neural network (CRNN), and basically image
recognition is performed where log-mel spectrograms are used as input
features.

Systems built by participating teams of the DCASE challenge in recent years
has performed better and better and the work done there can be considered as
the current state-of-the-art for SED.
In "A Dataset and Taxonomy for Urban Sound Research" [10], Salamon et al. compared five different machine learning (ML) classifiers, SVM (RBF kernel), random forest (500 trees), decision tree (J48), k-nearest neighbour (k-NN with k=5) and majority vote classifier (ZeroR). The authors used 25 MFCCs to calculate summary statistics over time based on the per-frame values for the coefficients. Calculating the minimum, maximum, median, mean, variance, skewness, kurtosis and the mean and variance of the first and second derivatives based on the MFCCs, the resulting feature vectors contained 225 elements. SVM and RF performed the best and achieved similar accuracy scores in the range [61 – 70]%, while k-NN achieved [50 – 59]%, J48 [48 – 40]% and ZeroR around 10%.
Chapter 2

Theoretical Background on Audio Features

2.1 Digitizing Sound

Within the medium of air, a sound wave is an oscillation of pressure that is created by the vibration of the auditory source. These pressure waves (mechanical waves) vibrate our ear drums and then gets translated into nerve impulses (electrical signals) that the brain interprets as sound. A microphone similarly measures the different pressures of the air in order to record the sound. As these recordings are usually meant to be played back to humans, they measure the pressure at 44.1 kHz, i.e. 44 100 samples per second. This is the Nyquist rate for sampling a signal of 22 050 Hz, the upper limit of human hearing [18], and is required to not get any aliasing for these frequencies. After being sampled, this signal can then be transformed using the short-time Fourier transform (STFT) that will be described in the following section. The STFT will be the basis for both spectrograms and MFCCs which will be introduced in section 2.2 and 2.5 respectively. Where the original signal is in the time domain, the signal will be transformed to a time-frequency domain, and it can be represented in different kinds of 3D plots.

2.2 Spectrograms

The spectrogram is a commonly used audio feature which is a time-frequency representation of an input signal. The calculation of a spectrogram is centered around performing Fourier transforms. The discrete Fourier transform (DFT) is the most common transformation used for audio signals. The DFT represents the input signal with a superposition of sinusoidal basis functions where each basis function is characterized by its magnitude and phase [19]. Given an input signal $x[n]$, the DFT is defined by [20]
\[ X[k] = \sum_{n=0}^{N-1} x[n]e^{-i2\pi nk/N}, \quad 0 \leq k < N. \] (2.1)

In other words, the DFT will transform a sequence of \( N \) complex numbers \( x[n] \) into another sequence of \( N \) complex numbers \( X[k] \). The input signal \( x[n] \) will be transformed from the time domain and represented as a new signal \( X[k] \) defined on the frequency domain.

Calculating the DFT in equation (2.1) is computationally expensive and requires \( N^2 \) operations. To solve this, there exists a collection of fast algorithms used to compute the DFT, which are called fast Fourier transforms (FFT). The FFT algorithms reduces the number of operations to only \( N\log_2 N \), and this is why FFT is used in practice. Hereafter, note that the DFT is used for all mathematical related theory, but in practice and in presented plots, it is the FFT that is used.

The output of the DFT will give us information about which frequency components are most present in the original signal. This will now be illustrated with an example. Consider an input signal of multiple sine waves defined by

\[ x(t) = A_1\sin(2\pi f_1 t) + A_2\sin(2\pi f_2 t) + A_3\sin(2\pi f_3 t), \] (2.2)

where the amplitudes are given by \([A_1, A_2, A_3] = [1, 3, 6]\) and the frequencies are given by \([f_1, f_2, f_3] = [10, 30, 20]\) (Hz). When calculating the FFT, each bin number \( k \) is mapped to a frequency \( f_k \) in Hz according to \( f_k = k \frac{f_s}{N_{\text{FFT}}} \), where \( f_s \) is the sampling rate and \( N_{\text{FFT}} \) is the length of the FFT. The magnitude plot of the FFT of the input signal \( x(t) \) is presented in Figure 2.2.1. It can be seen that the biggest magnitude peak is 6 and it can be found at a corresponding frequency of 20 Hz. The medium peak is 3 and can be found at 30 Hz and the smallest peak is 1 and can be found at 10 Hz.
The kind of frequency component decomposition that is observed in Figure 2.2.1 is very useful, however, the drawback of the DFT is that the frequency components are averaged across the whole signal duration. Thus the output loses all temporal information contained in the original time domain input signal: we know which frequency components are present, but we do not know when they are more or less present. To get around this problem and reintroduce temporal information in the frequency domain, a short-time Fourier transform is performed. The STFT can be interpreted as a sliding window transform where a DFT is applied on a windowed frame of length $N_{\text{FFT}}$ of the input signal $x[n]$. Consider the signal segments $x_t[n]$ of $x[n]$ given by

$$
x_t[n] = x[n + tH]w[n], \quad 0 \leq n \leq N_{\text{FFT}} - 1,
$$

where $w[n]$ is a window function, $N_{\text{FFT}}$ is the window length, $n$ is the local time index within the window, $t$ is the frame index and $H$ is the hop length. The hop length determines the spacing between each consecutive frame, and if the hop length is smaller than the window length, there will be overlap between neighbouring frames. There are multiple choices of window functions $w[n]$ that can be implemented, some common choices are rectangular, Hanning (Hann), Hamming and Blackman windows. In addition, the window function will attenuate some of the effects of the DFT approximation and it will also enforce continuity and periodicity at the edges of the frames. The Hann window will be used in this thesis and is defined by

$$
w[n] = \begin{cases} 
\frac{1}{2} \left[ 1 - \cos \left( \frac{2\pi n}{N_{\text{FFT}}} \right) \right] & 0 \leq n < N_{\text{FFT}}, \\
0 & \text{otherwise}.
\end{cases}
$$

Figure 2.2.1: FFT plot.
An example of a Hann window applied to a waveform can be seen in Figure 2.2.2. Notice that the signal gets attenuated towards the edges.

Next, applying a DFT for each signal frame $t$, the STFT can be computed as

$$X[t, k] = \sum_{n=0}^{N_{FFT}-1} x[n + tH]w[n]e^{-j2\pi nk/N_{FFT}}.$$  \hspace{1cm} (2.5)

Now, calculating the power (square magnitude) of $X[t, k]$ results in a power spectrogram $|X[t, k]|^2$. Also, the magnitude has been converted to a dB-scale and then the spectrogram is plotted on a logarithmic scale. This is sharp: A three dimensional plot (2D matrix) is obtained where it can be seen how the frequency components evolve over time. Each point in the spectrogram is assigned a color indicating the amplitude of a frequency at any given time. A higher color intensity means that a particular frequency is more present, and lower intensity means that a frequency is less present.

The process of obtaining a spectrogram by taking the DFT for each frame is illustrated in Figure 2.2.3. Figure 2.2.4 shows the power spectrogram for an audio signal that consists of three guitar notes.
2.3 Mel Scale

Human ears do not interpret sound in a linear fashion, rather they work on a logarithmic scale. Therefore, it could be beneficial to use a non-linear frequency scale, which would better approximate the behaviour of the human auditory system. One such perceptually modeled scale is the mel frequency
scale, where one mel is defined as one thousandth of the pitch of a 1 kHz tone. The mel scale can be approximated by

$$B(f) = \frac{1125 \ln \left(1 + \frac{f}{700}\right)}{},$$

where $f$ is the frequency in hertz and $B$ is the frequency in mels. To get a spectrogram that uses the mel scale instead of the standard frequency scale, a filterbank $H_m[k]$ is defined with $M$ triangular filters ($m = 1, 2, \cdots, M$) that is given by

$$H_m[k] = \begin{cases} 
\frac{k-f[m-1]}{f[m]-f[m-1]} & f[m-1] \leq k \leq f[m], \\
\frac{f[m+1]-k}{f[m+1]-f[m]} & f[m] \leq k \leq f[m+1], \\
0 & \text{otherwise.}
\end{cases}$$

The boundary points $f[m]$ in equation (2.7) are uniformly spaced on the mel scale and given by

$$f[m] = \left(\frac{N}{F_s}\right) B^{-1} \left( B(f_l) + m \frac{B(f_h) - B(f_l)}{M+1} \right),$$

where $f_l$ and $f_h$ are the lowest and highest frequencies (Hz) of the filterbank, $F_s$ is the sampling frequency (Hz), $M$ is the number of filters, $N$ is the size of the FFT and $B^{-1}$ is the inverse mel scale transformation given by

$$B^{-1}(b) = 700 \left(\exp\left\{\frac{b}{1125}\right\} - 1\right).$$

An illustration of 10 triangular mel filters can be seen in Figure 2.3.1.

![Mel filterbanks using 10 Mel-filters.](image)

Figure 2.3.1: Mel filterbanks using 10 Mel-filters.

\(^1\)Note: The constant 1125 in equation (2.6) seem to differ depending on different literary sources, and 1125 was presented here for consistency regarding the sources used in this section. However, in practice a constant of 1127 was used in this thesis.
2.4 Log-mel Spectrograms

Now that the mel filters have been introduced, they can be applied to a power spectrogram to make it a mel spectrogram. This can be done by calculating

\[ Z[t, m] = \sum_j H_m[j]|X[t, j]|^2, \]  

(2.10)

where, \( H_m[j] \) represents the frequency response of the \( m \)th filter in the filterbank defined in equation (2.7). As before, the amplitude is converted to a dB-scale and then the mel spectrogram is plotted on a logarithmic scale. This results in a log-mel spectrogram which can be seen in Figure 2.4.1.

Figure 2.4.1: Log-mel spectrogram of the same audio signal with the three guitar notes implementing 64 mel filters. The y-axis elements of this spectrogram will therefore consist of 64 values (log-mel energies) for each frame. These 64 mel filters divide the y-axis into 64 bands, often called mel bands.

2.5 Mel-frequency Cepstral Coefficients

Another very commonly used audio feature are the Mel-frequency cepstral coefficients (MFCCs) [19]. They were originally developed for speech recognition by Davis and Mermelstein in the 1980s [24]. Davis and Mermelstein were motivated to formulate the mel frequency cepstrum (MFC) as their design is modeled after the human perception of sound, which is based on a frequency analysis that occurs in the inner ear [21]. Since then, MFCCs have been used by researchers for music information retrieval tasks such as genre recognition and sound scene recognition [25].

Given the DFT of the input signal \( x[n] \) from equation (2.1) and the mel filterbanks from equation (2.7), the log-energy at the output of each triangular filter is computed as [20]
Next, a discrete cosine transform (DCT-II) defined as

\[
C[k] = \sum_{n=0}^{N-1} x[n] \cos \left( \frac{\pi}{N} (n + \frac{1}{2}) k \right), \quad 0 \leq k < N,
\]

(2.12)
is applied to equation (2.11), which results in the mel frequency cepstrum \(mfcc[n] = \sum_{m=0}^{M-1} S[m] \cos \left( \frac{\pi}{M} (m + \frac{1}{2}) n \right)\), \(0 \leq n < M\).

(2.13)
The input signal \(x[n]\) will be windowed by a window function in the same manner as presented for the spectrogram calculation. Then equation (2.13) will be applied to \(S_t[m]\) (windowed version of equation (2.11)) for each frame \(t\). This will result in a three dimensional MFCC plot (2D matrix). Similarly as for the spectrogram, the MFCC plot will show how each MFC coefficient evolve over time, where the color intensity of each MFC indicate its value at any given time. In Figure 2.5.1 we can see the MFCC plot for the same audio signal with the three guitar notes.

![Figure 2.5.1: First 13 MFCCs calculated with 64 Mel-filters of the same three guitar notes.](image)

One big advantage of using MFCCs is that it can reduce the size of the audio feature matrix (spectrogram matrix or MFCC matrix) dramatically. Usually the number of MFC coefficients that are computed for each frame are less than 25. This can be compared to log-mel spectrograms that often uses 128 mel bands which results in 128 elements extracted for each frame.
Chapter 3

Theoretical Background on Machine Learning

3.1 Unsupervised Learning

Unsupervised models of machine learning are those algorithms that fit a model to unlabeled data. This is done through recognizing patterns in the data, in order to group data in different classes. This allows the algorithm to later assign new data to these groups in order to classify them. If the new data is also used to train the model, then it is called a dynamic model. The opposite is a static model, which does not change after initial training.

3.1.1 K-means Clustering

K-means clustering is a unsupervised model that is based on clustering together data points into \( k \) number of clusters. The aim of this algorithm is to find the \( k \) centroids in the middle of these clusters, that minimizes the square of the euclidean distance for each data point belonging to that cluster. As such, the resulting clusters should have a minimum variance for the data points of the respective cluster.

The standard algorithm for k-means when clustering \( n \) data points \( x \) is the following:

- Randomly choose \( k \) data points, they are now the cluster centroids \( m_c \) belonging to cluster \( c \) where \( c = \{1, ..., k\} \).

- Assign all data points \( x \) to the sets \( S_c \) such that:

\[
x_i \in \{ S_c : \|x_i - m_c\|^2 \leq \|x_i - m_p\|^2 \ \forall p, 1 \leq p \leq k \}.
\]  \hspace{1cm} (3.1)

- Recalculate the cluster centroids \( m_c \) as the mean of all data points \( x_i \) assigned to that set \( \forall x_i \in S_c \):
\[ m_c = \frac{1}{|S_c|} \sum_{x_i \in S_c} x_i. \]  

- Reassign all data points to these new centroids \( m_c \), and repeat the previous two steps until convergence is made.

With \( k \) clusters and \( N \) data points there are \( k^N \) combinations of assigning these data points to the clusters. This algorithm takes a naive approach and will only find a local optimum \cite{26}, but since the problem is NP-hard \cite{27} it is not guaranteed to find a global optimum with regards to minimum variance. For better results, this method can be run multiple times with different starting points. Figures 3.1.1, 3.1.2, 3.1.3, 3.1.4 show a simple clustering process of eight data points using k-means with three clusters.

Figure 3.1.1: Initial data points, and cluster centroids assignment.

Figure 3.1.2: The nearest centroid classifies each data points.

Figure 3.1.3: New centroids based on each cluster, boundaries are redrawn which reclassifies the data.

Figure 3.1.4: Final state of convergence. Each cluster centroid appear as an X.
3.1.2 Spherical K-means Clustering

Spherical k-means clustering is a similar clustering model to k-means, but instead of aiming to minimize the euclidean dissimilarity within a cluster, the value-function aims to minimize the cosine-dissimilarity [28], in other words, maximize the cosine-similarity. This makes it so that the data points in each cluster are forced to be more similar in their direction and less dependent of their magnitude. The cosine similarity is as follows:

\[ S_C(A, B) := \cos(\theta) = \frac{A \times B}{||A|| \cdot ||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}, \]  

\[ (3.3) \]

where \( S_C(A, B) = 1 \) means \( A \) and \( B \) have the same direction, \( S_C(A, B) = 0 \) means they are orthogonal, and \( S_C(A, B) = -1 \) means they are opposite direction.

With spherical k-means the data is first normalized to the unit-sphere, and then the centroids are forced to appear on the sphere as well. In Figures 3.1.5-3.1.8 convergence of the spherical k-means algorithm can be seen, with the end result of having one of the previously green labeled data points being classified as red instead.
3.2 Supervised Learning

Supervised models of machine learning are algorithms that fit a model to labeled data. This gives the model a chance of predicting new and unknown data to a specific category, instead of just an unknown cluster as before. If the new and unknown data is also used to train the model, then the model becomes a semi-supervised model. Labeled data is often referred to as strongly or weakly labeled, where strongly labeled data is often classified manually with specific distinct labels, and weakly labeled data is more generally categorized, with less precision.
3.2.1 Support Vector Machine

Support vector machine (SVM) is a supervised learning model which aims to separate $p$-dimensional data points into their classes using a $(p-1)$-dimensional hyperplane. This hyperplane should not only separate the data, but doing so with a maximal margin, meaning that the distances between the plane and the nearest data points are maximized.

For a given data-set $S = \{(x_1, y_1), ..., (x_n, y_n)\}$ of $n$ data points, where $x_i$ is a $p$-dimensional vector, and $y_i \in \{\pm 1\}$ is the corresponding label, a $(p-1)$-dimensional hyperplane defined by its normal vector $w \in \mathbb{R}^p$ will linearly separate the data into their labels if

$$y_i(w^T x_i + b) > 0 \quad \forall i = \{1, ..., n\} \quad (3.4)$$

hold true, where $b$ is a constant. The $w$ and $b$ can then be used to classify an unknown data point $x_{i+1}$ by the sign-function, $\text{sgn}(w^T x_{i+1} + b)$. In order to maximize the margin two parallel lines are defined

$$w^T x + b = +1, \quad (3.5)$$

and

$$w^T x + b = -1, \quad (3.6)$$

where all data points of label $+1$ are at or above the first line, and all the data points of label $-1$ are at or below the second line. The distance between these new lines will be $2/||w||$, so in order to maximize their distance, $||w||$ has to be minimized

$$\min_{w,b} ||w||^2 \quad \text{s.t.} \quad y_i(w^T x_i + b) \geq 1 \quad \forall i = \{1, ..., n\}. \quad (3.7)$$

As a result only the closest data points defines the hyperplane and are therefore called the support vectors for this plane, which gives the method its name. As seen in figure 3.2.1 only the vectors on the orange lines affect the decision boundary, as opposed to k-means where all data points defines the classification-boundary.
Figure 3.2.1: Example of how a support vector machine makes a decision boundary, marked in the figure as a dashed line. The support vectors are marked by the parallel orange lines that marks the maximum margin.

When data sets are not linearly separable, a kernel can be used to transform the data into a higher dimension where separation might be easier [27]. This is referred to as the kernel trick, and some examples of common kernels used are linear kernel, polynomial kernel (Poly), radial basis function kernel (RBF) and Sigmoid kernel. A loss-function can also be applied to allow for some missclassifications, a commonly used is the hinge-loss function

\[ l_{hinge}((w, b), (x_i, y_i)) = \max\{0, 1 - y_i(w^T x_i + b)\} \]

where a correct classification yields \( l_{hinge} = 0 \), and a miss-classification yields a loss proportional to the distance to the support vector plane. Then a new optimization problem occurs

\[
\begin{align*}
\text{minimize} & \quad \alpha \|w\|^2 + \frac{1}{n} \sum_{i=1}^{n} l_{hinge}((w, b), (x_i, y_i)) \\
\text{s.t.} & \quad y_i(w^T x_i + b) \geq 1 - l_{hinge}((w, b), (x_i, y_i)) \quad \forall i = \{1, ..., n\},
\end{align*}
\]

where \( \alpha > 0 \) is a tuning parameter introduced to control the trade-off between maximum marginal and the allowed classification errors.

### 3.2.2 Random Forest

The random forest (RF) algorithm is another supervised learning model. Random forest consists of a number of decision trees, these trees are either for classification or regression. In this thesis, only a forest of classification trees will be used.
A decision tree has a flowchart-like structure, where each branch is dependent on the attributes of the data point [27]. The decision trees are created by bootstrap aggregating (bagging) the training data for each tree, to induce randomness. The bagging is done by taking \( N \) random samples with replacement from a training data containing \( N \) data points. This results in an average of 63.2\% different data points in each bootstrapping sample if \( N \) is large enough. By dividing these training data into smaller portions, the decision trees can be built using the Gini impurity model given by

\[
G(p) = \sum_{i=1}^{J} p_i (1 - p_i) = \sum_{i=1}^{J} (p_i - p_i^2) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} p_i^2 = 1 - \sum_{i=1}^{J} p_i^2. \tag{3.10}
\]

The Gini impurity is defined as the probability of incorrectly classifying a data point. This is calculated by taking the probability \( p_i \) of the true classification for a data point and multiplying it with the probability of a miss-classification \((1 - p_i)\). The sum of all possible classifications are then the Gini impurity.

When constructing the random forest, each tree is assigned a number of random features that the data set can be split by. Normally the number of features is set to the square root of the total number of attributes. Each data split should be performed in such way that minimizes the impurity \( G(p) \). This is done until all data is split down into single branches. Once enough trees has been constructed, a classification can made by letting the forest decide through the decision trees by a majority vote.

### 3.3 Cross Validation

In order to avoid over fitting or selection bias, cross validation is often implemented during the testing of various modeling problems. The first step is to separate the data set into a training set and a test set, such that the test set is completely unseen by the model to give a more accurate result of the models future predictions. There are various ways of performing this but in this thesis the \( k \)-fold cross validation will be used. With \( k \)-fold cross validation, the data available is split into \( k \) folds, where the model is trained on \( k - 1 \) folds and then tested on the one remaining hold-out fold. This is repeated \( k \) times such that each fold is used as a test set once, and then the performance is averaged based on the results of all test runs [27 Sect. 11.2.4].

### 3.4 Evaluation Metrics

#### 3.4.1 Accuracy Score

What will be referred to as the accuracy score in this thesis is the ordinary and familiar classification measure which represents the percentage of correct
predictions made by a ML model. The accuracy score is both intuitive and simple to calculate, however, in some applications the accuracy score is not well suited to evaluate model performance, especially in cases where large class imbalances are present. More formally, the accuracy score is given by

\[
\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}. \tag{3.11}
\]

Consider a data set that consists of 100 cars, 90 cars are blue and 10 cars are yellow. In a binary classification problem (yellow or blue), a ML model that would predict cars as only blue would still obtain an accuracy score of 90%, even though it is clearly a wrong and useless model. This is why other evaluation metrics might be needed, and to this end, the F1-score is introduced.

### 3.4.2 F1-score

When evaluating models of binary classification it is important to have a value function that considers both the precision and the recall of the model. Precision is the measurement of accuracy of the result, i.e. the fraction of true positive classifications over the total number of positive classifications. It is given by

\[
\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}} = \frac{TP}{TP + FP}. \tag{3.12}
\]

The recall is the measurement of accuracy of the sensitivity i.e. the fraction of true positive classification over the total number of positive elements. Recall is given by

\[
\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}} = \frac{TP}{TP + FN}. \tag{3.13}
\]

The F1-score is defined as the harmonic mean of precision and recall

\[
F_1 = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}. \tag{3.14}
\]

If the definitions of recall and precision is used, the F1-score is given by

\[
F_1 = \frac{2 \frac{TP}{TP + FP}}{\frac{TP}{TP} + \frac{TP + FP}{TP + FP}} = \frac{2 TP}{2TP + FN + FP}. \tag{3.15}
\]

An F1-score of 1 means perfect accuracy with no false positives and no false negatives, while a score of 0 means there was no true positives found, thus both precision and recall are zero.
Chapter 4

UrbanSound8K

Before getting to the main part of this thesis, a slight detour will first be taken in order for us to get a good understanding of audio based machine learning. The work done here will serve as a sort of hands-on textbook example which will give us some practical experience that we can apply on the main part of the thesis.

4.1 Background

The UrbanSound8K data set \[10\] is a strongly labeled data set containing 8732 audio clips of up to 4 second excerpts of sounds most commonly heard in an urban setting. The audio is focused around ten different classes: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren, and street music. These classes were chosen based on their frequency in noise complaints in urban settings. The data set is available online for free for research purposes, and is split into ten similarly sized folds for cross validation.

4.2 Methods

The data from the UrbanSound8K was used for both supervised and unsupervised models, and some of the methods and feature extractions implemented in this section was inspired by the work done by the authors of UrbanSound8K \[10\], in order to partly recreate some of the results they obtained. The classification process is illustrated in Figure 4.2.1.
For all the methods below, the duration of one frame was set to be 23.2 ms. This is the result of using a sampling rate of 22050 Hz, an FFT frame size of $N_{\text{FFT}} = 1024$ and a hop length of 512. This means that there are 50% overlap between the frames. 64 mel bands were implemented for the calculations and only the first 25 MFCCs were kept.

### 4.2.1 Feature Extraction 1: Image Flattening

A common strategy when performing machine learning on images (arrays) is to flatten the input image before giving it to the model. Array flattening is a method to convert arrays of multiple dimensions into a one dimensional array (vector). Since images are essentially 2D arrays where each element is one pixel, this method can be applied to both log-mel spectrograms as well as MFCC magnitude plots. This method is rather straightforward, however, the downside is that some of the audio samples need to be padded or cut in order for the input dimension to match (for audio samples with different durations). The padding/cutting can affect the ML performance in a negative way. The flattened arrays for log-mel energies will be referred to as "Feature set 1" and the flattened arrays for MFCCs will be referred to as "Feature set 2".

### 4.2.2 Feature Extraction 2: A Statistical Approach

Another method that was tested was inspired by Salamon et al. [10]: The base audio feature are first extracted and then various statistics can be calculated from these features, and concatenated to use as new input features. Whether log-mel spectrograms or MFCCs, statistics such as mean, standard deviation, maximum, minimum, variance, median and kurtosis can be calculated for each mel band or MFCC over time. Delta and delta-delta coefficients are also calculated for the MFCCs (referred to as $\Delta$MFCC and $\Delta\Delta$MFCC). These are roughly equivalent to calculating the first and second derivatives. The formula for calculating delta coefficients is given by
\[ d_t = \frac{1}{2}(c_{t+1} - c_{t-1}), \quad (4.1) \]

where \( d_t \) is the deltas at time \( t \) and \( c_t \) is the coefficients at time \( t \). Delta-delta coefficients are calculated using the same formula, but they are calculated using the deltas, rather than the static coefficients as in equation (4.1). This way, some temporal information can be introduced since information about the rate of change for each MFCC is obtained. Different combinations of statistical features were tried to see what would have the best performance.

For example, if 25 MFCCs are used as the base feature, and the mean and standard deviation is calculated, the resulting input vector will have 50 features (25+25). If instead, log-mel energies is used with 64 mel bands, and the mean, standard deviation, maximum and minimum is calculated, the resulting input vector will have 256 features (64+64+64+64). The advantage of this approach is that it no longer is necessary to pad or cut the audio signals to make them uniform in size, since only statistics calculated from the base feature are used rather than the base feature itself.

Combinations of features extracted are presented in Table 4.2.1 as different statistical feature sets.

<table>
<thead>
<tr>
<th>Feature set No.</th>
<th>Audio features</th>
<th>Statistical functions applied on the audio features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature set 3</td>
<td>MFCC</td>
<td>arithmetic mean, standard deviation</td>
</tr>
<tr>
<td>Feature set 4</td>
<td>MFCC</td>
<td>arithmetic mean, standard deviation, maximum, minimum</td>
</tr>
<tr>
<td>Feature set 5</td>
<td>MFCC</td>
<td>arithmetic mean, standard deviation, maximum, minimum, median, kurtosis, skewness, variance</td>
</tr>
<tr>
<td>Feature set 6</td>
<td>MFCC + ∆ MFCC</td>
<td>arithmetic mean, standard deviation</td>
</tr>
<tr>
<td>Feature set 7</td>
<td>MFCC + ∆ MFCC</td>
<td>arithmetic mean, standard deviation, maximum, minimum</td>
</tr>
<tr>
<td>Feature set 8</td>
<td>MFCC + ∆ MFCC</td>
<td>arithmetic mean, standard deviation, maximum, minimum, median, kurtosis, skewness, variance</td>
</tr>
<tr>
<td>Feature set 9</td>
<td>MFCC + ∆ MFCC + ∆∆ MFCC</td>
<td>arithmetic mean, standard deviation</td>
</tr>
<tr>
<td>Feature set 10</td>
<td>MFCC + ∆ MFCC + ∆∆ MFCC</td>
<td>arithmetic mean, standard deviation, maximum, minimum</td>
</tr>
<tr>
<td>Feature set 11</td>
<td>MFCC + ∆ MFCC + ∆∆ MFCC</td>
<td>arithmetic mean, standard deviation, maximum, minimum, median, kurtosis, skewness, variance</td>
</tr>
<tr>
<td>Feature set 12</td>
<td>log-mel energies</td>
<td>arithmetic mean, standard deviation</td>
</tr>
<tr>
<td>Feature set 13</td>
<td>log-mel energies</td>
<td>arithmetic mean, standard deviation, maximum, minimum</td>
</tr>
</tbody>
</table>

Table 4.2.1: Features extracted using log-mel energies or MFCCs as base.
4.2.3 Training and Testing ML Classifier

Feature set 1-13 is used to train the following ML models: k-means, spherical k-means, random forest and support vector machine with different model parameters in order to find the best features to use and which model parameters that achieves the highest scores. 10-fold cross validation was implemented according to the predetermined folds in UrbanSound8K. Feature standardization was applied for the training and testing arrays when k-means were used, removing the mean of the features, and scaling them to unit variance. This standardization process only improved the performance for k-means and was therefore omitted for the other models.

4.2.4 Evaluation

The true labels will be compared with the predicted labels and evaluation of the ML models is done based on the accuracy score and the F1-score.

4.3 Results

<table>
<thead>
<tr>
<th>Feature set No.</th>
<th>Accuracy train set</th>
<th>Accuracy test set</th>
<th>F1 train set</th>
<th>F1 test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature set 1</td>
<td>26.92%</td>
<td>23.70%</td>
<td>24.40%</td>
<td>22.16%</td>
</tr>
<tr>
<td>Feature set 2</td>
<td>32.59%</td>
<td>30.06%</td>
<td>27.89%</td>
<td>24.55%</td>
</tr>
<tr>
<td>Feature set 3</td>
<td>35.99%</td>
<td>35.35%</td>
<td>32.46%</td>
<td>31.34%</td>
</tr>
<tr>
<td>Feature set 4</td>
<td>36.37%</td>
<td>35.68%</td>
<td>32.77%</td>
<td>31.28%</td>
</tr>
<tr>
<td>Feature set 5</td>
<td>36.08%</td>
<td>35.93%</td>
<td>32.03%</td>
<td>30.96%</td>
</tr>
<tr>
<td>Feature set 6</td>
<td>31.39%</td>
<td>28.14%</td>
<td>23.26%</td>
<td>20.69%</td>
</tr>
<tr>
<td>Feature set 7</td>
<td>35.75%</td>
<td>34.56%</td>
<td>30.02%</td>
<td>28.90%</td>
</tr>
<tr>
<td>Feature set 8</td>
<td>35.43%</td>
<td>33.42%</td>
<td>29.12%</td>
<td>27.18%</td>
</tr>
<tr>
<td>Feature set 9</td>
<td>28.61%</td>
<td>27.08%</td>
<td>21.39%</td>
<td>19.94%</td>
</tr>
<tr>
<td>Feature set 10</td>
<td>32.48%</td>
<td>31.02%</td>
<td>28.53%</td>
<td>27.14%</td>
</tr>
<tr>
<td>Feature set 11</td>
<td>30.32%</td>
<td>29.03%</td>
<td>24.42%</td>
<td>22.87%</td>
</tr>
<tr>
<td>Feature set 12</td>
<td>29.32%</td>
<td>23.19%</td>
<td>26.81%</td>
<td>22.60%</td>
</tr>
<tr>
<td>Feature set 13</td>
<td>28.84%</td>
<td>23.37%</td>
<td>25.38%</td>
<td>20.93%</td>
</tr>
</tbody>
</table>

Table 4.3.1: Evaluation metrics presented for different models on both the training set and test set.
<table>
<thead>
<tr>
<th>Feature set No.</th>
<th>Accuracy train set</th>
<th>Accuracy test set</th>
<th>F1 train set</th>
<th>F1 test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature set 1</td>
<td>29.30%</td>
<td>24.08%</td>
<td>23.68%</td>
<td>18.09%</td>
</tr>
<tr>
<td>Feature set 2</td>
<td>34.43%</td>
<td>27.59%</td>
<td>30.79%</td>
<td>23.31%</td>
</tr>
<tr>
<td>Feature set 3</td>
<td>40.38%</td>
<td>37.50%</td>
<td>39.76%</td>
<td>35.67%</td>
</tr>
<tr>
<td>Feature set 4</td>
<td>40.17%</td>
<td>36.32%</td>
<td>39.76%</td>
<td>35.43%</td>
</tr>
<tr>
<td>Feature set 5</td>
<td>41.25%</td>
<td>37.06%</td>
<td>38.62%</td>
<td>33.99%</td>
</tr>
<tr>
<td>Feature set 6</td>
<td>39.91%</td>
<td>35.53%</td>
<td>40.01%</td>
<td>33.86%</td>
</tr>
<tr>
<td>Feature set 7</td>
<td>37.23%</td>
<td>31.42%</td>
<td>35.22%</td>
<td>28.52%</td>
</tr>
<tr>
<td>Feature set 8</td>
<td>39.03%</td>
<td>31.50%</td>
<td>34.97%</td>
<td>28.42%</td>
</tr>
<tr>
<td>Feature set 9</td>
<td>39.05%</td>
<td>34.99%</td>
<td>39.26%</td>
<td>33.24%</td>
</tr>
<tr>
<td>Feature set 10</td>
<td>37.93%</td>
<td>33.16%</td>
<td>36.21%</td>
<td>30.26%</td>
</tr>
<tr>
<td>Feature set 11</td>
<td>37.92%</td>
<td>34.68%</td>
<td>33.57%</td>
<td>29.80%</td>
</tr>
<tr>
<td>Feature set 12</td>
<td>30.46%</td>
<td>24.79%</td>
<td>23.77%</td>
<td>18.60%</td>
</tr>
<tr>
<td>Feature set 13</td>
<td>32.38%</td>
<td>29.68%</td>
<td>25.76%</td>
<td>22.80%</td>
</tr>
</tbody>
</table>

Table 4.3.2: Evaluation metrics presented for different models on both the training set and test set.

<table>
<thead>
<tr>
<th>Feature set No.</th>
<th>Accuracy train set</th>
<th>Accuracy test set</th>
<th>F1 train set</th>
<th>F1 test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature set 1</td>
<td>100%</td>
<td>51.21%</td>
<td>100%</td>
<td>52.85%</td>
</tr>
<tr>
<td>Feature set 2</td>
<td>100%</td>
<td>56.12%</td>
<td>100%</td>
<td>57.60%</td>
</tr>
<tr>
<td>Feature set 3</td>
<td>100%</td>
<td>66.03%</td>
<td>100%</td>
<td>66.63%</td>
</tr>
<tr>
<td>Feature set 4</td>
<td>100%</td>
<td>66.46%</td>
<td>100%</td>
<td>67.76%</td>
</tr>
<tr>
<td>Feature set 5</td>
<td>100%</td>
<td>67.32%</td>
<td>100%</td>
<td>68.46%</td>
</tr>
<tr>
<td>Feature set 6</td>
<td>100%</td>
<td>68.60%</td>
<td>100%</td>
<td>68.88%</td>
</tr>
<tr>
<td>Feature set 7</td>
<td>100%</td>
<td>68.94%</td>
<td>100%</td>
<td>69.85%</td>
</tr>
<tr>
<td>Feature set 8</td>
<td>100%</td>
<td>69.14%</td>
<td>100%</td>
<td>69.75%</td>
</tr>
<tr>
<td>Feature set 9</td>
<td>100%</td>
<td>68.21%</td>
<td>100%</td>
<td>68.58%</td>
</tr>
<tr>
<td>Feature set 10</td>
<td>100%</td>
<td>68.93%</td>
<td>100%</td>
<td>69.84%</td>
</tr>
<tr>
<td>Feature set 11</td>
<td>100%</td>
<td>69.46%</td>
<td>100%</td>
<td>70.18%</td>
</tr>
<tr>
<td>Feature set 12</td>
<td>100%</td>
<td>60.01%</td>
<td>100%</td>
<td>60.48%</td>
</tr>
<tr>
<td>Feature set 13</td>
<td>100%</td>
<td>60.87%</td>
<td>100%</td>
<td>61.93%</td>
</tr>
</tbody>
</table>

Table 4.3.3: Evaluation metrics presented for different models on both the training set and test set.
### Unsupervised (k-means)

<table>
<thead>
<tr>
<th># clusters</th>
<th>Accuracy train set</th>
<th>Accuracy test set</th>
<th>F1 train set</th>
<th>F1 test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>35.99%</td>
<td>35.35%</td>
<td>32.46%</td>
<td>31.34%</td>
</tr>
<tr>
<td>20</td>
<td>45.90%</td>
<td>40.39%</td>
<td>46.44%</td>
<td>41.40%</td>
</tr>
<tr>
<td>64</td>
<td>55.53%</td>
<td>48.27%</td>
<td>57.60%</td>
<td>50.49%</td>
</tr>
<tr>
<td>128</td>
<td>65.28%</td>
<td>50.37%</td>
<td>67.15%</td>
<td>52.91%</td>
</tr>
<tr>
<td>256</td>
<td>74.43%</td>
<td>49.67%</td>
<td>75.41%</td>
<td>52.21%</td>
</tr>
<tr>
<td>512</td>
<td>83.28%</td>
<td>53.27%</td>
<td>83.64%</td>
<td>55.23%</td>
</tr>
</tbody>
</table>

Table 4.3.4: K-means using different number of clusters for feature set 3.

### Supervised (random forest)

<table>
<thead>
<tr>
<th># trees</th>
<th>Accuracy train set</th>
<th>Accuracy test set</th>
<th>F1 train set</th>
<th>F1 test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>100%</td>
<td>61.39%</td>
<td>100%</td>
<td>62.29%</td>
</tr>
<tr>
<td>64</td>
<td>100%</td>
<td>63.88%</td>
<td>100%</td>
<td>64.59%</td>
</tr>
<tr>
<td>128</td>
<td>100%</td>
<td>65.14%</td>
<td>100%</td>
<td>65.74%</td>
</tr>
<tr>
<td>256</td>
<td>100%</td>
<td>65.59%</td>
<td>100%</td>
<td>66.24%</td>
</tr>
<tr>
<td>500</td>
<td>100%</td>
<td>66.03%</td>
<td>100%</td>
<td>66.63%</td>
</tr>
<tr>
<td>1000</td>
<td>100%</td>
<td>66.37%</td>
<td>100%</td>
<td>66.90%</td>
</tr>
</tbody>
</table>

Table 4.3.5: Random forest using different number of trees for feature set 3.

### Supervised (SVM)

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Accuracy train set</th>
<th>Accuracy test set</th>
<th>F1 train set</th>
<th>F1 test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>80.69%</td>
<td>59.31%</td>
<td>81.87%</td>
<td>60.62%</td>
</tr>
<tr>
<td>Poly</td>
<td>83.96%</td>
<td>57.22%</td>
<td>85.27%</td>
<td>59.21%</td>
</tr>
<tr>
<td>RBF</td>
<td>92.13%</td>
<td>65.64%</td>
<td>92.54%</td>
<td>67.03%</td>
</tr>
<tr>
<td>Sigmoid</td>
<td>48.41%</td>
<td>41.49%</td>
<td>49.24%</td>
<td>42.18%</td>
</tr>
</tbody>
</table>

Table 4.3.6: Support vector machine with different kernels for feature set 3.

### Supervised (SVM)

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Accuracy train set</th>
<th>Accuracy test set</th>
<th>F1 train set</th>
<th>F1 test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>99.71%</td>
<td>64.10%</td>
<td>99.75%</td>
<td>65.59%</td>
</tr>
<tr>
<td>Poly</td>
<td>66.38%</td>
<td>45.96%</td>
<td>69.88%</td>
<td>47.02%</td>
</tr>
<tr>
<td>RBF</td>
<td>92.70%</td>
<td>68.32%</td>
<td>93.34%</td>
<td>69.35%</td>
</tr>
<tr>
<td>Sigmoid</td>
<td>58.12%</td>
<td>47.11%</td>
<td>60.40%</td>
<td>49.18%</td>
</tr>
</tbody>
</table>

Table 4.3.7: Support vector machine with different kernels for feature set 11.
4.4 Discussion

In Table 4.3.1 it can be seen that feature sets 3-5 seems to achieve the highest scores for k-means. This is in concordance with the spherical k-means results in Table 4.3.2 which also presents the highest scores for feature set 3-5. For both these unsupervised models, the results indicate that more features does not necessarily aid the classifier, and can actually make the results worse than using less features. In Table 4.3.4 it can be seen that higher scores are achieved the more number of clusters that is implemented. Though, there is not a big difference in performance using more than 64 clusters according to the test set scores.

Worth noting for the unsupervised clustering methods is that the scores for the training sets does not matter as much as the test sets, since it is more important to see how good new data fits in the formed clusters after the model has been fitted to the training set. Usually, labels for the input samples do not exist when implementing unsupervised methods, making it hard to evaluate performance. However, the unsupervised models have been implemented in the same manner as usually done for supervised models, since labels for the data set are available, true model performance can be observed.

The random forest seem to perform better the more features that are extracted since feature set 11 achieved the highest scores, this can be seen in Table 4.3.3. Though, it might not be worth the performance boost going from 100 features (feature set 6) to 600 features (feature set 11) for barely a percentage point. Furthermore, in Table 4.3.5 it can be seen that higher scores are obtained the more trees that make up the forest. However, there is not a big difference in the performance scores going from 128 trees to 1000 trees, meaning there seems to be a sweet spot between model simplicity and model complexity. Also, a note on the 100% accuracy and F1-scores on the training set for the random forests: This is a result of having no maximal depth of the decision trees that make up the forest.

For the SVM, a clearly better result in performance can be observed when switching from feature set 3 to feature set 11, as seen in Tables 4.3.6 and 4.3.7. This is not surprising as feature set 11 contain more features compared to feature set 3. The exception being of the poly-kernel which is performing worse. Sigmoid has the greatest improvement going from feature set 3 to 11, but is still worse than RBF. The RBF-kernel is scoring the highest in both feature sets, with a slight increase in the F1-score going from feature set 3 to feature set 11, but at the steep cost of having to use 12 times more features.

In general, the flattening method which is the basis for feature set 1 and 2 seems to perform poorly overall and will be avoided going forward. This is most likely due to the fact that it is highly sensitive when the duration of a sound event is shorter than the total length of the audio clip and could have a
different onset and offset than other events of the same classification.
Chapter 5

Methods

As previously shown in chapter 1, there are multiple ML models that can be implemented for the purpose of SED. The building of a SED system can be quite challenging since a lot of factors will affect the performance of the system, from the audio sampling as a first step in the pipeline, to the feature extraction and later the application of a ML model, and everything in between. What audio features are good and informative for the task at hand? What suitable ML models can be applied and how can the performance be evaluated? There is no general right answer for these questions, and it will be different for different scenarios depending on the goals.

The data that we have access to only have weak labels, i.e. there is only one time stamp for each event that the sensor detects. This time stamp lies somewhere in between the onset and offset times in regards to the produced noise the vehicles makes when passing the sensor. Since vehicles do not pass by instantaneously, strong labels are necessary in order to build a supervised SED system. In this case, strong labels refer to the onset and offset times being annotated for each event. Each audio clip could be manually annotated by us, meaning that each clip is listened to one by one and then the onset and offset times for each event is annotated by hand. However, as discussed in the introduction, this task is very laborious. Also, it invites for inconsistency, since the event onset and offset times is subjective, it can be interpreted different by different people. Therefore, it would be nice to have this done automatically for two reasons: First, to save both time and effort and second, to have the annotations uninfluenced by the subjective nature of human sound interpretation. To solve this problem, we built an unsupervised SED system (described in section 5.6), with the sole purpose of creating strong labels that can be used for SED in a supervised manner (described in section 5.7).

Essentially, starting out with weak labels and ending up with strong labels that can be used for supervised SED. The whole process is illustrated in Figure 5.0.1.
Figure 5.0.1: Flow chart of the SED process.

In Figure 5.0.2, an illustration of the weak-to-strong labels steps from the flow chart are presented. Figure 5.0.3 illustrates how a noise-data trained supervised system can predict events based on the strong labels generated by the unsupervised system.
Figure 5.0.2: An example of a vehicle pass-by recording where log-mel energies are used as features for the ML model. Given a weak label in the neighbourhood of a vehicle pass-by, strong labels are generated by the unsupervised SED system.

Figure 5.0.3: An example of a vehicle pass-by recording where log-mel energies are used as features for the ML models. Strong labels are used for training a supervised classifier. The event as detected by the unsupervised system is used as a reference to compare the predicted event from the supervised system.

The methods that will be described below were done based on our own ideas and the experience obtained in chapter 4, however, Ykhlef et al. [14] essentially present a similar setup for the feature extraction as the one performed by us. Also, they formulated the supervised ML problem in a
simple informative way. In light of this, we will borrow some of their mathematical notations and formulations for sections 5.2, 5.3, 5.6 and 5.7.

5.1 Data Acquisition & Preparation

The setup for monitoring traffic is located along the four-laned Hornsgatan in Stockholm. The setup consists of two cameras, one in each direction, one microphone that continuously records the environment (see the green circle in Figure 5.1.1), and four sets of radar sensors that records when a vehicle passes in any respective lane (seen as rectangles in Figure 5.1.1).

The data used in this project was gathered over multiple days starting from the 8th of January and ongoing until 29th January. During this time, the radar detected 447,574 traffic events and most of them were captured when multiple vehicles were present in some of the lanes, effectively polluting the quality of the audio stream. Out of these events, 277,317 license plates were captured by the cameras. In order to make the traffic events sparse, audio clips needs to be extracted from the audio stream in such a way that only one vehicle pass-by is allowed during a certain time period. The radar saves tags of detected events, i.e. a timestamp of when an event is detected. 30 second long audio recordings are extracted from the audio stream using the radar tags as rough center points for each event that fulfills the condition that no other event was detected either 20 seconds before or 20 seconds after the considered event. Most audio recordings extracted this way would therefore only contain one vehicle pass-by. Within this time period, 2326 noise events fulfilled these requirements and they were extracted resulting in 2326 audio recordings. Out of these audio recordings, only 200 will make up the sparse traffic noise data set that will be the basis of the SED systems due to time constraints. However, for the sake of testing robustness and generalization of the SED systems, only two runs will be performed using all 2326 recordings. Furthermore, among these recordings, only 913 could be associated with an image taken of the license plate from one of the two cameras, enabling specific vehicle information to be extracted based on the license plate. Thus, only these 913 audio recordings will be used for some of the unsupervised classification.

5.2 Formulation of the ML Problem

A straightforward ML approach would be to formulate the SED system in a binary classification framework, meaning that the classifier only will assign labels with one of two possible values to individual input samples. In essence, samples are classified as "class active" or "class not active", indicating only if the sound/event in question is present or not in a given sample. Let
\(y \in \mathcal{Y} = \{0, 1\}\), where \(y\) is a class label and \(\mathcal{Y}\) is the space of labels. Also, let \(x \in \mathcal{X}\), where \(\mathcal{X}\) is the feature space and each sample \(x\) is a feature vector.

Now, the classification problem can be formulated as

\[
\hat{y} = f(x, \theta),
\]

where the classification model is the estimated mapping function \(f\), which take feature vector \(x\), and some model parameters \(\theta\), as input, and produces and output label \(\hat{y}\).

5.3 Feature Extraction

Let \(\Omega\) be the set of \(r\) audio recordings \(\Omega = \{\omega_i \mid i = 1, \ldots, r\}\). To begin with, the audio signal \(\omega_i\) is divided into \(m_i\) overlapping frames, where each frame has been subject to a Hann window in order to smooth the frame boundaries. Next, a feature vector \(x_{im} \in \mathbb{R}^N\) is extracted from each frame \(m \in (1, \ldots, m_i)\), where \(N\) denotes the number of extracted features. In this case, the \(N\) features will be generated by one of the following four cases: First, given that log-mel energies are extracted for each frame from a log-mel spectrogram, each element of \(x_{im}\) will be the log-mel energy and there will be one element for each mel band, hence \(N\) will be the number of mel bands. Second, given that MFCCs are extracted for each frame, each element of \(x_{im}\) will be the MFCC and there will be one element for each coefficient, hence \(N\) will be the number of MFC coefficients. Third, \(\Delta\) and \(\Delta\Delta\) features are extracted for either log-mel energies or MFCCs, and appended to respective base features. For example, if MFCC + \(\Delta\)MFCC are extracted using 25 MFCCs, \(N\) will have
length 50 (25+25). Fourth, if instead a statistical feature set is used, the elements of \( x_i^m \) will be a given by the elements of the feature set, and \( N \) will be the length of the feature set.

The \( x_i^m \) feature vectors can be stored in a matrix \( x^i \in \mathbb{R}^{m_i \times N} \), and done for all audio recordings \( \omega_i \), we obtain a new matrix \( \chi \in \mathbb{R}^{M \times N} \): a feature representation of the whole data set \( \Omega \), where \( M = \sum_{i=1}^{r} m_i \).

Note, in some of the following sections, the word "frame" are used interchangeably with "feature vector" in the context of each feature vector being an individual sample for the ML classifiers. If it is not otherwise apparent by the context, the word "frame" is really referring to the feature vector \( x_i^m \in \mathbb{R}^N \) extracted from frame \( m \). The motivation behind this is to try to keep the theory intuitive and easy to follow, and each frame can thus logically be thought of as an input sample for a ML classifier.

### 5.4 Smoothen Classifications

Since all frames are considered as individual samples according to the ML models (described in sections 5.6-5.7), and due to the complexity of audio recordings, classification errors are inevitable. Missclassified frames can cause major issues for the overall performance of the SED system and needs to be mitigated. To solve this problem, smoothening methods will be implemented on the model frame predictions. Two basic rules that will smoothen model predictions on a frame basis was inspired by Lu et al. \cite{29}. They can be presented as follows

\[
\text{Rule 1: if } s(m) = c \text{ and } s(m+1) \neq c \text{ and } s(m+2) = c, \quad \text{then } s(m+1) = c. \\
\text{Rule 2: if } s(m) = c \text{ and } s(m-1) \neq c \text{ and } s(m+1) \neq c, \quad \text{then } s(m) = s(m-1),
\]

where \( s(m) \) denotes the prediction at frame \( m \), and \( c \) denotes the class label.

The motivation of the two rules is best illustrated by two examples:

\[
\begin{align*}
[0, 1, 0, 2, 2, 0, 2, 2, 2] &\rightarrow [0, 0, 0, 2, 2, 2, 2, 2, 2] \text{ by rule 1,} \\
[0, 0, 1, 2, 2, 0, 0, 0] &\rightarrow [0, 0, 0, 2, 2, 0, 0, 0, 0] \text{ by rule 2.}
\end{align*}
\]

The effect of rule 1 is essentially as follows: whenever there is a frame label of a particular class that is sandwiched between frame labels of another class, the middle label will be considered as a misclassification and will get smoothened to the same class as its adjacent frame labels. In a similar manner, in a segment of three consecutive frames where all labels are different from each
other, rule 2 will smoothen the middle frame label to the previous frame label.

The smoothening rules described in equation (5.2) is done on a frame-by-frame basis, and since each frame is quite short, this can be considered as smoothening on a micro level. Each frame is not considered as an event by itself due to its short duration, rather each event-frame is contributing to creating a whole event-segment. Thus, it would be fruitful to also implement smoothening rules on an event-segment basis, i.e. smoothening on a macro level. An event-segment is defined as a group of multiple frames that are next to each other, all of which belonging to the same class.

For this purpose, we have essentially implemented the same rules described in equation (5.2), but applied on whole event-segments. Each segment will be compared to its neighbouring segments and smoothened in a logical way. One event-segment is considered to be all adjacent frames that is of the same class. Many event-segments will be short in duration due to the nature of the frame-by-frame classifications. In fact, some of them will be too short to be correctly interpreted as an event. Therefore, it is crucial to solve the problem of short events and smoothen them into bigger event-segments. All event-segments that is less than 1.2 seconds will be considered too short and will be smoothened. The smoothening applied here is best illustrated with an example as seen in Figure 5.4.1

Figure 5.4.1: An example of a vehicle pass-by recording where log-mel energies is the base feature used for clustering frames with k-means. Three clusters where used and is denoted by different colors, yellow, blue and magenta. In this case, the magenta event-segment is containing all the frames where a vehicle was considered detected.
5.5 Classification Rules

In order to evaluate the performance of the SED systems, rules determining when predicted events are classified as correct or not need to be established. There are multiple ways to do this and different classification rules can dramatically affect the overall performance. The rules described in this section only apply to the event-based evaluation scores, since calculating the performance on a frame-by-frame basis is straightforward. One simple classification rule that will be implemented is described by Mesaros et al. [30], and it introduces the notion of collars, which are tolerances applied to the onset and offset of each true event. A predicted event will only be classified as correct if: 1. the predicted event’s onset is within a ±1 second collar of the true onset and 2. the predicted offset is within a ±50% collar of the true offset with respect to the true event duration. The adaptive nature of the offset collar allows for sound events to be correctly classified since it covers differences between short and long event durations. Due to the use of collars, these event based scores will be referred to as the *collar-based accuracy score* and the *collar-based F1-score*. Naturally, both scores will get affected if the SED systems detects false positives, however, since the F1-score takes into account false positives in addition to true positives and false negatives, it will be the more informative score of the two.

5.6 Unsupervised SED

What could be considered as a relatively simple approach was opted for when building the unsupervised SED system that will generate the strong labels. This system will have an audio signal as input and then output time stamps (onset and offset times) for the detected events. The system is based on a frame-by-frame classification approach and is described in details below.

Audio sampling: The audio recordings are resampled to a sampling rate of 22050 Hz, and if any recording is stereo, it will be converted to mono.

Feature extraction: Log-energies (feature set i) or MFCCs (feature set iv), see Table 5.7.1, will be used for the clustering of frames. Features are extracted using a FFT frame size of $N_{FFT} = 4096$, and a hop length of 2048. This will make the duration of each frame to be 92.88 ms and there will be 50% overlap between frames. 64 mel bands were implemented and only the first 25 MFCCs were kept.

Clustering frames with ML: First, audio sample $\omega_i$ is divided into $m_i$ frames and a feature vector $x_i^m \in \mathbb{R}^N$ is extracted from each frame $m \in m_i$. Second, the feature vectors are stored in a matrix $x^i \in \mathbb{R}^{m_i \times N}$, which is used as input for the k-means or spherical k-means classifiers. The basic idea is to try to find patterns among frames, and cluster each frame according to similarity. For
example, if one car passes the sensor close to the 15 second mark, in a 30 second long audio recording, it is expected that the feature vectors $x^i_m$ will be noticeably different for all frames within a multiple second long interval containing the 15 seconds time point. This is because, ideally, everything outside this interval would be silent, and when the car passes by, there would be an increase in the noise level during that time interval. Therefore, in theory, all feature vectors $x^i_m$ where a vehicle is far away from the sensor will be classified as one cluster, and the feature vectors that are near/in the time interval when the vehicle passes by will be classified as another cluster. This way, frames are differentiated based on the information that is contained in the feature vectors. After using $x^i \in \mathbb{R}^{m_i \times N}$ as input for k-means, the output for each audio sample $\omega_i$ will be a vector $s^i \in \mathcal{C}^{m_i}$, where $m_i$ denotes the amount of frames that $\omega_i$ was divided into and $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$ denotes the set of $k$ clusters.

Next, the amount of clusters to use when applying k-means or its spherical counterpart need to be determined. The most logical amount would be to use only two clusters, one cluster containing frames that indicate when noise producing events are active, and all the remaining frames would be associated with background noise. However, this might not be the best number of clusters to use in practical application due to the nature of polyphonic environments. Consider a regular street in a big city, and all the sounds that could be present in any given audio recording, vehicles passing by, people talking, birds chirping, construction work and so on. To cluster frames when everything aforementioned can be a sound source present, in only two clusters might not be the best approach. This is backed up by the results presented in Table 4.3.4 where it was noticed that it might be better to implement more clusters than actual classes in the data set. Thus, the performance of using different amount of clusters for the unsupervised models must be tested. To this end, frames will be clustered in two, three and four clusters.

**Smoothen classifications:** Once the frames has been clustered by k-means or spherical k-means, the smoothening functions described in section 5.4 will be applied in order to get the resulting event-segments which will be the basis for the next step.

**Locating active event-segments:** Now all frames have been divided into event-segments, which is characterized by a cluster group. The problem is that it is not known which event-segments are active or not, i.e. which event-segments that contain the vehicle pass-by and which only contain background noise. Since the clustering is done in an unsupervised manner, a robust way to find the event-segments of interest is needed. A statistical approach is proposed to solve this problem: For log-mel energies, the mean energy over all mel bands and all frames in event-segments of the same cluster group is added and averaged. All event-segments that are associated with the cluster group that had the highest average energy will be labeled as the
event-segments of interest, and it is assumed that these segments will contain
the pass-by vehicles. A similar approach is done if MFCCs are used, just in
this case, only the first MFC coefficient will be the basis for the mean
calculation. This way, when noise is present in particular event-segments that
is louder than that of the background, it is more likely that those particular
event-segments will be determined as the events of interest.

Every event-segment has two timestamps, a start time and an end time. For a
particular active event-segment, the start time of that event is referred to as
the onset and the end time is referred to as the offset. These are the
timestamps that will be used for annotating the audio in the supervised SED
system (described in section 5.7).

Removing outliers: The audio recordings are cut into 30 second clips such that
the radar detection tag is placed in the middle around the 15 second mark. All
the recordings will be times stamped, indicating where the events is active
(detected event-segments), and where there is no active events (background
noise), and any recording with detected events located in the first or last 5
seconds will be deemed as an outlier and removed from the data set.

System evaluation: How do we know that the system has successfully
identified the onsets and offsets for all pass-by vehicles in an audio recording
given the active event-segments obtained from the last step? Unfortunately,
this question can not be answered, at least not in a sense that is satisfactory.
Since the system is unsupervised, there really is no robust way to determine
the results. What can be done, however, is to randomly pick some audio
recording from the whole data set, and listen to them and determine if the
timestamps from the unsupervised system correspond to the onsets and offsets
of the noise produced by pass-by vehicles present in the audio recordings. As
mentioned before, there is a subjective nature in determining precise onsets
and offsets of sound events, however, this will not be considered as a problem
since most of the vehicle pass-by durations span multiple seconds. Thus, super
precision is not important and minor differences in evaluating the true onsets
and offsets will not have any meaningful impact on the system. If most of the
predicted vehicle pass-by timestamps from the randomly picked samples is in
concordance with the perceived timestamps based on manual evaluation, it
can be inferred that the performance will be similar for the whole data set.

5.7 Supervised SED

The supervised SED system is built in a similar frame-by-frame structure as
described in section 5.6. However, now that strong labels have been obtained
from the unsupervised system, each frame can be labeled as active or not
active, in order to apply supervised ML models. The system is described in
details below.
Audio sampling: The audio recordings is resampled to a sampling rate of 22050 Hz, and if any recording is stereo, it will be converted to mono.

Feature extraction: Table 5.7.1 presents the different feature sets that will be used as features for training and testing the SED systems. Features are extracted using a FFT frame size of $N_{\text{FFT}} = 4096$, and a hop length of 2048. This will make the duration of each frame to be 92.88 ms and there will be 50% overlap between frames. 64 mel bands were implemented and only the first 25 MFCCs were kept.

<table>
<thead>
<tr>
<th>Feature set No.</th>
<th>Audio features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature set i</td>
<td>log-mel</td>
</tr>
<tr>
<td>Feature set ii</td>
<td>log-mel+$\Delta$log-mel</td>
</tr>
<tr>
<td>Feature set iii</td>
<td>log-mel+$\Delta$log-mel+$\Delta\Delta$log-mel</td>
</tr>
<tr>
<td>Feature set iv</td>
<td>MFCC</td>
</tr>
<tr>
<td>Feature set v</td>
<td>MFCC+$\Delta$MFCC</td>
</tr>
<tr>
<td>Feature set vi</td>
<td>MFCC+$\Delta$MFCC + $\Delta\Delta$MFCC</td>
</tr>
</tbody>
</table>

Table 5.7.1: Features extracted using log-mel energies or MFCCs as base.

Labeling data: The timestamps obtained from the unsupervised SED system described in section 5.6, enables the creation of annotations for each audio sample. That is, every audio sample $\omega_i$ is divided into $m_i$ frames, and each frame can be labeled with either a one or a zero depending on whether or not the given frame falls within active-event intervals defined by the timestamps. More formally, the resulting annotated feature representation of the data set can be presented as

$$\chi_{\text{anno}} = \{(x_{i}^m, y_{i}^m), \ldots, (x_{m_i}^i, y_{m_i}^i), \ldots, (x_{m}^i, y_{m}^i) \mid i = 1, \ldots, r\}, \quad (5.3)$$

where $x_{i}^m \in \mathbb{R}^N$ denotes the feature vector extracted from frame $m$ of audio recording $\omega_i$, and $y_{m_i}^i = 1$ if the sound/event occurs in frame $m$ and $y_{m_i}^i = 0$ otherwise. Each frame $m$ of every audio recording $\omega_i$, is considered as an individual sample in the data set $\chi_{\text{anno}}$.

Training ML classifier: Equation (5.3) presents the main idea of labeling individual frames for all audio recordings in the whole data set $\Omega$. However, as is customary for ML problems, the data set $\Omega$ will be divided into a training set $\Omega_{\text{train}}$ and a test set $\Omega_{\text{test}}$, and it will require unnecessary work to perform the train-test split on $\chi_{\text{anno}}$. The reason for this is that one needs to keep track of which frame feature vector $x_{m}^i$ is coming from which audio recording and at what frame position. Therefore it will be much easier to perform the
train-test set split at an audio recording level. The whole set of recordings $\Omega$ will be randomly sampled and split into $\Omega_{\text{train}}$ and $\Omega_{\text{test}}$, where 90% of $\Omega$ will be used for model training and 10% of $\Omega$ will be used for model testing. This is done 10 times over through the process of 10-fold cross validation. The labeling described in equation (5.3) will be done for both $\Omega_{\text{train}}$ and $\Omega_{\text{test}}$, resulting in labeled feature representations $\chi_{\text{train}}$ and $\chi_{\text{test}}$. Also, feature standardization is performed on both training and test sets. Next, a supervised ML model (random forest or support vector machine) will be trained on $\chi_{\text{train}}$.

**Testing ML classifier:** Once the ML model have been trained, it will make predictions on $\chi_{\text{test}}$, that is, each individual frame will be labeled by the classifier as either active ($y = 1$), or not active ($y = 0$).

**Smoothen classifications:** When the frames has been labeled by the classifier, they will be subject to the smoothening functions described in section 5.4. Once this has been done, event-segments are obtained and defined as either event active (vehicle present) or event not active (vehicle not present). The onset and offset timestamps for each active event-segment is saved and will be the basis for evaluation of the supervised SED system.

**System evaluation:** Two different evaluations will be done, one will be calculated of predictions on a frame-by-frame basis and the other one will be calculated on an event-segment basis. Accuracy and F1-score will be calculated on an frame-by-frame basis. This is easily done since only the label of each predicted frame will be compared to the corresponding reference label found in $\chi_{\text{test}}$. The frame-by-frame evaluation basically provides information about how well the system can make predictions based on each individual frame feature vector. In other words, how well the system can make predictions based on very short temporal durations.

For the sake of SED, the more important scores are given by the collar-based accuracy score and the collar-based F1-score, where event-segments are classified according to the rules described in section 5.5. The scores based on whole sound events rather than on an frame-by-frame basis can be considered as more important since finding the onset and offset times of sound events is a central goal of many SED systems.

### 5.8 Unsupervised Vehicle Classification

The unsupervised classification is built upon the results of the SED described in section 5.6. After onset and offset of the traffic noise has been successfully detected by the unsupervised methods, the vehicles are then grouped by use of k-means and spherical k-means. Spherical k-means is used in order to hopefully be able to cluster the sounds by their characteristics, instead of their noise levels. In this case, the unsupervised methods are not used on the time
frames of a single event, but by the statistical features (see Table 4.2.1) of all
time frames that belongs to each event, which are the basis of how all the
events are clustered. The resulting clusters are then evaluated based on their
known classifications. These classifications are provided partly from the radar
that detects the traffic in the first place, but also from a traffic camera that
captures license plates and provides registered data associated with that plate.
The radar and camera are not integrated in the same system, so each data
point has to be matched with each other. A grace period of 5 seconds has been
implemented to account for unsynchronized time stamps of the systems. Of
the 2248 events used, 913 could be tied to license plates detected by the
camera. The clustering is done several times, with \( n \) numbers of clusters,
where \( n \) is the number of unique classifications that exists of different
categories. Some categories are excluded from testing due to being irrelevant,
for example, the color of the vehicle, or because they are deemed too infeasible
to cluster, such as the age of the vehicle.
Chapter 6

Results

6.1 Unsupervised SED

Figure 6.1.1: Comparison of time events based on k-means (top) and spherical k-means (bottom).

Figure 6.1.2 shows how the events gets shorter for higher number of clusters.
Figure 6.1.2: Comparison of time events based on k-means with 2, 3, and 4 clusters respectively from top to bottom.

As seen in the top half of Figure 6.1.3, some events were detected at the start and/or end of the audio clip. These were deemed as outliers as the recordings were intentionally cut such that the radar detection was centered in the clips. The bottom half of the Figure 6.1.3 show the data set with the 78 outliers removed, resulting in total of 2248 recordings.
Figure 6.1.3: Comparison of time event before and after outliers are removed from the data set.

6.2 Supervised SED

6.2.1 Random Forest

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>Accuracy (frame-by-frame)</th>
<th>F1 (frame-by-frame)</th>
<th>Collar-based accuracy</th>
<th>Collar-based F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>95.41%</td>
<td>93.03%</td>
<td>79.00%</td>
<td>43.94%</td>
</tr>
<tr>
<td>ii</td>
<td>95.39%</td>
<td>92.96%</td>
<td>78.50%</td>
<td>43.84%</td>
</tr>
<tr>
<td>iii</td>
<td>95.36%</td>
<td>92.86%</td>
<td>78.00%</td>
<td>43.70%</td>
</tr>
<tr>
<td>iv</td>
<td>95.21%</td>
<td>92.64%</td>
<td>75.00%</td>
<td>42.66%</td>
</tr>
<tr>
<td>v</td>
<td>95.19%</td>
<td>92.54%</td>
<td>76.00%</td>
<td>42.95%</td>
</tr>
<tr>
<td>vi</td>
<td>95.19%</td>
<td>92.58%</td>
<td>76.00%</td>
<td>42.97%</td>
</tr>
</tbody>
</table>

Table 6.2.1: Evaluation scores for different feature sets and k-means (3 clusters) for the generation of strong labels.
Random forest (500 trees) & spherical k-means for strong labels

<table>
<thead>
<tr>
<th>Feature</th>
<th>Accuracy (frame-by-frame)</th>
<th>F1 (frame-by-frame)</th>
<th>Collar-based accuracy</th>
<th>Collar-based F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>83.44%</td>
<td>71.32%</td>
<td>15.34%</td>
<td>12.99%</td>
</tr>
<tr>
<td>ii</td>
<td>83.40%</td>
<td>71.22%</td>
<td>13.98%</td>
<td>11.93%</td>
</tr>
<tr>
<td>iii</td>
<td>83.37%</td>
<td>70.44%</td>
<td>12.90%</td>
<td>11.03%</td>
</tr>
<tr>
<td>iv</td>
<td>82.26%</td>
<td>68.89%</td>
<td>8.24%</td>
<td>7.52%</td>
</tr>
<tr>
<td>v</td>
<td>82.77%</td>
<td>69.78%</td>
<td>11.50%</td>
<td>10.13%</td>
</tr>
<tr>
<td>vi</td>
<td>82.14%</td>
<td>68.03%</td>
<td>8.96%</td>
<td>8.12%</td>
</tr>
</tbody>
</table>

Table 6.2.2: Evaluation scores for different feature sets and spherical k-means (3 clusters) for the generation of strong labels.

6.2.2 Support Vector Machine

Support vector machine (RBF-kernel) & k-means for strong labels

<table>
<thead>
<tr>
<th>Feature</th>
<th>Accuracy (frame-by-frame)</th>
<th>F1 (frame-by-frame)</th>
<th>Collar-based accuracy</th>
<th>Collar-based F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>95.33%</td>
<td>92.93%</td>
<td>76.10%</td>
<td>42.89%</td>
</tr>
<tr>
<td>ii</td>
<td>95.59%</td>
<td>93.26%</td>
<td>76.50%</td>
<td>43.14%</td>
</tr>
<tr>
<td>iii</td>
<td>95.55%</td>
<td>93.21%</td>
<td>77.00%</td>
<td>43.25%</td>
</tr>
<tr>
<td>iv</td>
<td>95.58%</td>
<td>93.21%</td>
<td>75.00%</td>
<td>42.66%</td>
</tr>
<tr>
<td>v</td>
<td>95.44%</td>
<td>92.99%</td>
<td>76.00%</td>
<td>42.96%</td>
</tr>
<tr>
<td>vi</td>
<td>95.47%</td>
<td>93.04%</td>
<td>75.00%</td>
<td>42.61%</td>
</tr>
</tbody>
</table>

Table 6.2.3: Evaluation scores for different feature sets and k-means (3 clusters) for the generation of strong labels.

Support vector machine (RBF-kernel) & spherical k-means for strong labels

<table>
<thead>
<tr>
<th>Feature</th>
<th>Accuracy (frame-by-frame)</th>
<th>F1 (frame-by-frame)</th>
<th>Collar-based accuracy</th>
<th>Collar-based F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>83.79%</td>
<td>73.04%</td>
<td>14.95%</td>
<td>12.69%</td>
</tr>
<tr>
<td>ii</td>
<td>83.60%</td>
<td>72.31%</td>
<td>14.19%</td>
<td>12.07%</td>
</tr>
<tr>
<td>iii</td>
<td>83.57%</td>
<td>72.20%</td>
<td>13.93%</td>
<td>11.95%</td>
</tr>
<tr>
<td>iv</td>
<td>83.56%</td>
<td>72.83%</td>
<td>13.74%</td>
<td>11.90%</td>
</tr>
<tr>
<td>v</td>
<td>83.68%</td>
<td>73.19%</td>
<td>14.21%</td>
<td>11.95%</td>
</tr>
<tr>
<td>vi</td>
<td>83.57%</td>
<td>72.95%</td>
<td>14.59%</td>
<td>12.43%</td>
</tr>
</tbody>
</table>

Table 6.2.4: Evaluation scores for different feature sets and spherical k-means (3 clusters) for the generation of strong labels.
6.2.3 RF & SVM on the Whole Sparse Data Set

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Accuracy (frame-by-frame)</th>
<th>F1 (frame-by-frame)</th>
<th>Collar-based accuracy</th>
<th>Collar-based F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF</td>
<td>94.09%</td>
<td>89.51%</td>
<td>64.32%</td>
<td>39.13%</td>
</tr>
<tr>
<td>SVM</td>
<td>94.52%</td>
<td>90.46%</td>
<td>68.44%</td>
<td>40.62%</td>
</tr>
</tbody>
</table>

Table 6.2.5: Evaluation scores for RF and SVM on the whole sparse data set (2248 audio recordings) using feature set i and k-means (3 clusters) for the generation of strong labels.

6.3 Unsupervised Vehicle Classification

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Vehicle class 0</th>
<th>Vehicle class 1</th>
<th>Vehicle class 2</th>
<th>Vehicle class 5</th>
<th>Vehicle class 12</th>
<th>Vehicle class 15</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3 (5)</td>
<td>14 (12)</td>
<td>257 (222)</td>
<td>10 (199)</td>
<td>103 (125)</td>
<td>3 (1)</td>
<td>540 (554)</td>
</tr>
<tr>
<td>2</td>
<td>2 (1)</td>
<td>13 (16)</td>
<td>260 (211)</td>
<td>134 (136)</td>
<td>83 (90)</td>
<td>1 (0)</td>
<td>493 (454)</td>
</tr>
<tr>
<td>3</td>
<td>4 (2)</td>
<td>4 (7)</td>
<td>109 (200)</td>
<td>134 (55)</td>
<td>81 (45)</td>
<td>1 (2)</td>
<td>333 (321)</td>
</tr>
<tr>
<td>4</td>
<td>2 (2)</td>
<td>10 (2)</td>
<td>137 (103)</td>
<td>108 (115)</td>
<td>61 (92)</td>
<td>1 (1)</td>
<td>319 (315)</td>
</tr>
<tr>
<td>5</td>
<td>2 (3)</td>
<td>4 (6)</td>
<td>84 (118)</td>
<td>112 (124)</td>
<td>90 (58)</td>
<td>3 (4)</td>
<td>295 (313)</td>
</tr>
<tr>
<td>6</td>
<td>0 (0)</td>
<td>16 (8)</td>
<td>143 (136)</td>
<td>56 (85)</td>
<td>51 (59)</td>
<td>2 (3)</td>
<td>268 (291)</td>
</tr>
</tbody>
</table>

Table 6.3.1: Comparison of clustering methods using feature set 3 and vehicle classifications from radar data. Sorted by cluster size.

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Vehicle class 0</th>
<th>Vehicle class 1</th>
<th>Vehicle class 2</th>
<th>Vehicle class 5</th>
<th>Vehicle class 12</th>
<th>Vehicle class 15</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3 (3)</td>
<td>13 (12)</td>
<td>254 (217)</td>
<td>192 (161)</td>
<td>89 (90)</td>
<td>4 (3)</td>
<td>515 (491)</td>
</tr>
<tr>
<td>2</td>
<td>2 (2)</td>
<td>13 (9)</td>
<td>243 (144)</td>
<td>140 (139)</td>
<td>86 (113)</td>
<td>1 (3)</td>
<td>485 (410)</td>
</tr>
<tr>
<td>3</td>
<td>0 (2)</td>
<td>18 (12)</td>
<td>184 (155)</td>
<td>68 (127)</td>
<td>63 (72)</td>
<td>2 (1)</td>
<td>335 (369)</td>
</tr>
<tr>
<td>4</td>
<td>3 (1)</td>
<td>10 (16)</td>
<td>132 (240)</td>
<td>113 (72)</td>
<td>63 (31)</td>
<td>0 (2)</td>
<td>321 (362)</td>
</tr>
<tr>
<td>5</td>
<td>3 (3)</td>
<td>3 (4)</td>
<td>100 (118)</td>
<td>131 (121)</td>
<td>80 (96)</td>
<td>1 (1)</td>
<td>318 (343)</td>
</tr>
<tr>
<td>6</td>
<td>2 (2)</td>
<td>4 (8)</td>
<td>77 (116)</td>
<td>100 (84)</td>
<td>88 (62)</td>
<td>3 (1)</td>
<td>274 (273)</td>
</tr>
</tbody>
</table>

Table 6.3.2: Comparison of clustering methods using feature set 4 and vehicle classifications from radar data. Sorted by cluster size.
### Table 6.3.3: Comparison of clustering methods using feature set 5 and vehicle classifications from radar data. Sorted by cluster size.

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Vehicle class 0</th>
<th>Vehicle class 1</th>
<th>Vehicle class 2</th>
<th>Vehicle class 3</th>
<th>Vehicle class 4</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3 (4)</td>
<td>18 (12)</td>
<td>250 (272)</td>
<td>149 (276)</td>
<td>94 (166)</td>
<td>516 (734)</td>
</tr>
<tr>
<td>2</td>
<td>3 (3)</td>
<td>8 (9)</td>
<td>195 (157)</td>
<td>183 (189)</td>
<td>114 (151)</td>
<td>506 (511)</td>
</tr>
<tr>
<td>3</td>
<td>1 (3)</td>
<td>9 (17)</td>
<td>177 (221)</td>
<td>162 (147)</td>
<td>97 (106)</td>
<td>447 (497)</td>
</tr>
<tr>
<td>4</td>
<td>1 (2)</td>
<td>12 (3)</td>
<td>100 (124)</td>
<td>33 (25)</td>
<td>31 (2)</td>
<td>431 (233)</td>
</tr>
<tr>
<td>5</td>
<td>2 (0)</td>
<td>8 (9)</td>
<td>140 (93)</td>
<td>19 (11)</td>
<td>0 (3)</td>
<td>516 (511)</td>
</tr>
<tr>
<td>Total</td>
<td>13 (13)</td>
<td>61 (61)</td>
<td>990 (990)</td>
<td>704 (704)</td>
<td>469 (469)</td>
<td>2248 (2248)</td>
</tr>
</tbody>
</table>

### Table 6.3.4: Comparison of clustering methods using feature set 11 and vehicle classifications from radar data. Sorted by cluster size.

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Vehicle class 0</th>
<th>Vehicle class 1</th>
<th>Vehicle class 2</th>
<th>Vehicle class 3</th>
<th>Vehicle class 4</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3 (4)</td>
<td>18 (8)</td>
<td>247 (190)</td>
<td>149 (218)</td>
<td>94 (124)</td>
<td>513 (546)</td>
</tr>
<tr>
<td>2</td>
<td>3 (4)</td>
<td>8 (17)</td>
<td>194 (229)</td>
<td>180 (152)</td>
<td>113 (111)</td>
<td>501 (516)</td>
</tr>
<tr>
<td>3</td>
<td>3 (3)</td>
<td>6 (7)</td>
<td>135 (133)</td>
<td>162 (170)</td>
<td>138 (138)</td>
<td>447 (451)</td>
</tr>
<tr>
<td>4</td>
<td>1 (1)</td>
<td>9 (15)</td>
<td>174 (169)</td>
<td>161 (69)</td>
<td>93 (46)</td>
<td>439 (301)</td>
</tr>
<tr>
<td>5</td>
<td>1 (0)</td>
<td>12 (6)</td>
<td>100 (132)</td>
<td>33 (73)</td>
<td>31 (48)</td>
<td>178 (260)</td>
</tr>
<tr>
<td>6</td>
<td>2 (2)</td>
<td>8 (8)</td>
<td>140 (93)</td>
<td>19 (11)</td>
<td>0 (3)</td>
<td>170 (174)</td>
</tr>
<tr>
<td>Total</td>
<td>13 (13)</td>
<td>61 (61)</td>
<td>990 (990)</td>
<td>704 (704)</td>
<td>469 (469)</td>
<td>2248 (2248)</td>
</tr>
</tbody>
</table>

### Table 6.3.5: Comparison of clustering methods using feature set 3 and vehicle classifications from license plate camera. Sorted by cluster size.

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Electric</th>
<th>Electric Hybrid</th>
<th>Charging Hybrid</th>
<th>Euro 4</th>
<th>Euro 5</th>
<th>Euro 6</th>
<th>Unknown</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13 (14)</td>
<td>72 (53)</td>
<td>12 (13)</td>
<td>2 (0)</td>
<td>3 (8)</td>
<td>70 (87)</td>
<td>15 (6)</td>
<td>187 (181)</td>
</tr>
<tr>
<td>2</td>
<td>10 (16)</td>
<td>44 (82)</td>
<td>13 (14)</td>
<td>0 (0)</td>
<td>5 (3)</td>
<td>76 (53)</td>
<td>13 (13)</td>
<td>161 (181)</td>
</tr>
<tr>
<td>3</td>
<td>12 (11)</td>
<td>48 (71)</td>
<td>10 (5)</td>
<td>0 (2)</td>
<td>5 (2)</td>
<td>67 (45)</td>
<td>7 (17)</td>
<td>149 (153)</td>
</tr>
<tr>
<td>4</td>
<td>12 (10)</td>
<td>71 (61)</td>
<td>4 (9)</td>
<td>0 (0)</td>
<td>2 (4)</td>
<td>29 (42)</td>
<td>6 (6)</td>
<td>124 (132)</td>
</tr>
<tr>
<td>5</td>
<td>7 (4)</td>
<td>48 (33)</td>
<td>7 (3)</td>
<td>0 (0)</td>
<td>4 (3)</td>
<td>38 (61)</td>
<td>7 (16)</td>
<td>111 (120)</td>
</tr>
<tr>
<td>6</td>
<td>1 (4)</td>
<td>14 (5)</td>
<td>3 (5)</td>
<td>0 (0)</td>
<td>6 (3)</td>
<td>67 (62)</td>
<td>15 (7)</td>
<td>106 (86)</td>
</tr>
<tr>
<td>7</td>
<td>5 (1)</td>
<td>17 (9)</td>
<td>4 (4)</td>
<td>0 (0)</td>
<td>3 (5)</td>
<td>40 (37)</td>
<td>6 (4)</td>
<td>75 (60)</td>
</tr>
<tr>
<td>Total</td>
<td>60 (60)</td>
<td>314 (314)</td>
<td>53 (53)</td>
<td>2 (2)</td>
<td>28 (28)</td>
<td>387 (387)</td>
<td>69 (69)</td>
<td>913 (913)</td>
</tr>
<tr>
<td>Cluster</td>
<td>Electric</td>
<td>Electric Hybrid</td>
<td>Charging Hybrid</td>
<td>Euro 4</td>
<td>Euro 5</td>
<td>Euro 6</td>
<td>Unknown</td>
<td>Total</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>1</td>
<td>18 (14)</td>
<td>101 (67)</td>
<td>11 (20)</td>
<td>1 (1)</td>
<td>5 (6)</td>
<td>72 (88)</td>
<td>15 (21)</td>
<td>223 (217)</td>
</tr>
<tr>
<td>2</td>
<td>16 (9)</td>
<td>77 (55)</td>
<td>17 (8)</td>
<td>1 (0)</td>
<td>7 (4)</td>
<td>82 (101)</td>
<td>18 (16)</td>
<td>218 (193)</td>
</tr>
<tr>
<td>3</td>
<td>9 (10)</td>
<td>31 (77)</td>
<td>5 (15)</td>
<td>0 (0)</td>
<td>11 (3)</td>
<td>69 (64)</td>
<td>11 (12)</td>
<td>136 (181)</td>
</tr>
<tr>
<td>4</td>
<td>6 (14)</td>
<td>48 (61)</td>
<td>12 (1)</td>
<td>0 (1)</td>
<td>1 (2)</td>
<td>57 (35)</td>
<td>9 (7)</td>
<td>133 (121)</td>
</tr>
<tr>
<td>5</td>
<td>7 (8)</td>
<td>33 (23)</td>
<td>2 (2)</td>
<td>0 (0)</td>
<td>3 (9)</td>
<td>59 (52)</td>
<td>11 (8)</td>
<td>115 (102)</td>
</tr>
<tr>
<td>6</td>
<td>2 (3)</td>
<td>16 (17)</td>
<td>5 (2)</td>
<td>0 (0)</td>
<td>1 (3)</td>
<td>25 (28)</td>
<td>1 (4)</td>
<td>50 (57)</td>
</tr>
<tr>
<td>7</td>
<td>2 (2)</td>
<td>8 (14)</td>
<td>1 (5)</td>
<td>0 (0)</td>
<td>0 (1)</td>
<td>23 (19)</td>
<td>4 (1)</td>
<td>38 (42)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>60 (60)</td>
<td>314 (314)</td>
<td>53 (53)</td>
<td>2 (2)</td>
<td>28 (28)</td>
<td>387 (387)</td>
<td>69 (69)</td>
<td>913 (913)</td>
</tr>
</tbody>
</table>

Table 6.3.6: Comparison of clustering methods using feature set 11 and vehicle classifications from license plate camera. Sorted by cluster size.
Chapter 7

Discussion

7.1 Unsupervised SED - Generation of Strong Labels

7.1.1 Amount of Clusters Used For Clustering

Different amount of clusters were experimented with for standard or spherical k-means to determine the amount that would yield the best, most consistent results. Between two, three and four clusters, it was decided that three clusters were optimal for the purposes of this thesis. In general, increasing the amount of clusters made the detected event-segments more narrow, which can be seen in Figure [6.1.2]. Thus, using only two clusters, the detected events were longer in duration, meaning that the detected onset and offset times were more likely to be less obvious and more merged with the background noise. This was determined by manual listening.

Increasing to three clusters, it made the detected event slightly shorter in duration, and more inline with what could be determined to be the onset and offset times by manual listening. Having more narrow active event-segments increases the likelihood that the supervised ML model is only trained on frames that are definitely active. With more narrow event-segments, one can be more certain that feature frames between the detected onset and offset times are actually distinct from the feature frames that come before and after the detected event.

The use of four clusters made the event-segments even more narrow. In fact, so narrow that the likelihood of two potential problems increased. First, if the event-segments are too narrow, the neighbouring feature frames that come before and after the detected event will be very similar to that of the detected event. This is a problem because then, the supervised ML model will be trained on active frames that have been labeled as not active, which is
detrimental for the ability to discern active feature frames from not active ones. Second, with four clusters, event-segments are more narrow and depending on the audio recording, sound events that make up the background noise can be wrongly detected as the vehicle pass-by that is searched for by the algorithm. Thus, the ideal number of clusters was determined to be three, and it is what was used for the rest of the results.

7.1.2 Spherical vs Standard K-means

The use of k-means gave more consistent results when finding the event in each audio recording, as opposed to spherical k-means, which can be seen in Figure 6.1.1. Since the audio recordings was extracted such that the detected event was centered around the 15 second mark, the results from the k-means was generally better.

The k-means algorithm as well as the spherical k-means has shown potential to locate an onset and offset of known traffic noises. The results of them do differ a bit, with the regular k-means limiting the events more narrowly in time, this is most likely due to the fact that spherical k-means cluster data points together based on the direction of the their vectors. As such for a continuous sound of a car passing by the microphone, going from faint to loud to faint, only the magnitude change, and not the direction of the vector representing the sound. Whilst for regular k-means the magnitude matter more, therefore the vehicle pass-by will not be clustered with the noise it makes far away.

Whether or not one is better than the other is debatable, as these noises are along a road they gradually increase and it is very subjective to label when they start and stop. Had the recordings been of vehicles suddenly starting and stopping, such as a parking lot, the distinction would probably have been greater, and spherical k-means might have performed better, especially if background noise is present. The importance of distinct audio for spherical k-means to perform well can be seen from the result of the UrbanSound8K data set in Tables 4.3.1 and 4.3.2. However, in conjunction with supervised learning methods, the use of k-means for the generation of strong labels is shown to be superior, as seen when comparing Tables 6.2.1 to 6.2.2 and Tables 6.2.3 to 6.2.4.

7.2 Supervised SED

7.2.1 Feature Selection

The influence of using different feature sets for the SED systems will only be analyzed based on Table 6.2.1 and 6.2.3 since the spherical k-means performed poorly in comparison to ordinary k-means. In Table 6.2.1, it can be seen that most scores are similar using different feature sets, however, there is
a clear decrease in the collar-based accuracy score comparing feature set i with iv, v and vi. Thus, it seems that RF performs better using log-mel energies based features in comparison to MFCC based features.

Table 6.2.3 show very consistent scores for all feature sets and it seems like SVM might be slightly less sensitive to the choice of different feature sets compared to RF.

The addition of ∆- and ∆Δ- coefficients for MFCCs did not seem to improve the overall results to any significant degree. These results are in contrast to the results obtained by Ykhlef et al. [14], who obtained higher scores with the addition of ∆-features for the SVMs. It is hard to draw any conclusions regarding the influence ∆-features has on the performance of the SED systems, since differences in scores are relatively small.

For the RF, log-mel energies (feature set i) achieved highest scores across the board, although differences are minute in the frame-by-frame accuracy- and F1-scores, as can be seen in Table 6.2.1 Therefore, log-mel energies (feature set i-iii) might be preferred over MFCCs (feature set iv-vi) when RF is used. For the SVM (Table 6.2.3), differences in scores are not as apparent when comparing log-mel energies to MFFCs, and no conclusion is drawn.

7.2.2 Random Forest vs Support Vector Machine

Comparing Table 6.2.1 to Table 6.2.3 it can be seen that both RF and SVM achieved very similar results over all different scores, when k-means were implemented to generate strong labels used for training. RF performed slightly better based on the collar-based accuracy score for feature set i and ii. Differences of around one percent are not considered as very significant since they could be due to statistical differences.

When spherical k-means were used for generating strong labels, differences between RF and SVM become more apparent, as can be seen comparing Table 6.2.2 and 6.2.4. The frame-by-frame accuracy scores are very similar for both systems, however for the three other scores, SVM seems to perform at least as good as RF or noticeably better for the different feature sets, again, indicating that it might be less sensitive to different feature sets than RF.

In Table 6.2.5 RF and SVM performed similarly for all scores except the collar-based accuracy, where SVM significantly outperformed RF.

7.2.3 SED System Performance

The evaluation scores each report different information and it is important to know the difference. When the SED systems achieves high scores (accuracy and F1) on an frame-by-frame basis, it indicates that the system was good in
differentiating frames as active, or not active, meaning it was able to detect small temporal differences in the audio recording. This is important as it is the frame classifications that will be the basis for the event-scores. However, the collar-based scores can be considered more important in a SED setting since they are based on the onset and offset times of detected events, which is much more difficult to find in general. This is why the collar-based scores are lower than the frame-by-frame based scores. With this in mind, frame-by-frame accuracy- and F1-scores of $\sim 95\%$ respective $\sim (92 - 93)\%$ for both RF and SVM can be considered very good. Due to the tough task of finding onset and offset times for events, the collar-based scores, ranging from $\sim (75 - 79)\%$ for accuracy and $\sim [42 - 43]\%$ for F1 between the RF and SVM, the performance can be considered as relatively good for such simple SED systems.

For the sake of generalization and testing system robustness, two runs were done on the large data set (2248 audio recordings) for RF and SVM using log-energies for features (feature set i) and k-means (three clusters) for the generation of strong labels. Only two runs were performed due to extremely long time durations it took to run the systems. As can be seen in Table 6.2.5, the systems performed similarly using 2248 recordings as only 200, where the most apparent difference was in the collar-based accuracy scores which decreased significantly. This generalization might indicate that the results obtained using only 200 audio recordings is valid, despite the small sample size.

7.2.4 Smoothening Methods & Classification Rules

The smoothening rules described in section 5.4 are not arbitrary, rather, they are based on literature, intuition, and empirical analysis. The smoothening done on a frame-by-frame basis works as desired. However, the smoothening done on an event-segment level might seem more arbitrary since short events (less than 1.2 seconds) are smoothened out. Again, the 1.2 second threshold was chosen based on empirical testing, but different threshold can be experimented when moving forward.

The classifications rules described in section 5.5 greatly affects the collar-based scores. The chosen rules used for event classification were inspired by what could be found in the literature and modified to fit the purposes of this thesis. The collar threshold for the onset was chosen to be $\pm 1$ second since the duration of events are relatively long (roughly between 5 to 10 seconds in duration), and it seemed like a good value to start with. The offset collar is adaptive based on the event duration and this seems like a generally good approach. Shorter collars would decrease the collar-based scores and longer collars would increase them.

Furthermore, some SED systems only focus on correctly detecting the onset time of an event, which is a simpler task than to find the offset in addition to the onset. Naturally, the obtained scores would be higher if only the onset
time of an event was considered. However, it was decided to use both the onset and offset times for evaluation of the SED system since it would provide more useful information than onset only, and for the sake of system robustness.

7.2.5 Computational Cost

When running the SED systems (RF or SVM), RF seem to be noticeably faster than SVM to complete the task, which is appreciated. However, there seem to be a trade-off since RF is much more CPU-intensive, often requiring $\sim [97 - 99\%]$ CPU usage. In comparison, SVM is slower but only requires $\sim 15\%$ CPU usage.

A note on system architecture: Since the SED systems are based on a frame-by-frame approach, it is very computationally intensive to run it for many long audio recordings. The systems were run on personal computers, and due to the time constraints, the data set was limited to only 200 audio recordings.

7.3 Unsupervised Classification

As for the unsupervised classification, unfortunately, neither k-means nor spherical k-means clustering yield clear groupings of the known classifications of traffic noises. As seen in table 6.3.1-6.3.6 using different feature sets, there is no clear clustering for any classification across any of the categories available.

A possible explanation for the poor results could be that everything is based on human hearing, from the sampling rate of the microphone, to the mel scale transformation and also the computation of MFCCs. This has its successes when classifying different distinct sounds such as dog barks compared to music, but lacks distinction when applied to different car sounds. Which is understandably, since that distinction is harder for people to make as well. A better approach might be to use features that are not based on the human perception of hearing, such as sampling at a higher frequency, and transform the signal in a non-logarithmic, non-mel scale way. Perhaps keeping a linear scale, or putting more emphasis on specific frequencies through filtering as these might be the defining differences between vehicle noises, might work better.
Chapter 8
Conclusion

Based on the results presented in this thesis, machine learning methods can successfully be implemented as a tool for automated sound event detection of sparse traffic noise events. Log-mel energies and MFCCs seem to be good audio features to use for ML models, where MFCCs has the potential advantage of reducing the data dimensions without losing much information. Both of these features can be implemented for SED purposes by systems based on a frame-by-frame classification approach utilizing random forests or support vector machines. However, the addition of ∆- and ∆∆- features for log-energies or MFCCs did not seem to improve system performance in any significant way. Further research can be done in order to develop automated SED systems that perform better in the context of detection of sparse traffic noise events.

Statistics based on log-mel energies and MFCCs has been used for the classification of sound events, but lacked the capability of differentiating between classes of different vehicles. As seen in Tables 6.3.1-6.3.6, the cluster sizes had a lower variance than the underlying categories, and showed no correlation to each other. As such, a different approach would be necessary to fully classify sound events originating from traffic noise, preferably with a more balanced data set in regards to the categories that are supposed to be clustered. A more extensive analytical approach of the expected differences between the sounds will be needed, as the k-means and spherical k-means are not sufficient to cluster the sounds in regards to their log-mel or MFCC spectrums.
8.1 Further Studies

If future work is based on a similar approach as the one presented in this thesis for SED, there are multiple things that can be experimented with to potentially increase the overall performance of the system. As mentioned in the discussion, different smoothening functions can be experimented with in order to find a good and robust smoothening method. This is an important step for most SED systems that is based on a similar frame-by-frame architecture. Since all classifications are based on individual frames, some of which will be misclassified and it is the frame classifications that directly will affect all evaluation scores.

Another thing that can be fruitful to experiment with is the different classification rules implemented to evaluate the detection performance. If similar rules are used as described in section 5.5, further analysis could be done in order to determine the collar thresholds for the event-based scores. There are many other rules that have been used in the past that can be tested. Also, there might be a need to design completely new rules that are more fitting to this particular vehicle detection problem, and not based on any rules found in the literature.

Furthermore, a parameter analysis can be done to determine the influence of different parameters (e.g. \( N_{\text{FFT}} \), hop length, number of mel-bands, number of MFCCs) used for the feature extraction and how they will impact the final results. The use of other audio features can also be explored to see how new features will impact system performance. In addition, a hyperparameter analysis can be done for the chosen ML models in order to determine the optimal ones.

Further research on this topic could explore different ML models for SED, especially some of the ones presented in section 1.5. Bringing the SED system complexity up to a higher level would be to try and implement DNN-based architectures in a image recognition approach, as done by many researches in the DCASE challenge [15], in the quest of achieving state-of-the-art SED performance.

When it comes to the unsupervised learning for vehicle classifications, more focus needs to be put in an analytical study of how the sounds could differ depending on categorization. These could include, but are not limited to, frequency-filtering, transient effects, and the intersection of different categories. A larger data set with more balanced representation across the different categorizations would probably be needed for this, as the clustering algorithm largely try to keep comparable number of data points in each cluster.
References


