

MA-004-526-1

Université de BIDA 1  
Faculté des Sciences  
Département d'Informatique



MASTER THESIS  
Option : Systèmes Informatiques et Réseaux

---

# AN EXPERIMENTAL STUDY OF SOUND EVENT DETECTION TECHNIQUES FOR SMART SYSTEMS

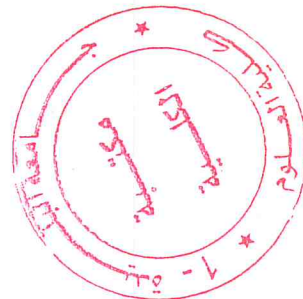
---

By  
Sylia CHIBOUB

In front of a jury composed of:

Mr. Mohamed OULD-KHAOUA  
Ms. Imene CHIKHI  
Ms. Hadjer YKHLEF  
Mr. Farid YKHLEF

President  
Examiner  
Supervisor  
Supervisor



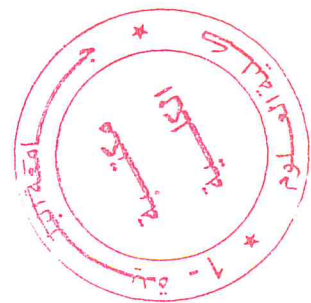
2018/2019

MA-004-526-1

## Abstract

The primary goal of this thesis is to conduct extensive experimental comparisons among Sound Event Detection systems using a combination of DCASE 2016 and DCASE 2017 datasets. We have carried out two sets of experiments. First, We have examined 3 different types of features extracted from short time frames of each audio recording (Mel frequency cepstral coefficients MFCCs, Log Mel-band Energy and a combination of MFCCs with  $\Delta$ MFCCs and  $\Delta\Delta$ MFCCs) and 4 classification paradigms while varying their parameters (Support Vector Machine, Convolutional Neural Network, Adaboost and Random Forest). We have supported our analysis and discussion with numerous statistical tests to analyze and compare the effect of the above mentioned features and classifiers on the detection performance. Our experimental findings indicate the effectiveness of the ensemble-based classifiers (Random Forest and Adaboost) along with MFCCs and the Support Vector Machine classifier along with MFCCs +  $\Delta$ MFCCs +  $\Delta\Delta$ MFCCs. More specifically, for both classifiers, we have obtained an overall class-based detection accuracy of 83% using 1 second segment based evaluation technique. Second, we have investigated the effect of the number of features on the generalization performance of the Random Forest classifier through invoking a feature selection approach, namely Minimum Redundancy—Maximum Relevance. However due to the lack of data in this field of research, this technique has caused a slight degradation in the performance of the system as the selected features were not sufficient for learning an effective model.

**Keywords:** Sound Event Detection, Feature Engineering, Feature Selection, Machine Learning.



## Résumé

L'objectif principal de ce mémoire est de réaliser une étude expérimentale pour analyser et comparer les performances des systèmes de détection d'événements sonores. Durant ce travail, nous avons utilisé une combinaison deux bases de données DCASE 2016 et DCASE 2017. Dans un premier temps, pour comparer les performances de nos systèmes, nous avons examiné trois méthodes d'extraction des caractéristiques (Mel Frequency Cepstral Coefficients MFCC, l'énergie Mel de bande et une combinaison des coefficients MFCC,  $\Delta$ MFCC, et  $\Delta\Delta$ MFCC) et quatre méthodes d'apprentissage automatique en variant leurs paramètres (machine à vecteurs de supports, les réseaux de neurones convolutif, les forêts aléatoires et Adaboost). Nous avons utilisé des méthodes statistiques pour interpréter et comparer les résultats obtenus. Cela nous a permis de démontrer l'efficacité des méthodes d'ensembles (Forêts aléatoires et Adaboost) qui utilisent les coefficients MFCC et la méthode de machine à vecteurs de supports qui utilise les coefficients MFCC+ $\Delta$ MFCC+ $\Delta\Delta$ MFCC. En effet, la performance de ces méthodes a été démontrée avec une moyenne de précision égale à 83%. Ensuite, en utilisant l'algorithme des forêts aléatoires nous avons mené une 2<sup>ème</sup> expérience où nous avons réduit le nombre des caractéristiques contenant l'ensemble des coefficients MFCC+ $\Delta$ MFCC+ $\Delta\Delta$ MFCC en invoquant une méthode de sélection des caractéristiques nommé mRMR (minimisation de redondance-maximisation de la pertinence) pour étudier son effet sur la performance de notre système. Cependant, en raison du manque de données dans ce domaine, cette technique a mené à une légère dégradation des performances du système.

**Mots clés :** Détection d'évènements sonores, extraction des caractéristiques, sélection des caractéristiques, apprentissage automatique.

## ملخص

تهدف هذه الأطروحة الى اجراء عدة تجارب لمقارنة اداء انظمة اكتشاف الاحداث الصوتية باستعمال مجموعتين من القواعد البيانية. حيث تطرقنا أولاً الى دراسة أداء أنظمتنا باستعمال 3 طرق لاستخراج الخصائص الصوتية المتواجدة بقاعدة البيانات من بينها معاملات تردد الطيفية و ترددات الطاقة في سلم ميل كما استعملنا اربع مصنفات متمثلة في ; نموذج الشبكات العصبية ,نموذج المتجهات الداعمة, نموذج ادبوست و الغابة العشوائية .من اجل تقييم اداء هذه الانظمة استندنا على عدة اختبارات احصائية حيث اكدت النتائج فعالية المصنف ادبوست والغابة العشوائية من حيث دقة الاداء و التكلفة الحسابية كما اكدت ايضا فعالية مصنف المتجهات الداعمة.

في تجربة اخرى تطرقنا لدراسة تأثير خوارزميات اختيار الخصائص الصوتية على اداء مصنف الغابة العشوائية حيث اكدت نتائج هذه الدراسة على عدم مساهمة هذه التقنية في تحسين اداء هذا المصنف.

**كلمات المفاتيح :** خوارزميات اختيار الخصائص الصوتية, انظمة اكتشاف الاحداث الصوتية,تقنيات استخراج الخصائص الصوتية,المصنفات.

## Notation and Acronyms

SED	: Sound Event Detection
DCASE	: Detection and Classification of Acoustic Scenes and Events
DCT	: Discrete Fourier Transform
FFT	: Fast Fourier Transform
MFCC	: Mel Frequency Cepstral Coefficients
DCT	: Discrete Cosine Transform
SVM	: Support Vector Machine
CNN	: Convolutional Neural Network
MRMR	: Minimum Redundancy-Maximum Relevance
TP	: True Positive
TN	: True Negative
FP	: False Positive
FN	: False Negative

## **Acknowledgments**

First and foremost, I wish to express my deepest gratitude to my supervisors Hadjer YKHLEF and Farid YKHLEF for creating a stimulating and positive working environment in the lab, without their help, encouragement, support and guidance it would be impossible for me to complete this thesis.

A Special thanks also goes to the examiners for the time they spent on carefully reading this thesis and for their constructive comments.

In addition, I wish to extend my thanks to all of my professors for sharing their invaluable knowledge.

Finally, I would like to thank Google for making the GPU computing accessible to masses for free with Colaboratory.

# Contents

<b>INTRODUCTION</b> .....	<b>1</b>
<b>PART I: FUNDAMENTALS OF SOUND EVENTS DETECTION</b> .....	<b>5</b>
CHAPTER 1:  SOUND REPRESENTATION .....	6
1.1    Introduction .....	6
1.2    Sound Acquisition .....	6
1.3    Time and frequency representation .....	7
1.4    Feature engineering .....	8
1.5    Conclusion .....	12
CHAPTER 2:  MACHINE LEARNING FOR SOUND EVENT DETECTION .....	13
2.1    Introduction .....	13
2.2    Fundamentals of classification .....	14
2.3    Common classifiers .....	15
2.4    Feature selection .....	23
2.5    Challenges .....	24
2.6    Evaluation of sound event detection models .....	24
2.7    Statistical tests .....	29
2.8    Conclusion .....	31
<b>PART II:  EXPERIMENTS</b> .....	<b>32</b>
CHAPTER 3:  EXPERIMENTAL SETUP .....	33
3.1    Introduction .....	33
3.2    Dataset .....	33
3.3    Tools .....	35
3.4    System description .....	36
3.5    Cross-validation .....	36
3.6    Feature extraction and selection .....	38
3.7    Classification approaches .....	41
3.8    Conclusion .....	43

CHAPTER 4:	EXPERIMENTAL RESULTS AND DISCUSSION.....	44
4.1	Introduction .....	44
4.2	First set of experiment .....	44
4.3	Second set of experiment .....	53
4.4	Training time .....	53
<b>CONCLUSION</b> .....		<b>54</b>
<b>REFERENCES</b> .....		<b>56</b>



## List of Figures

Figure 1.1: Annotation with full temporal information [21].	7
Figure 1.2: Hamming window.	9
Figure 1.3: The overall extraction process of MFCCs.	10
Figure 1.4: Kilo Hertz vs Mel scale [8].	11
Figure 2.1: Support vector machine [36].	16
Figure 2.2: Convolutional neural network architecture [39].	19
Figure 2.3: Convolution in CNN [40].	20
Figure 2.4: Pooling [40].	21
Figure 2.5: Four-cross validation [20].	25
Figure 2.6: Confusion matrix [25].	26
Figure 2.7: Calculation of segment-based-metrics [46].	26
Figure 3.1: Residential area: (a) time domain, (b) frequency domain.	34
Figure 3.2: Street: (a) time domain, (b) frequency domain.	34
Figure 3.3: Screenshot of online Google Colab platform.	35
Figure 3.4: The process of Sound Event Detection.	37
Figure 3.5: The process of feature engineering.	38
Figure 3.6: Log Mel band energies for a residential area recording.	39
Figure 3.7: Log Mel band energies for a street recording.	39
Figure 3.8: MFCCs of a residential areas recording.	40
Figure 3.9: $\Delta$ MFCCs of a residential area recording.	40
Figure 3.10: $\Delta\Delta$ MFCCs of a residential area recording.	41
Figure 3.11: The training and testing phase of a SED system [21].	42
Figure 4.1: Comparison of all systems against each other with the Nemenyi test. Groups of techniques that are not significantly different (at $\alpha = 0.10$ ) are connected.	45
Figure 4.2: The average accuracy scores (%) over all events for each SED system.	46
Figure 4.3: Comparison of all systems against each other with the Nemenyi test. Groups of techniques that are not significantly different (at $\alpha = 0.10$ ) are connected.	48
Figure 4.4: The average accuracy scores (%) over all events for each SED system.	49
Figure 4.5: Overall Fscore results (%) of SED systems for 1 <sup>st</sup> case study (left), the 2 <sup>nd</sup> (right).	52

## List of Tables

Table 3.1: Frequency of occurrence of each sound event within the dataset.....	34
Table 3.2: Log Mel-band energy setup. ....	38
Table 3.3: MFCC setup. ....	39
Table 3.4: MFCC+ $\Delta$ MFCC+ $\Delta\Delta$ MFCC setup.....	40
Table 3.5: Machine learning setup. ....	43
Table 4.1: Average classification accuracy (%) results based on MFCCs.....	45
Table 4.2: Average classification accuracy (%) results based on Log Mel-band energy. ....	47
Table 4.3: Average classification accuracy (%) results of SVMs using MFCC and MFCC+ $\Delta$ MFCC+ $\Delta\Delta$ MFCC .....	50
Table 4.4: Average classification accuracy (%) results of ensemble learners using MFCC and MFCC+ $\Delta$ MFCC+ $\Delta\Delta$ MFCC. ....	50
Table 4.5: Overall Fscore (%) results for each case study. ....	52

# INTRODUCTION

## 1. Context and problem statement

Sounds carry a great deal of information about our everyday environment and the events that take place in it. These sounds collected by acoustic sensors allow computers to perceive the world as human. In addition, with the raise of machine learning and deep learning fields that have given computers the ability to learn, many solutions have emerged to improve the lives of human beings. Recently, Sound Event Detection (SED) has become a very popular research area. It involves **recognizing the sound events present in an audio recording and determining their corresponding start and end times**. A sound event is a sound that we perceive as a separate, individual entity that we can name and recognize [1] such as footsteps, car passing by, speech etc.

SED has made enormous contributions in myriad of applications such as audio surveillance [2], urban sound analysis [3], multimedia event detection [4] and smart home devices[5,6].For instance smart home systems utilize SED to detect sound events [7] such as glass breaking, gunshot, etc. It is worth noting that audio surveillance can be advantageous in many scenarios, since sounds travel through obstacles, is not affected by lighting conditions, and capturing sound typically consumes less power [8].

A SED system can be categorized as **monophonic** or **polyphonic**. We consider a detection problem as **monophonic** in cases where SED systems can detect maximum one sound event at any time (the most prominent event) regardless of the actual number of sound events present in the audio recording. However in real environments recording, it is common to have multiple events occurring at the same time such as a busy street where we have car passing while people are speaking and walking. This problem is known as **polyphonic detection**. In this case the SED system is trained to detect all the **overlapping sound events** that are simultaneously present at a certain time within the audio recording.

The process of SED consists of 2 main stages: Feature extraction and Classification. First, the feature extraction stage involves dividing the audio signal into equal overlapping frames in order to perform feature extraction and obtain a feature vector per frame. The most common types of features used in the literature include Mel-frequency cepstral coefficients [9], Log Mel-band Energy [10, 11] and spectral centroid [12]. Second, classification is concerned with the development of learning algorithms that are able to learn from an exemplary set of labeled data and to generalize their behavior to new unseen instances. The resulting model, known as classifier or learner, enables us to predict the class label of an unseen sample. In the case of SED, the classifier takes the feature vector of each frame and output the event presence predictions for each sound event class. The outputs are combined for consecutive time frames in order to determine the **onset** (starting time) and **offset** (ending time) of the predicted event. The classifiers that are commonly used in the literature are the neural networks [10, 11] and Hidden Markov models [13].

The majority of SED datasets are proprietary and the only available datasets do not contain sufficient data to learn the characteristics of each sound event. To address this issue, previous efforts have used **data augmentation** techniques to improve the generalization ability of their classifiers. This technique consists of artificially expand the size of the training dataset by creating modified versions of the audio recordings [14, 15].

The main problem in the detection of real-life sound events is that features and classification approaches are not able to efficiently reflect the different characteristics of each sound event, since the features extracted from an audio recording usually composed of multiple overlapping events are not as representative as the features of an individual sound event. A possible solution would be to introduce feature selection. It aims at finding a representative subset of features. The challenge consists of reducing the number of features, while maintaining or even improving the generalization power of the system. The problem of feature selection has been demonstrated to be NP-complete [16]. To cope with this issue, many approaches have been developed such as Mutual Information Feature Selection (MIFS) [17], Joint Mutual Information (JMI) [18], and the well-known min-Redundancy Max-Relevance (mRMR) [19].

## 2. Related work

SED has been an active field of research in recent years. Various solutions have been proposed using different methodologies. S. Chu et al. [9] have investigated the effect of the number of features on the performance of SED system using 3 classification methods: KNearest Neighbor (KNN), Gaussian Mixture Models (GMM) and Support Vector Machine (SVM). They have concluded that a high number of features is not always beneficial to classification and feature selection leads to higher accuracy. Most recently, E. Cakir [10] have proposed to use log Mel-band energy features along with a Convolutional Recurrent Neural Network (CRNN). They have used multiple datasets to highlight the improvement in the performance of their proposed method. Moreover, A. Mesaros et al. [13] have used a Hidden Markov Model (HMM) to study the effect of ambient background noise on event classification performance. Table 1 provides a summary of the achieved performance of SED systems in each of the above mentioned works.

Table 1. SED related works

Ref	Dataset	Features	Classifier	Performance
[10]	TUT SED 2016	40 Log Mel-band Energies	CNN CRNN	Fscore= 26.4% Fscore= 30.3%
[13]	CLEAR 2007	48 MFCC+48 $\Delta$ MFCC+48 $\Delta\Delta$ MFCC	HMM	Accuracy=23.8%
[9]	Private Dataset	9 MFCCs+3 $\Delta$ MFCCs+Spectral Flatness + Energy range+ $\Delta$ Energy range+ Frequency roll off	SVM	Accuracy= 96.6%

## 3. Contributions

This thesis is about building a **monophonic SED system** that is able to indicate whether the event is present or not within the given set of **polyphonic recordings**. This problem has been addressed by the research community using various methodologies. Most of them have focused on finding relevant features and the suitable classifier to improve the performance of SED systems. In order to derive guidelines for practitioners and researchers, we have conducted extensive experimental comparisons among sound analysis methods based on DCASE datasets. In what follows, we summarize our main contributions.

- First, we have conducted extensive experimental comparison among sound analysis methods using a set of event sounds [2]. We have thoroughly examined 3 different types of features (Mel frequency cepstral coefficients MFCCs, Log Mel-band Energy and a combination of MFCCs with  $\Delta$ MFCCs and  $\Delta\Delta$ MFCCs). We have also examined 4 classification paradigms (Support Vector Machine, Convolutional Neural Network, Adaboost and Random Forest) while varying their parameters. We have supported our analysis and discussion with numerous statistical tests.
- Second, we have investigated the effect of the number of features on the generalization performance. Specifically, we have invoked a feature selection approach, namely mRMR [19], in order to automatically determine the optimal set of extracted features. To the extent of our knowledge, only very few attempts have considered incorporating feature selection step into the design of their systems.
- Third, in many attempts, the research community has considered the use of complex deep learning architectures along with data augmentation step in order to address the overfitting problem. However, due to the lack of dedicated computational platform, we have run our experiments using strong learners that work well in this context. Specifically, we have tested Ensemble-based inducers such as Adaboost and Random Forest.

#### 4. Thesis structure

This thesis consists of **two primary parts**. The first part covers the state-of-the-art notions that are necessary for understanding the ideas developed in this thesis. We give in **Chapter 1** an overview of acoustic features used to represent audio signals. Specifically, we present the different feature extraction techniques that are frequently used in literature. In **Chapter 2**, we review some relevant classification concepts, providing a brief description of the supervised classifiers, evaluation metrics and statistical tests invoked in this work. The second half of this thesis describes the methodology that we have followed for comparing event detection systems. We provide in **Chapter 3** detailed description of the experimental setup, including framing, windowing and parameters setting. In **Chapter 4**, we present the obtained results through performance tables and statistics-based plots. Most importantly, we have backed our discussion based on well-known statistical tests. Finally, we conclude by summarizing the contributions of this thesis, the lines of limitations and future work.

---

# PART I: FUNDAMENTALS OF SOUND EVENTS DETECTION

In this part we explain the notions that are necessary for understanding the ideas developed in this thesis. It is composed of **two chapters**. We give in **Chapter 1** an overview of acoustic features used to represent audio signals. Specifically, we describe data required for the development of sound event detection systems and highlight the importance of feature engineering to transform the signal into a suitable representation. Furthermore, we present the different feature extraction techniques that are frequently used in literature. In **Chapter 2**, we review some relevant classification concepts, providing a brief description of the supervised classifiers, feature selection techniques, evaluation metrics and statistical tests invoked in this work.

# Chapter 1: Sound Representation

## 1.1 Introduction

This chapter is organized as follows: Section 1.2 introduces the sound collection principle motivating the choice made in the recording and annotating stages. Section 1.3 deals with the transformation of the signal into the frequency domain in order to allow further sound processing. Section 1.4 highlights the importance of feature engineering to transform the signal into a suitable representation and explains the different feature extraction techniques that are frequently used in literature. The chapter is concluded in Section 1.5.

## 1.2 Sound Acquisition

The performance of the SED system is dependent on the data used to develop it. Therefore, the acquisition of data is an important step in the development process. The collected data should be as realistic as possible, recorded in conditions which are close to the target application. It should also have sufficient amount of representative examples of all sound event classes necessary to enable the machine learning models to learn parameters and generalize well [21] (discussed in more detail in Section 2.3). However, the availability of SED datasets are limited, this problem affects the performance of the system and makes the SED development process really challenging. Generally, data is collected from sound libraries or databases. It includes audio materials and **reference metadata** associated with it. The **reference metadata** is required for the supervised learning approaches because it contain the class labels [21] (discussed in more detail in section 2.3). It is often manually annotated during the data collection process and provides temporal information about the **onset** and **offset** of each event class within the audio sound as depicted in Figure 1.1.



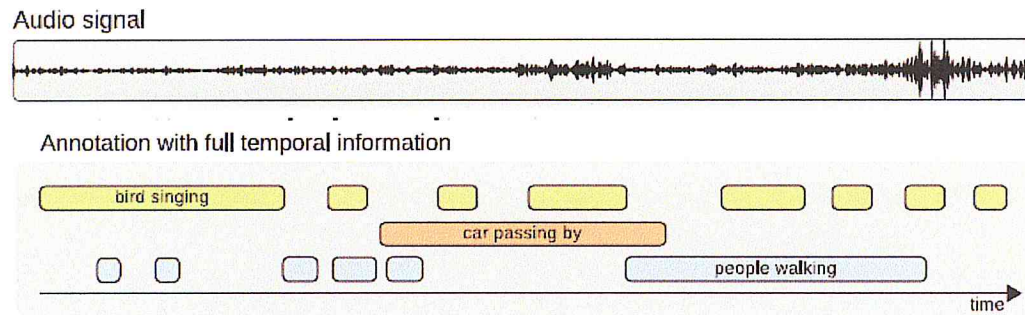


Figure 1.1: Annotation with full temporal information [21].

## 1.3 Time and frequency representation

### 1.3.1 Notation

Let  $K$  be a set of possible frequencies and  $N$  the total number of samples in a given sound signal, where  $k_i \in K$  denotes the  $i^{\text{th}}$  frequency and  $n_i \in N$  denotes the  $i^{\text{th}}$  sample of the signal. We define  $x(k_i)$  as the amount of the  $i^{\text{th}}$  frequency in the signal,  $x(n_i)$  as the amplitude of the signal at the  $i^{\text{th}}$  sample. We also denote by  $M$  the total number of samples in each frame and  $m_i$  the  $i^{\text{th}}$  sample of the frame. The resulting value  $w(m_i)$  represents the windowed value.

### 1.3.2 Fourier Transform

Signal is defined as any physical quantity that varies as a function of time. It conveys information in its patterns of variation. The manipulation of this information involves the acquisition, storage, transmission, and transformation. In order to find the different frequencies that are present in a signal we apply the Fourier transform. The goal of Fourier transform is to turn a function of time into a function of frequency. This is done by breaking up the signal into summations of sinusoidal or complex exponential components. The representation of magnitude as a function of frequency is known as the spectrum of a signal [22].

In order to make it possible for an audio signal to be stored and processed, we deal with discrete time signals these signals are obtained by sampling the original audio sequence at uniformly spaced times with a specific sampling rate. The exact form of the Fourier transform used to determine the spectrum from the discrete time signals is known as the discrete Fourier transform (DFT) and it is given by the mathematical formula:

$$x(k_i) = \sum_{n_i=0}^{N-1} x(n_i) e^{-j2\pi \frac{kn_i}{N}} \quad (1.1)$$

It is worth underscoring that the DFT algorithm has a complexity of  $O(N^2)$ , whereas, the Fast Fourier transform (FFT) implementation has a quasi-logarithmic complexity  $O(N \log_2 N)$ . For this reason, the FFT implementation is commonly used in practice.

## 1.4 Feature engineering

### 1.4.1 The process of feature engineering

The audio analysis is commonly based on acoustic features extracted from audio signal to provide a numerical representation of the signal that is relevant for machine learning (discussed in more detail in Chapter 2), characterizing the signal with values which have connection to its physical properties: **signal energy, its distribution in frequency and change over time** [21].

The necessary property of the acoustic features is low variability among features extracted from examples assigned to the same class, and at the same time high variability allowing distinction between features extracted from example assigned to different classes [23]. This property will help making the learning problem easier.

We can classify acoustic features into 3 main categories: temporal features, spectral features and prosodic features. Before performing the extraction of the above mentioned features we need to perform Framing and windowing.

### 1.4.2 Framing

The properties of audio signals change rapidly over time (non-stationary) which will make the analysis process hard. In order to solve this issue we perform a short-time processing approach where the analysis is done periodically in short-time **frames** shifted with a fixed timestamp. This shift needs to ensure that the consecutive frames are overlapping at least 50% [21] in order to capture the signal in a quasi-stationary state.

### 1.4.3 Windowing

Windowing is used in order to reduce the effect of spectral leakage (sudden changes in frequency at the **frame** boundaries) which will cause an insignificant peak in frequency in the spectrum. This is done by multiplying each **frame** with a window function generally **hamming window** function.

The **hamming window** function has a sinusoidal shape .It smooths discontinuities at the beginning and at the end of the sampled signal. It is defined as:

$$w(m_i) = \begin{cases} 0.54 - 0.46\cos(2\pi m_i/M) & , 0 \leq m_i \leq M \\ 0 & , otherwise \end{cases} \quad (1.2)$$

The following Figure shows the shape of this function.

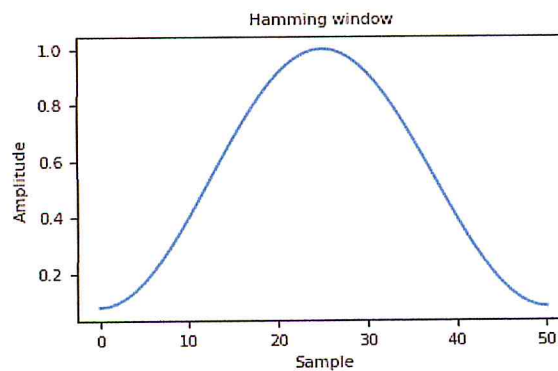


Figure 1.2: Hamming window.

### 1.4.4 Temporal features

Temporal features are represented as amplitude fluctuation with time (waveform signal) .They are extracted directly from the audio signals [24].

### 1.4.5 Spectral features

Audio signals rely on spectral / cepstral features these features characterize the short-time spectrum and allow us to have the spectral energy distribution which is an effective feature for event detection. They are based on a spectral representation of the audio signal. This representation is computed based on two main processes: The Fourier transform and logarithm which will grant the identification of the basis frequency elements of an audio signal.

Log Mel-band energy and Mel-frequency cepstral coefficients (MFCCs) are commonly used in the literature. This is due to the fact that they mimic the human auditory perception of sounds that focuses only on magnitudes of frequency components. The following subsections provide brief description of these techniques.

### A Mel Frequency Cepstral Coefficients (MFCC)

It is a very common and efficient technique for signal processing. It models the spectral energy distribution and provides a compact representation of the spectrum taking into consideration the nonlinear human perception of frequencies as described by the Mel scale (the human system perceives and interprets frequency that are in 0-1000Hz range linearly and above 1000Hz the perception becomes logarithmic) [8, 25]. The overall extraction process of MFCCs is shown in Figure 1.3.

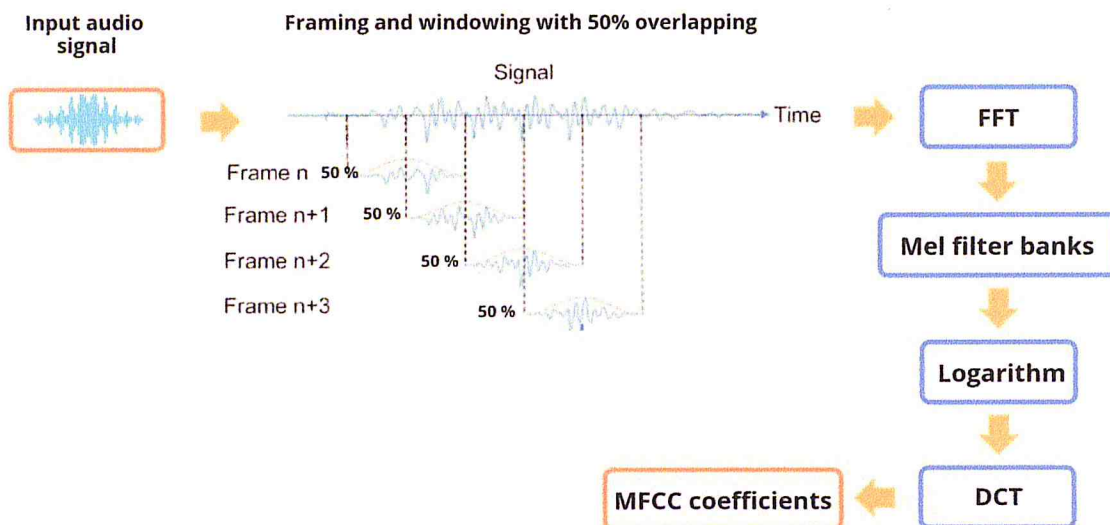


Figure 1.3: The overall extraction process of MFCCs.

**Mel filter banks:** A filter bank is an array of band-pass filters used to separate the input audio signal into multiple frequency bands. This is done by passing frequencies within a certain range and attenuating frequencies outside that range. We use filter banks in order to mimic the two main characteristics of the human auditory system: the Mel scale and critical bands. The concept of critical bands is used to quantify the ability of the human ear to distinguish between individual frequency tones. The Mel scale as shown in Figure 1.4 indicates how to space the filter banks

which will produce a bunch of frequency bands, in each frequency band we sum the energy and take their logarithm to obtain an estimate of how much energy exist in various frequency regions. We convert between Hertz and Mel using the following equations.

$$\begin{aligned} mel &= 2595 \log_{10} \left( 1 + \frac{frequency}{700} \right) \\ frequency &= 700 (10^{mel/2595} - 1) \end{aligned} \quad (1.3)$$

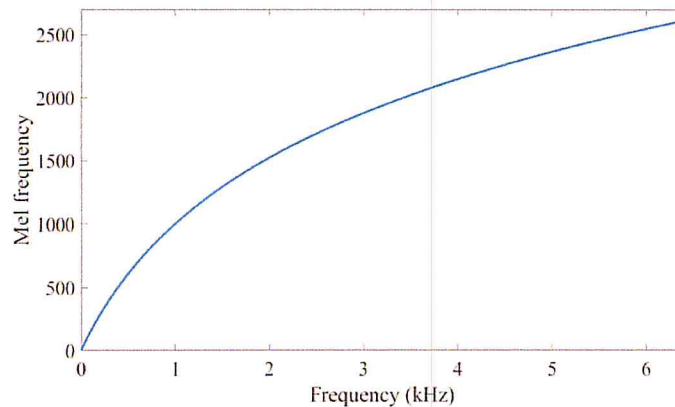


Figure 1.4: Kilo Hertz vs Mel scale [8].

**Discrete cosine transform (DCT)** The goal of taking the DCT of the log filter bank energies is to extract the MFCCs. The DCT has an important property called energy compaction which will allow the distribution of most signal energy in the lower-order coefficients.

## B Log Mel band energy features

The log Mel-band energy features represent the energy distribution within the frame. The steps involved for their extraction are similar to the MFCCs steps but without computing the DCT.

### 1.4.6 Prosodic features

Prosodic features indicate information with semantic meaning in the content of human listeners. They are organized according to semantically meaningful aspects of sounds including pitch (fundamental frequency), loudness and rhythm (the duration of pitch) [24].

### 1.4.7 Other approaches

**Delta and delta-deltas MFCCs** were proposed to add dynamic information to the MFCCs. The delta features are computed by taking the first derivative of the MFCCs. They provide information about the trajectories of the MFCCs over time (velocity). On the other hand, the delta-deltas features provide information about the acceleration of the MFCCs over time. They are calculated by taking the first derivative of the delta features.

## 1.5 Conclusion

Through this chapter, we reviewed three important stages required to prepare the audio signals for machine learning algorithm. The first one deals with the importance of sound recording and annotating during the sound acquisition process as it influences directly the performance of the SED system. The second one deals with the frequency representation of the signal to allow further manipulation and processing of sounds. The last one is the most crucial stage, it concerns feature engineering where we presented several types of features used in literature.

# Chapter 2: Machine Learning for Sound Event Detection

## 2.1 Introduction

In the previous chapter we have discussed the notions of sound acquisition and representation required to prepare the audio signals for **machine learning** and **deep learning** approaches. **Machine learning** is the field of study that gives computers the ability to learn without been explicitly programed [26]. In the other hand, **deep learning** is a subset of **machine learning**. It consists of a set of models known as neural networks that unlike the machine learning models are hungry which means they need a large amount of data to perform well. There are many machine learning approaches that fall into three major categories supervised, unsupervised and semi supervised learning. We largely focus on the **supervised learning** approach as it is the most frequently used for the analysis of sound events. It consists of building models to learn a mapping between the extracted features and class labels for sound classes, where the labels are predefined in advance and determined from the **reference annotations**.

The rest of this Chapter is structured as follows. We present in Sections 2.2 and 2.3 common supervised classifiers used in our study, providing a short introduction to the relevant concepts of classification. Moreover, in Section 2.4 we explain the concept of feature selection then we discuss in Section 2.5 some challenges related to SED research. Furthermore, in Sections 2.6 and 2.7 we talk about model evaluation techniques and statistical tests. Finally in Section 2.8 we conclude this chapter by summarizing the main concepts that we have discussed.

## 2.2 Fundamentals of classification

Classification is considered as the most common task in **machine learning** [27]. It is concerned with the problem of attributing class labels to unseen objects. An object (also called pattern, sample and instance) is characterized by a feature vector  $x \in X$  and by its class labels  $y \in Y = \{c_1, c_2, \dots, c_m\}$ . We can formally express a classification problem as a mapping from the feature space  $X$  to the space of class labels  $Y$  [28]. In **supervised learning**, the role of any given classification algorithm is to learn predictive model from a set of  $m$  data samples  $\Gamma = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$  which have been labeled beforehand, where  $x_i \in X$  and  $y_i \in Y$ . A classification model (also called classifier, learner and hypothesis) is the estimated mapping function  $f$  which takes in a feature vector  $x \in X$ , some parameters  $\tau$  and produces an output  $\hat{y}$ .

$$\hat{y} = f(x, \tau) \quad (2.1)$$

We can distinguish between 3 types of outputs [29]:

- **Class labels** :  $\hat{y} \in Y$
- **Probability distribution**: The classifier returns a probability vector over the  $m$  class labels  $\mu = [\mu_1, \mu_2, \dots, \mu_k]^T \in [0,1]^k$ .
- **Oracle output**: It is defined as a Boolean vector  $Z = [z_1, \dots, z_m]^T$ , where  $m$  is the size of the training set  $\Gamma$ , with  $z_i = 1$  if the learner correctly classifies instance  $i$ , and 0 otherwise.

Usually a classifier is seen as a two-step algorithm: **training phase** and **testing phase**. The first phase concerns the task of learning a hypothesis from the training data. Learning is the process of optimizing for an objective to increase the generalization ability of the model (the ability to perform well on unseen data). This objective is known as a loss function that calculates the difference between the actual and the predicted outputs. The model is updated in order to decrease this loss through various optimization techniques such as gradient descent until it reaches convergence (more details can be found in Section 2.3.4). In the second phase, the produced model is used to predict the class labels of unseen objects drawn from a testing set. Numerous learning models have been introduced by the machine learning community. For the remainder of this thesis, we will focus on learning Support Vector Machines, Random Forests, Adaboost and Convolutional Neural Network models. The first three models produce oracle



outputs while the last model produces probability outputs. These models are just a few examples of supervised learning algorithms. They are all based on different paradigms Section 2.3 provides an extended explanation of these approaches.

The aim of any learning algorithm is to find the model parameters  $\tau$  (Equation 2.1) that give the best predictive performance. However, there is no single learning algorithm that induces the most accurate classifier. The natural approach is to try many learners and select the one with the best performance on a separate sample set. For this purpose, we need to measure the performance of a classifier. This is done using multiple evaluation metrics (more details can be found in Section 2.6) with the most common being the error rate i.e. the ratio between the number of misclassified samples to the total number of samples. It is assumed that a classifier that accurately predicts the training samples is expected to perform well on testing examples. However, a model that fits the training data perfectly can have worse performance than a simple model with higher training error. This paradox is known as **overfitting** [28]. Given multiple learning algorithms, model evaluation aims at identifying which algorithm produces the most accurate classifiers. This concern is one among the fundamental issues in machine learning. In order to address it, Dietterich [30], Demsar [31], Garcia et al. [32, 33], and Japkowicz et al. [34] introduced several statistical tests such as Friedman and Nemenyi for performance comparison. In Section 2.7 we briefly review the statistical tests that we have invoked in our experiments [31, 34].

## 2.3 Common classifiers

### 2.3.1 Support Vector Machines

Let's consider a binary classification problem formalized as  $\Gamma = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$  where  $x_i \in X$  and  $y_i \in Y = \{-1, 1\}$ . Support vector Machines classifiers (SVMs) find an optimal separating hyperplane that maximizes the margin between the two classes, where the margin is defined as the distance of the closest point in each class to the separating hyperplane [35] as illustrated in Figure 2.1.

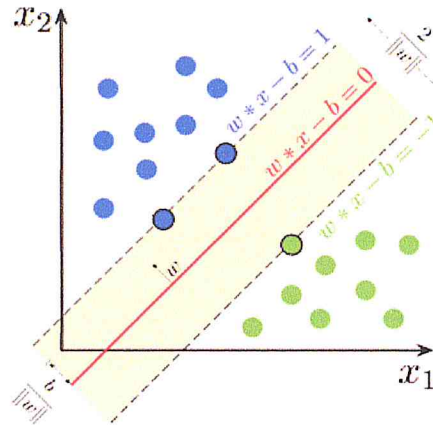


Figure 2.1: Support vector machine [36].

If the learning problem is linearly separable then constructing the optimal hyperplane implies solving an optimization problem where the optimization criteria are the width of the margin between the classes as defined by the following system:

$$\min \frac{1}{2} \|w\|^2 \quad (2.2)$$

$$y_n(w \cdot x_n + b) \geq 1, \quad n = 1, \dots, m$$

resulting in an optimal hyperplane decision function (or classification rule) represented as a linear combination of the training samples. Where  $f(x)$  determines the class of the test sample  $x$ .

$$f(x) = \sum_{n=1}^m y_n \alpha_n(x, x_n) + b \quad (2.3)$$

As we can observe from the mathematical formula, constructing this optimal hyperplane is equivalent to finding the nonzero  $\alpha_n$ . Any data point  $x_n$  corresponding to nonzero  $\alpha_n$  is termed “support vectors”. **Support vectors are the training patterns closest to the separating hyperplane.**

Generally, the samples of different classes cannot be linearly separable. In order to solve this kind of classification problems we use a concept known as the **kernel trick** to enable the SVMs to handle nonlinear separating hyperplanes defined by the following equation:

$$f(x) = \sum_{n=1}^m y_n \alpha_n k(x, x_n) + b \quad (2.4)$$

where  $k(x, x_n)$  is the kernel function . Popular kernel functions include:

- **Gaussian (RBF) kernel** with the standard deviation  $\sigma$  of the Gaussian function:

$$k(x, x_n) = \exp\left(\frac{-\|x, x_n\|^2}{2\sigma^2}\right) \quad (2.5)$$

- **Polynomial kernel function** with the polynomial order  $p$ :

$$k(x, x_n) = ((x \cdot x_n + 1))^p \quad (2.6)$$

### 2.3.2 Random Forest

Random Forest classifier is an **ensemble algorithm** that combines a set of decision trees and aggregates their votes for each label to decide the final class of the test sample. A decision tree is a hierarchical model for supervised learning. It is composed of a root node connected by successive branches to other internal nodes that are similarly connected until we reach the leaf nodes [37]. Let us consider a binary classification problem formalized as  $\Gamma = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$  where  $x_i \in X$  and  $y_i \in Y = \{-1, 1\}$ . The classification of training samples begins at the root node which takes  $\Gamma$  as an input and asks for the value of a particular feature of the samples that can split  $\Gamma$  into different possible subsets. Different links from the root node correspond to the different possible subsets of  $\Gamma$  and based on the answer we follow the appropriate link to a descendent node. The links must be distinct i.e. one and only one link will be followed. The next step is to ask for the value of a particular feature that can split the chosen subset of  $\Gamma$  into other different subsets. We continue this way until we reach a leaf node, which has no further questions. Each leaf node bears a class label and the test samples are assigned to the class of the leaf node reached [27]. The way  $\Gamma$  is split is based on two important concepts that form our objective function that we are optimizing for to improve the performance of our model. These concepts are: entropy and information gain.

The entropy measures the impurity of the node to find the best value of the feature  $x_i$  that allows splitting  $\Gamma$  into different subsets. These subsets should minimize their entropies. This process is repeated recursively until we reach a leaf. Impurity means that each subset of features represents one type of class, in this case the *entropy* = 0 and the information gain reaches its

maximum value of 1. However, some percentage of impurity is tolerated in order to stop further division to reduce the training time. The entropy  $H$  and information gain are computed using the following mathematical formulas.

$$H(X) = \sum_{i=1}^m -x_i \log_2(x_i) \quad (2.7)$$

$$\text{Information gain} = H(\text{parent}) - (\text{weighted average}).H(\text{children})$$

### 2.3.3 ADABOOST

ADABOOST, short for “Adaptative boosting”, was initially proposed by Freund and Schapire [38] as an **ensemble method** for improving the performance of a **weak learner** i.e. a classifier that performs better than random guessing such as decision trees. Adaboost is a sequential algorithm in which each new inducer is built by taking into account the performance of the previously trained ensemble members. At stage  $t$ , every training sample  $x_i$  receives a weight  $w_i^{(t)}$  that indicates its probability of being selected to train a new **weak classifier**. The first classifier is built by setting these weights to  $1/m$ , where  $m$  denotes the number of training samples, i.e. all samples initially have the same importance. If a training sample is correctly classified, then its chance of being reused in the next stage is decreased. Conversely, if a sample is misclassified, then its chance of being reselected is increased. In this way, the subsequent classifiers focus on examples that are difficult to classify. Adaboost assigns to the new trained classifier a weighting coefficient  $\alpha_i$ : accurate members receive higher weights. This process continues until the desired number of base learners or the overall accuracy has been reached. The final classification decision of a test sample is based on the weighted linear combination of these **weak classifiers**.

### 2.3.4 Convolutional Neural Networks

Convolutional neural networks (CNNs) are biologically-inspired from the animal’s visual cortex that is composed of a collection of cells (also called neurons) connected to each other and organized in hierarchical layers. Each set of neurons responds very specifically to specific patterns (also called the cell’s receptive field). In general, all CNNs architectures consist of a set of layers. The first one is called the **convolutional layer** followed by the **activation and pooling**

layer, these three layers form a hidden layer. The last layer is the **dense layer** (known as a fully connected layer). The architecture of the CNN is depicted in Figure 2.2

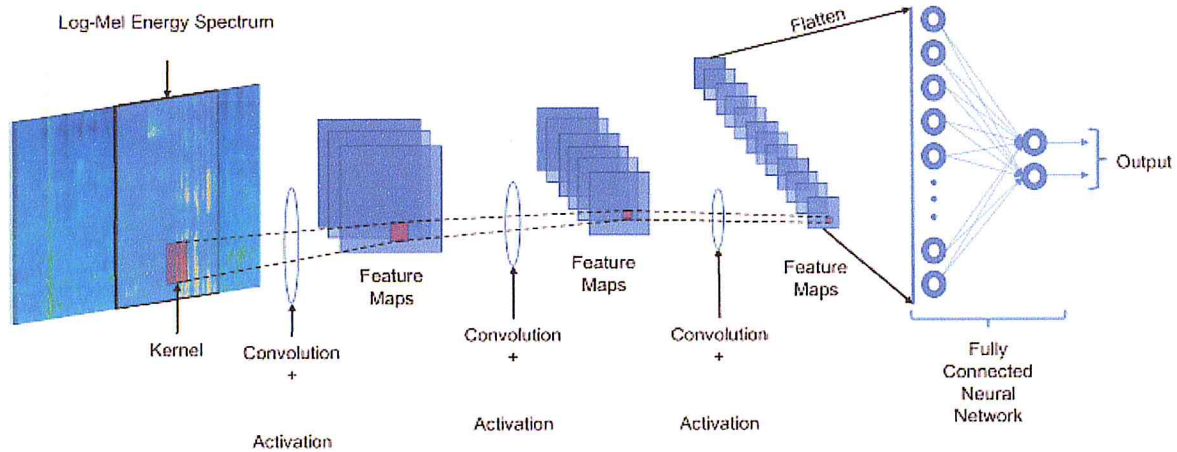


Figure 2.2: Convolutional neural network architecture [39].

### Notation

We denote  $x^i$  as a single feature value within the feature map. We also define  $x_i$  as a single probability value within the normalized feature map  $W$ . This normalization is done by applying the **softmax function** to each  $x^i$  of the feature map. We represent the resulted loss of the cross entropy loss function over the feature map  $W$  as  $J(W)$  and we denote  $\eta \times \frac{\partial J(W)}{\partial W}$  as the partial derivative of the loss function (also called the gradient), multiplied by  $\eta$  where  $\eta$  represents the value of the learning rate. We also denote the oracle output of the classifier as  $y_i$  which is equal to 1 to represent the presence of the class and 0 otherwise.

### The Convolutional layer

It consists of a set of filters (also called feature matrix, kernel, weight, and feature map). The filter moves over each portion of the input with a certain stride value and a mathematical operation called **convolution** is computed. It consists of a dot product between the matrix that represents the portion  $p$  of the input and the feature map matrix. This process is repeated until the entire input is traversed. The resulting matrix is known as the **convolved feature matrix** (or simply feature map), it consists of a set of feature values that matches the input exactly and a set

of zeros that represents the irrelevant parts where there were no match. The objective of convolution is to enable the CNN model to learn only the most relevant features. An illustration of this operation is depicted in Figure 2.3.

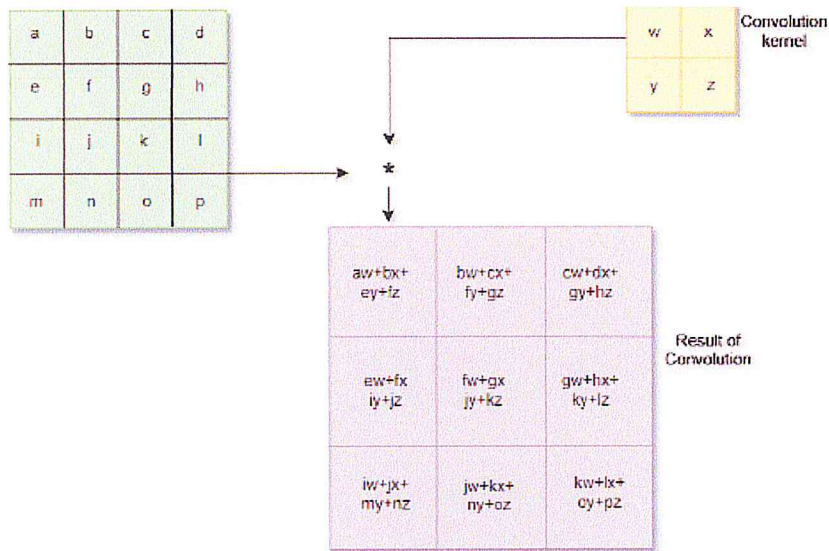


Figure 2.3: Convolution in CNN [40].

### Activation layer

The **activation layer** applies an activation function over the entire convolved feature matrix to enable the model to learn nonlinear mapping function  $f$  to classify nonlinear data. The most commonly used activation function is known as **Rectified linear Units activation function (Relu)**. It is defined as follows.

$$R(x^i) = \begin{cases} 0 & , x^i < 0 \\ x^i & , otherwise \end{cases} \quad (2.8)$$

### Pooling layer

It is employed to reduce the dimensions of the convolved feature matrix by extracting only the most relevant and dominant features. Furthermore, it is useful to decrease the computational power required to process the data. The most common type of pooling operation is known as **maximum pooling** which returns the maximum value from each portion of the convolved feature matrix. To perform **pooling** a window is used over each portion of the convolved feature matrix

and is moved with a specific window stride value. An illustration of this operation is depicted in Figure 2.4.

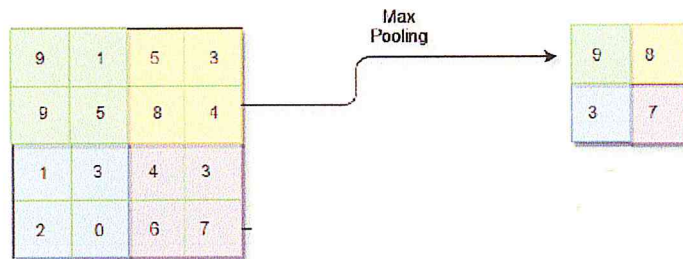


Figure 2.4: Pooling [40].

### Dense layer

The **convolutional** and **maxpooling** layers output a feature map. The **dense** layer which is the last layer in the CNN flatten this matrix and normalize it using a **softmax function** into a vector of probability values  $\in [0,1]$  whose total sums up to 1. This vector is then used to train the CNN in order to perform classification. The training is done by finding the weights (feature map) that minimize the loss between the actual and the predicted output in the training set using the cross entropy loss function. This process is known as gradient descent (also called backpropagation) that will be repeated over a series of epochs (iterations) to update the feature map until convergence. After the learning step, the dense layer outputs a probability distribution for each predefined class. The mathematical formula of the **softmax function** is shown below:

$$x_i = \text{softmax}(x^{(i)}) = \frac{e^{x^{(i)}}}{\sum_{k=0}^n e^{x_k^{(i)}}} \quad (2.9)$$

### Gradient descent

Gradient descent is the process of finding the weights that minimize the loss function over a series of epochs during the training phase until convergence. This optimization algorithm is said to converge when, as the iterations over a series of epochs proceed, the gradient get closer to 0 and remains stable.

In order to measure the performance of a classification model that produces oracle outputs we use the binary cross entropy loss function. The gradient descent algorithm and the binary cross entropy mathematical formulas are shown below.

$$j(W) = \frac{1}{n} \sum_{i=1}^n (y_i \log(x_i) + (1 - y_i) \log(1 - x_i)) \quad (2.10)$$

---

**Algorithm 1** Gradient Descent
 

---

1. Initialize weights randomly

2. Loop until convergence :

2.1. Compute gradient :

$$\frac{\partial j(W)}{\partial W} \quad (1)$$

2.2 Update weights :

$$W \leftarrow W - \eta \frac{\partial j(W)}{\partial W} \quad (2)$$

3. Return weights

---

### Hyper parameters: Learning rate

Gradient descent is based on an important hyper-parameter called the learning rate  $\eta$ . It is assumed that for a sufficient small  $\eta$  the value of  $\frac{\partial j(W)}{\partial W}$  should decrease on every iteration until convergence. If  $\eta$  is too small, gradient descent can be slow to converge, whereas, if  $\eta$  is too large, the gradient descent may not decrease at every iteration and consequently it may not converge.

### Hyper parameters: Dropout

One of the most fundamental problems in machine learning is overfitting. To address this problem, we use regularization techniques to improve the generalization ability of our model. The most popular regularization technique in neural networks is called dropout. Dropout forces the network to take several paths in the process of learning to increase its generalization ability. This is done by randomly dropping a set of neurons in each epoch.

### Hyper parameters: Batch size

In order to train our model we need to specify the batch size. It is the number of samples in the input data that will be passed through the network at one time. The larger the batch size, the



faster our model will be trained. However, the quality of our model may degrade as we set our batches larger.

## 2.4 Feature selection

Feature selection aims at finding a compact and effective subset of features. The challenge consists of reducing the number of features while maintaining or even improving the generalization power of the system. Given a set of  $n$  features, one straightforward and naïve strategy consists of searching for a subset that best optimizes a criterion indicative of the generalization accuracy. This task involves evaluating  $2^n - 2$  subsets (excluding the empty set and the entire set), which becomes intractable for moderate and large number of features. This problem has been demonstrated to be NP-complete [16]. To cope with the computational burden, numerous approaches have been developed in the literature such as Mutual Information Feature Selection (MIFS) [17], Interaction Gain Feature Selection [41], Conditional MIFS (CMIFS) [42], Joint Mutual Information (JMI) [18], and the well-known min-Redundancy Max-Relevance (mRMR) [19].

### 2.4.1 Forward Selection and Minimum Redundancy—Maximum Relevance criterion (MRMR)

The minimum redundancy-maximum relevance (mRMR) criterion has been proposed in [19, 43]. Given a set  $X_S$  of selected variables (features), the criterion consists of updating  $X_S$  with the variable  $X_i \in X_{-S}$  that maximizes  $u_i - z_i$ , where  $u_i$  is a relevance term and  $z_i$  is a redundancy term. More precisely,  $u_i$  is the relevance of  $X_i$  to the output  $Y$  alone and  $z_i$  is the average redundancy of  $X_i$  to the selected variable  $X_j \in X_S$ .

$$u_i = I(X_i; Y) \quad (2.11)$$

$$z_i = \frac{1}{d} \sum_{X_j \in X_S} I(X_i; Y) \quad (2.12)$$

where  $I(X_i; Y)$  is the Mutual Information. It represents the amount of shared information between 2 random variables. We denote by  $d$  the size of  $X_S$  during each iteration of forward search. At each step  $d$ , this method selects the feature which has the best trade-off between relevance and redundancy. This selection criterion is fast and efficient. [44].

## 2.5 Challenges

It can be claimed that the research progress on SED has been stagnant until the recent years. One of the reasons is that there are several challenges for a robust SED system that can operate in real-life conditions [8]. Some of the challenges are listed below.

### 2.5.1 Intra-class Variability

Sound events for SED systems are often defined broadly such as phone ringing, doorbell etc., this latter presents a challenge for SED methods in the form of intra-class variability. For instance, doorbell class can be used to represent all types of doorbells. The acoustic features can vary significantly among the examples of this class. Therefore, in order to claim that a SED system can robustly detect doorbells, it should be able to detect or extract the acoustic features that are found in common among a **variety** of doorbell sounds [8].

### 2.5.2 Overlapping Events

The earlier research on SED has been focused on the detection of individual sound events recorded in isolated environments [45]. However, in the real world, sound events often occur simultaneously. For instance, a recording from a children's park may include children shouting, adults speaking, footsteps and birds singing; all happening at the same time. Therefore, the SED system should be able to distinguish the acoustic characteristics of each individual sound event among this mixture [8].

## 2.6 Evaluation of sound event detection models

### 2.6.1 Cross-validation

A common approach for evaluating SED is to learn a hypothesis from a training set and to measure its generalization error on a test set. It is worth underscoring that the training and testing data should not overlap, otherwise the estimated performance can be overoptimistic. This approach requires a large amount of data in order to obtain a reliable estimate of the generalization error, which is rare in most situations. A possible alternative consists of invoking a resampling technique such as k-fold cross validation. Figure 2.5 illustrates the way the data is divided using 4-cross validation technique.

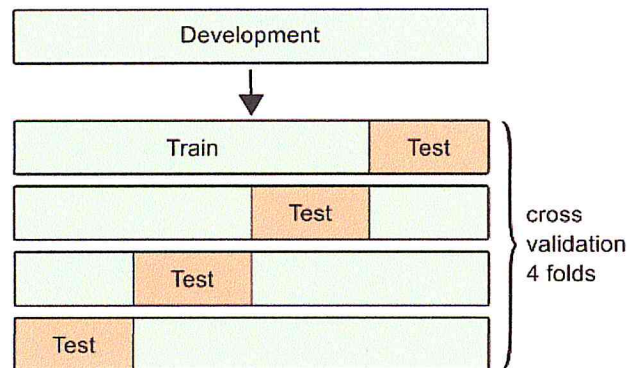


Figure 2.5: Four-cross validation [20].

### 2.6.2 Performance evaluation approaches

Evaluation of SED system is done by comparing the system output with **the reference annotations** available for the test data. Metrics used in detection of sound events include: accuracy, precision, recall, F-score, acoustic event error rate (AEER). However, there is no consensus over which metric is universally good for measuring performance of sound event detection as they each reflect a different perspective on the ability of the system [21]. The way these metrics are computed is based on the evaluation approach used to evaluate the system. We can distinguish two evaluation approaches: **segment-based evaluation** and **event based evaluation**. For each we need to define what constitutes correct detection and what type of errors the system produces. We refer to these as **intermediate statistics** [46].

#### A Segment-based evaluation

The evaluation of SED system performance using the segment-based approach shows how good the system is at correctly detecting the temporal regions where the sound event is present. This is done by comparing the system output and reference in **one second long segment** to determine the active/inactive state for each event. The **intermediate statistics** can be summarized in a **confusion matrix** as shown in Figure 2.6 and are defined as follows [46].

- **True positive (TP)** the reference and system output both indicate an event to be active in that segment.

- **False positive (FP)** the reference indicates an event to be inactive in that segment, but the system output indicates it as active.
- **False negative (FN)** the reference indicates an event to be active in that segment, but the system output indicates it as inactive.
- **True negative (TN)** the reference and system output both indicate an event to be inactive.

		classifier outcome		total
		P	n	
actual value	p'	True positive	False negative	p'
	n'	False positive	True negative	N'
total		P	N	

Figure 2.6: Confusion matrix [25].

Segment-based Accuracy, Precision, Recall and F-score are calculated based on 1 second long segment-based intermediate statistics using instance based averaging or class-based averaging. The calculation is illustrated in Figure 2.7.

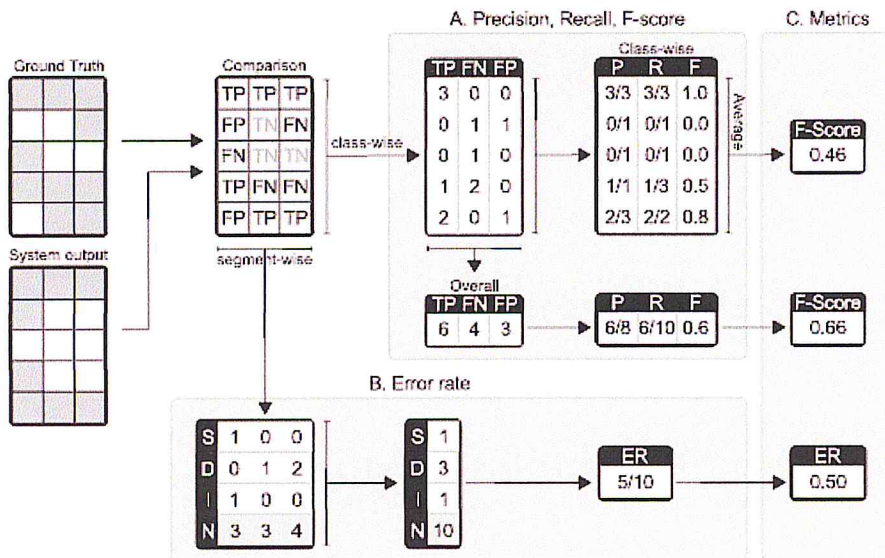


Figure 2.7: Calculation of segment-based-metrics [46].

In **instance based averaging** (micro-averaging) intermediate statistics are accumulated over the entire data and the overall performance is calculated, the values of the resulting metrics are affected by the performance of the most common event classes. In **class-based averaging** (macro-averaging), intermediate statistics are accumulated separately for each event class and are used to calculate class-wise metrics. Overall performance is then calculated as the average of class-wise performance, resulting in values that emphasize the system behavior on the smaller event classes [46].

## B Event based evaluation

The evaluation of SED system performance using the event-based approach shows how good the system is at correctly detecting the event instances with their corresponding onset and offset. This is done by comparing event instances one to one but since the onset-offset of the system output may not match the reference exactly, a time misalignment threshold is allowed to be applied to the onset and offset. In their study, Giannoulis et al. [47] have set this threshold to  $\pm 100$  ms allowing an event to be considered as correctly detected if its onset-offset were within  $\pm 100$  ms of the corresponding reference event. The event based intermediate statistics are defined as follows [46].

- **True positive (TP)** an event in the system output that has a temporal position overlapping with the temporal position of an event with the same label in the reference. A collar is usually allowed for the onset and offset, or a tolerance with respect to the reference event duration.
- **False positive (FP)** an event in the system output that has no correspondence to an event with same label in the reference within the allowed tolerance.
- **False negative (FN)** an event in the reference that has no correspondence to an event with same label in the system output within the allowed tolerance.

### 2.6.3 Performance metrics

#### A Accuracy

Accuracy measures how often the classifier makes correct decisions; it is defined as the ratio of correct system outputs to total number of outputs [46]. Formally it is computed as:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.13)$$

## B Precision, Recall and F-score

**Precision** is defined as the ratio of the correctly detected events among all the detection made by the system [8]. It reaches its highest value of 1 when no false alarms occur. It is calculated as:

$$P = \frac{TP}{TP + FP} \quad (2.14)$$

**Recall** is an indication for missed detections; it reaches its highest value of 1 when all the active events are detected and decreases when the active events are missed [25]. It is calculated as:

$$R = \frac{TP}{TP + FN} \quad (2.15)$$

**F-score** is defined in terms of precision and recall. It is given by:

$$F = \frac{2 \times P \times R}{P + R} \quad (2.16)$$

It is worth noting that adapting these metrics to the segment-based evaluation is based on the definitions of the intermediate statistics depicted in Figure 2.6. More formally:

$$F = \frac{2 \times TP}{2 \times TP + FP + FN} \quad (2.17)$$

## C Acoustic event error rate (AEER)

The acoustic event error rate has been specifically defined for SED system [48]. Three types of errors were introduced [25].

- **Deletion (D)** An event was not detected at all (missed event).
- **Insertion (I)** An event was detected although it didn't really occur (extra event).
- **Substitution (S)** An event was detected but not the correct one.

To calculate segment-based error rate, errors are counted segment by segment. We obtain the following formulas:

$$\begin{aligned}
S(k) &= \min(FN(k), FP(k)) \\
D(k) &= \max(0, FN(k) - FP(k)) \\
I(k) &= \max(0, FP(k), FN(k))
\end{aligned} \tag{2.18}$$

$S(k)$  denotes the number of substitutions errors in the segment,  $D(k)$  is the number of deletions in the segment  $k$  and  $I(k)$  designates the number of insertions in the segment  $k$ .

Total error rate is calculated by integrating segment-wise counts over the total number of segments  $K$ . with  $N(k)$  being the number of sound events marked as active in the reference in segment  $k$  [46].

$$ER = \frac{\sum_{k=1}^K S(k) + \sum_{k=1}^K D(k) + \sum_{k=1}^K I(k)}{\sum_{k=1}^K N(k)} \tag{2.19}$$

## 2.7 Statistical tests

### 2.7.1 Friedman test

The Friedman test is useful for comparing several algorithms over multiple sound events. It first ranks the techniques for each sound event separately according to the generalization accuracy in descending order. The best performing technique gets the rank 1, the second best gets rank 2...etc. In case of ties, average ranks are assigned. Let  $r_i^j$  be the rank attributed to the  $j^{th}$  system on the  $i^{th}$  event; and let  $R_j = \frac{1}{N} \sum_{i=1}^N r_i^j$  denote the *average rank* of system  $j \in \{1, \dots, t\}$  over  $N$  events. Under the null hypothesis, it is assumed that all techniques are equivalent; hence, their average ranks should be equal. The statistic  $\chi_F^2$  follows chi-squared distribution with  $t - 1$  degrees of freedom for sufficiently large  $N$  and  $t$  (usually  $N > 10$  and  $t > 5$ ). In their study, Iman and Davenport reported that  $\chi_F^2$  is conservative and derived a new statistic  $F_F$  which is distributed according to the F-distribution with  $t - 1$  and  $(t - 1)(N - 1)$  degrees of freedom.

$$\chi_F^2 = \frac{12N}{t(t+1)} \left[ \sum_{j=1}^t R_j^2 - \frac{t(t+1)^2}{4} \right], \quad F_F = \frac{(N-1)\chi_F^2}{N(t-1) - \chi_F^2} \tag{2.20}$$

This test provides only an assessment whether the observed differences in the performances are statistically significant. In order to have a zoomed-in view of what these differences correspond to precisely i.e. identify pairs of techniques with significant different performances, usually we perform a **post hoc** test when Friedman test rejects the null hypothesis. Nemenyi is an example of **post hoc** tests that are widely used in conjunction with Friedman test.

### 2.7.2 Nemenyi test

This test is invoked when all techniques are compared with each other. The performance of two methods is significantly different if their corresponding average ranks differ by at least the critical difference

$$CD = q_\alpha \sqrt{\frac{t(t+1)}{6N}} \quad (2.21)$$

where the critical value  $q_\alpha$  is defined based on the Studentized range statistic divided by  $\sqrt{2}$ .

### 2.7.3 Wilcoxon signed-ranks test

Wilcoxon signed-ranks test is a non-parametric test and is considered the best strategy to compare two algorithms over multiple domains. The formulation of this test is the following. We designate by  $d_i$  the difference between the performance scores of two techniques on  $N$  datasets.  $i \in \{1, \dots, N\}$ . We first rank these differences according to their absolute values; in case of ties average ranks are attributed. Then, we compute the sum of ranks for the positive and the negative differences, which are denoted as  $R^+$  and  $R^-$ , respectively. Their formal definitions are given by:

$$R^+ = \sum_{d_i > 0} \text{rank}(d_i) + \frac{1}{2} \sum_{d_i = 0} \text{rank}(d_i) \quad R^- = \sum_{d_i < 0} \text{rank}(d_i) + \frac{1}{2} \sum_{d_i = 0} \text{rank}(d_i). \quad (2.22)$$

Notice that the ranks of  $d_i = 0$  are split evenly between  $R^+$  and  $R^-$ . Finally the statistics  $T_w$  is computed as  $T_w = \min(R^+, R^-)$ . For small  $N$ , the critical value for  $T_w$  can be found in any textbook on general statistics [34], whereas for larger  $N$ , the statistics:

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \quad (2.23)$$



follows the normal distribution with 1 mean and 0 variance. For instance the hypothesis which states that two approaches perform equally is rejected if  $z \leq -1.96$  at a 5% significance level.

## 2.8 Conclusion

Through this chapter, we have reviewed the important concepts of classification. First, we have presented the main classifiers used in our work. Then we have discussed the challenges related to SED research. Furthermore we have presented the most common evaluation metrics and approaches used in the literature, highlighting the importance of resampling techniques in order to obtain a reliable estimate of the generalization ability of the SED systems. Finally, we have briefly reviewed the statistical tests used for comparing the performance of multiple classifiers.

## PART II: EXPERIMENTS

In this part we describe the methodology that we have followed for evaluating and comparing sound event detection approaches. It is composed of two chapters. In the first chapter we present the experimental setup defined to evaluate the performance of our SED system, whereas in the second chapter, we discuss the results of our experiments.

## Chapter 3: Experimental setup

### 3.1 Introduction

This chapter presents the experimental setup defined to conduct our experiments. First, in Section 3.2 we present our dataset. Next, in Section 3.3 we present the tools that we have used to conduct our experiments. Then in Section 3.4 we describe our implemented SED system. Furthermore, in section 3.5 we explain partitioning our dataset into training and testing sets. Finally in Sections 3.6 and 3.7 respectively we discuss the features and machine learning models used in our experiments.

### 3.2 Dataset

The dataset used for this task is a combination of DCASE 2017 and DCASE 2016 datasets. Both of them were collected in Finland by Tampere University of Technology between June 2015 and January 2016 [2, 20] and contain **multiple overlapping sound events**. The DCASE 2017 dataset consists of recordings of street with various levels of traffic and other activity. However, the DCASE 2016 dataset consist of recordings in Home and Residential area. The recordings were selected as representing an environment of interest for detection of sound events related to human activities and hazard situations. They were captured using 44.1 kHz sampling rate and are 3-5 minute long. Individual sound events in each recording were annotated by the same person, he was instructed to annotate all audible sound events, decide the start time and end time of the sounds and chose event labels freely [20].

In order to avoid the problem of rare events that are hard to detect which affect the performance of the system, we have decided to work with sound classes taken from residential areas and streets as they are the most dominant. Table 3.1 gives an overview of the number of recordings in each event class and the actual frequency of occurrence of each event. On the other hand, Figure 3.1 and 3.2, respectively, represent two sounds taken from our dataset. One sound was recorded in a street and the other in a residential area.

Table 3.1: Frequency of occurrence of each sound event within the dataset.

Event class	Occurrence
Brakes squeaking	287
Car	5116
Children	607
People speaking	1908
People walking	2691
<b>Total # of events</b>	<b>10609</b>

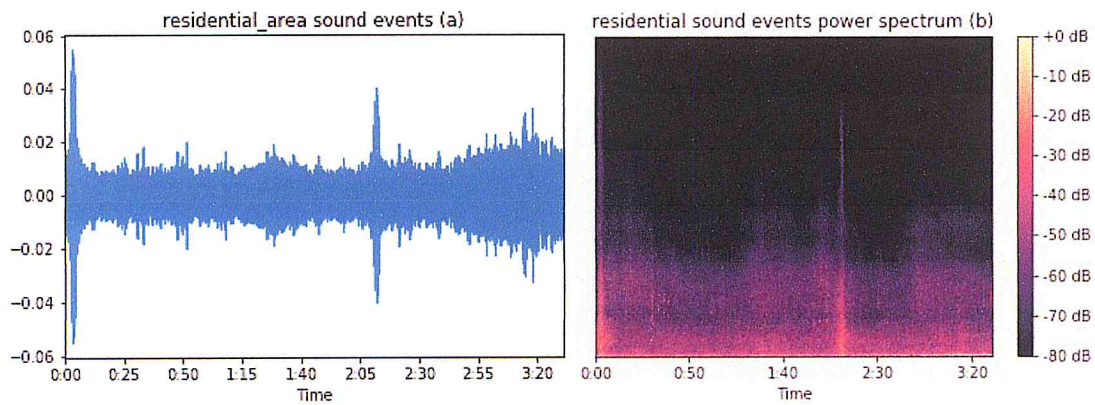


Figure 3.1: Residential area: (a) time domain, (b) frequency domain.

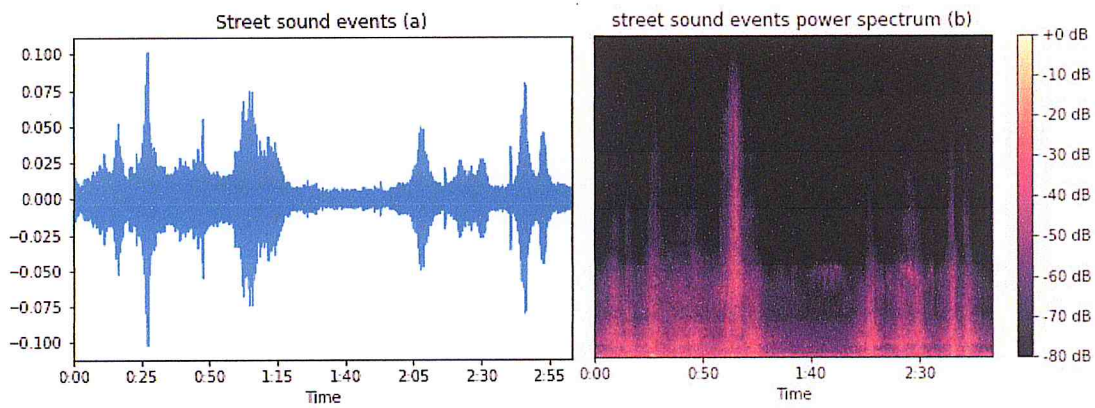
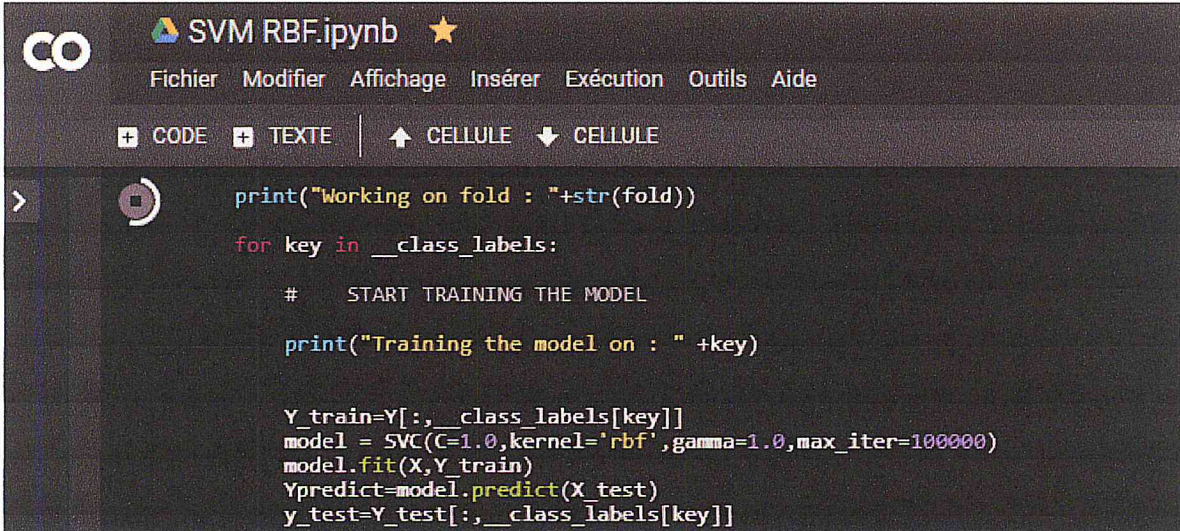


Figure 3.2: Street: (a) time domain, (b) frequency domain.

### 3.3 Tools

We have carried our experiments using Python 3.6 which is an object-oriented open source programming language [49]. First, Feature extraction was performed using Spyder 3.3.4 which is a scientific environment based on Python [50] and a collection of Python DCASE (Detection and Classification of Acoustic Scenes and Events) utilities created for the DCASE challenge 2017 and 2016 [2]. We have displayed our features using Librosa 0.6.3 which is a Python package for audio analysis [51]. Moreover, the machine learning process was performed using a set of Python packages such as Scikit-learn which is a machine learning library; it exposes a wide variety of machine learning algorithms [52]. Other libraries that were invoked include Numpy, Seaborn 0.9.0, Pandas, Keras, Tensorflow [53, 54].

We have trained our classifiers using [Google Colaboratory](#) which is a free Cloud service. it consists of executable Python notebooks stored within Google Drive and connected to a Cloud-based runtime to perform the execution of the Python code on Nvidia Tesla K80 GPUs and TPUs.



```
print("Working on fold : "+str(fold))

for key in __class_labels:

    # START TRAINING THE MODEL

    print("Training the model on : " +key)

    Y_train=Y[:, __class_labels[key]]
    model = SVC(C=1.0, kernel='rbf', gamma=1.0, max_iter=100000)
    model.fit(X, Y_train)
    Ypredict=model.predict(X_test)
    y_test=Y_test[:, __class_labels[key]]
```

Figure 3.3: Screenshot of online Google Colab platform.

### 3.4 System description

We aim at building different **monophonic SED systems** that are able to indicate whether the event is present or not within the given set of **polyphonic recordings** described by feature vectors. Each system utilizes a specific feature extraction technique along with different classification paradigms. We have carried out two sets of experiments. First, we have examined 3 different feature extraction techniques (MFCCs, Log Mel-band Energy and MFCCs +  $\Delta$ MFCCs +  $\Delta\Delta$ MFCCs) and 4 classification paradigms (SVM, CNN, Adaboost and Random Forest), while varying their parameters. We have utilized the **1 second segment-based approach** along with the **Fscore** and **Accuracy** metrics to perform the evaluation of our systems and have supported our analysis and discussion with numerous statistical tests. Second, we have investigated the effect of the number of features on the generalization performance of the Random Forest classifier where we have invoked a **feature selection** approach, namely **minimum redundancy maximum relevance (MRMR)** [55], in order to automatically determine the optimal set of extracted features. The process of SED is described in Figure 3.4. Additional details on the feature extraction techniques and the machine learning models that were used can be found in Sections 3.6 and 3.7.

### 3.5 Cross-validation

For evaluating our SED system, we have performed a **stratified 10-cross validation** to divide the whole dataset into non-overlapping training and testing sets. We have **shuffled** our data to generate different combinations in order to ensure that the events are present in the testing set. Cross validation leads to a considerable computation time increment, but it is a very common test used to evaluate the flexibility of the system when different training data are used. When all models are trained, an overall performance score is calculated by taking the average of the results over the 10 folds. We have performed **10 cross validation** using Scikit-learn.

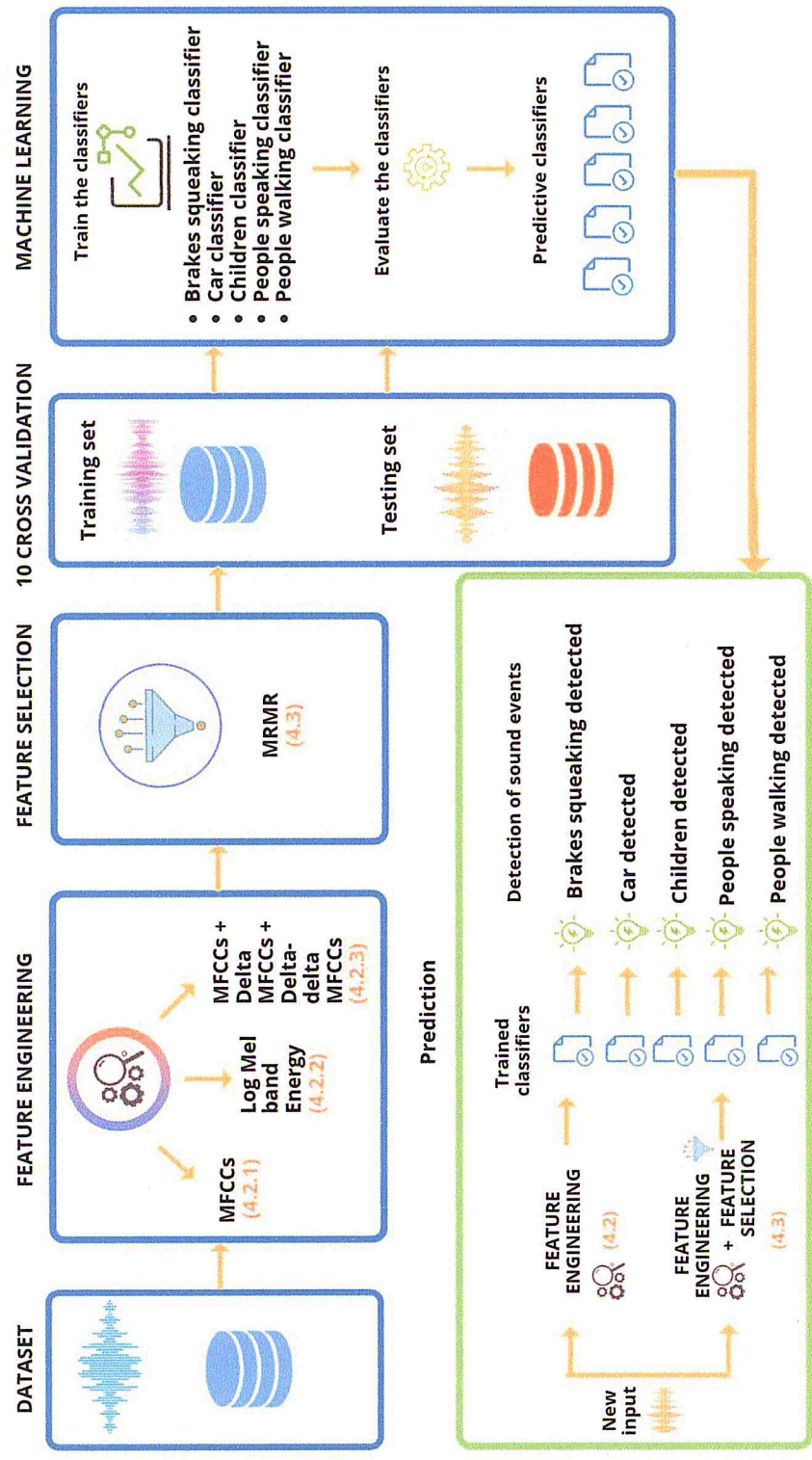


Figure 3.4: The process of Sound Event Detection.

### 3.6 Feature extraction and selection

We have used the frequency domain features **log Mel-band energies**, MFCCs and a combination of MFCCs +  $\Delta$ MFCCs +  $\Delta\Delta$ MFCCs. Figure 3.5 presents the feature engineering process that we have followed in our work.

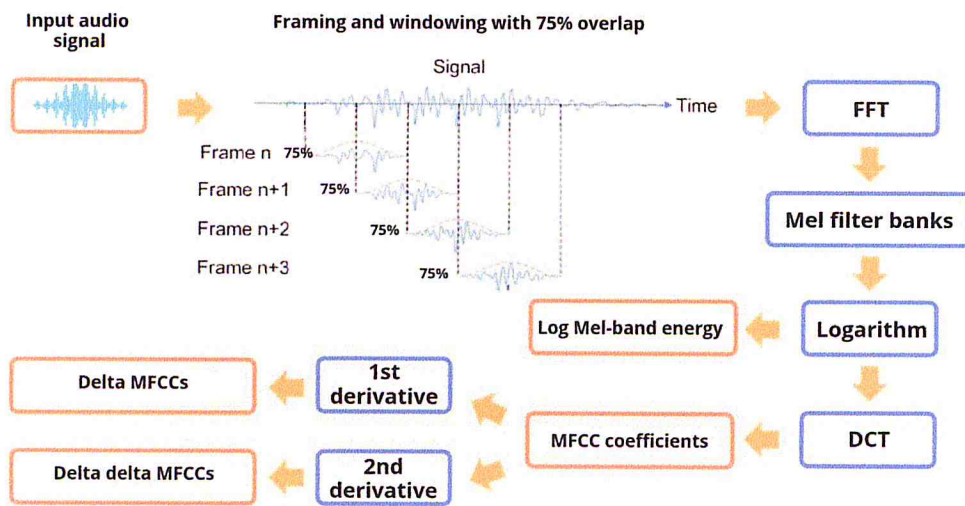


Figure 3.5: The process of feature engineering.

#### 3.6.1 Log Mel-band energy

Table 3.2 summarizes the parameters used for extracting the **Log Mel-band energy** features. Figures 3.6 and 3.7 depict the **Log Mel-band energy** features for 2 sounds taken from our dataset.

Table 3.2: Log Mel-band energy setup.

Parameter	Value
Sample rate	44100Hz
Frame length	40 ms
Overlap (%)	75%
#Mel band filters	40
#Log Mel energy	40
Window function	Hamming
<b>Total # of features</b>	<b>40</b>



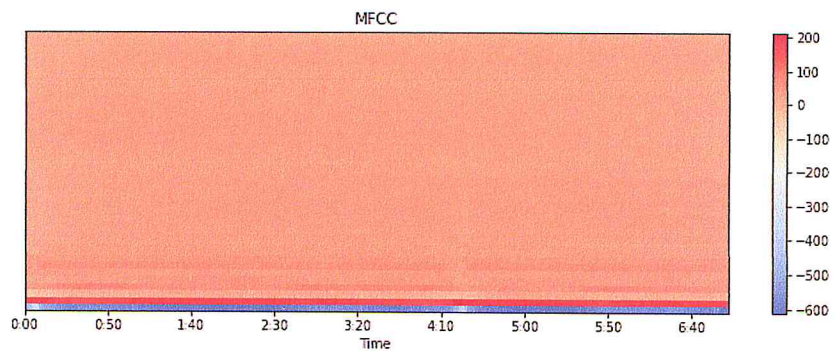


Figure 3.8: MFCCs of a residential areas recording.

### 3.6.3 MFCC and $\Delta$ MFCC and $\Delta\Delta$ MFCC

The setup is summarized in Table 3.4

Table 3.4: MFCC+ $\Delta$ MFCC+ $\Delta\Delta$ MFCC setup.

Parameter	Value
Sample rate	44100Hz
Frame length	40 ms
Overlap (%)	75%
#Mel band filters	40
#MFCC coefficients	40
# $\Delta$ MFCCs	40
# $\Delta\Delta$ MFCCs	40
Window function	Hamming
<b>Total # of features</b>	<b>120</b>

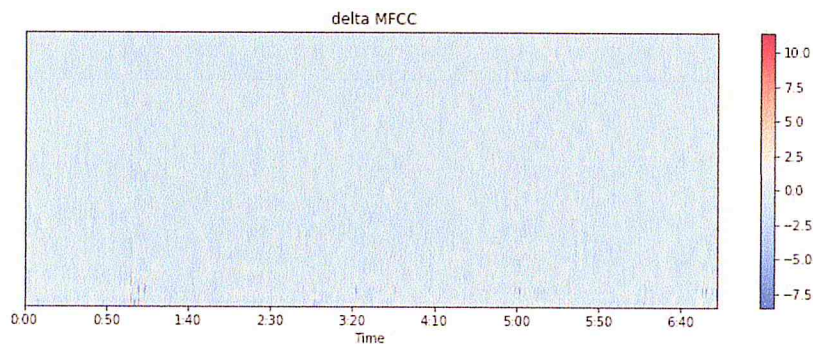


Figure 3.9:  $\Delta$ MFCCs of a residential area recording.

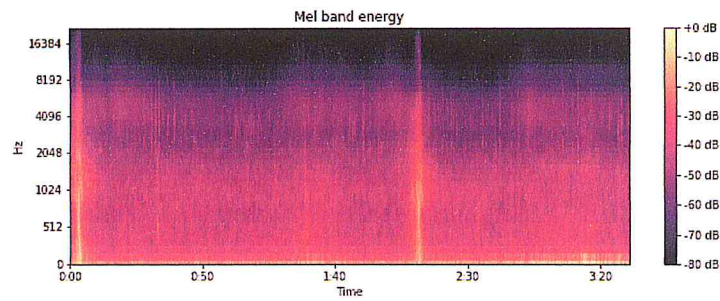


Figure 3.6: Log Mel band energies for a residential area recording.

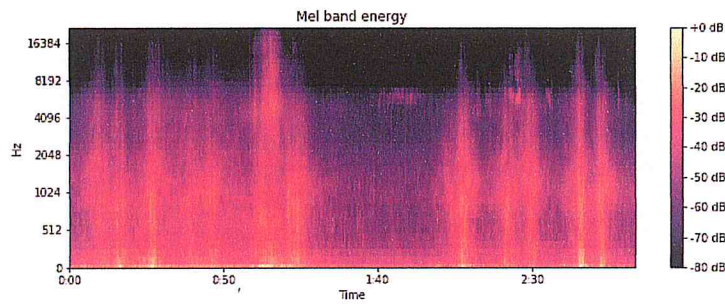


Figure 3.7: Log Mel band energies for a street recording.

### 3.6.2 Mel Frequency Cepstral Coefficients

After calculating the log Mel band energies, DCT of the 40 log Mel band energies is taken and 40 MFCC coefficients are obtained. Table 3.3 summarizes the parameters used for extracting the above features. Figure 3.8 displays the MFCC coefficients of a sound taken from our dataset.

Table 3.3: MFCC setup.

Parameter	Value
Sample rate	44100Hz
Frame length	40 ms
Overlap (%)	75%
#Mel band filters	40
#MFCC coefficients	40
Window function	Hamming
<b>Total # of features</b>	<b>40</b>

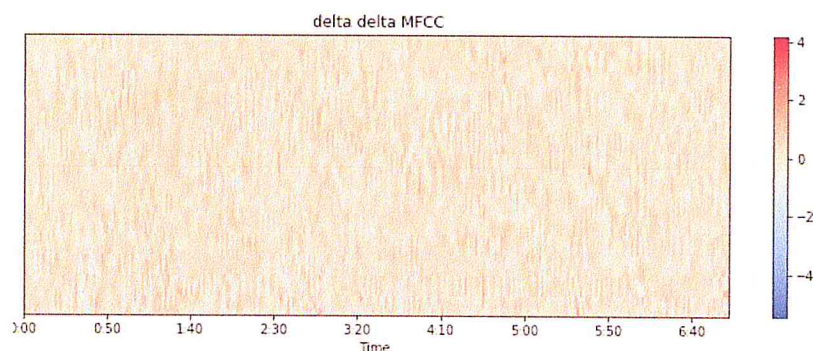


Figure 3.10:  $\Delta\Delta$ MFCCs of a residential area recording.

### 3.6.4 Feature selection

We have invoked a feature selection approach, namely minimum redundancy maximum relevance (MRMR) to automatically determine the optimal 40% , 60% and 80% sets of 120 extracted MFCC+ $\Delta$ MFCC+ $\Delta\Delta$ MFCC using Random Forest classifier in order to investigate the effect of the number of features on its generalization performance.

## 3.7 Classification approaches

The Classification process for sound event detection is illustrated in Figure 3.11. It is performed based on feature vectors composed of a set of acoustic features  $x_t$  extracted from  $t = 1, \dots, T$  frames. Each model takes as input acoustic feature  $x_t \in \mathbb{R}^X$  extracted from a single frame and is trained to learn a single event among the 5 available event classes. We define  $y_t \in Y = \{0,1\}$  as the class label associated to each frame. If the event class is present in frame  $t$  then  $y_t$  is set to 1 and 0 otherwise. A classifier is seen as a two-step algorithm: training phase and testing phase. In the training phase the model is trained to learn the mapping from  $T = \{(x_{1t}, y_{1t}), (x_{2t}, y_{2t}), \dots, (x_{mt}, y_{mt})\}$  and produces probability distribution output vector  $\hat{y} \in [0,1]$  (CNN model) or oracle output vector  $\hat{y} \in \{0,1\}$  (SVM, Adaboost and Random Forest models). In the testing phase, the produced model is used to predict the class labels for unseen sound recordings from the testing set. When the predicted outputs consist of probability distributions, a post-processing step is performed to obtain oracle outputs. This is done by using a **threshold value equal to 0.5**. The oracle outputs represent the frame-level binary estimate for

each sound event class in consecutive frames and allow us to obtain the **temporal activity information** i.e. Extract the **onset** and **offset** of each sound event class.

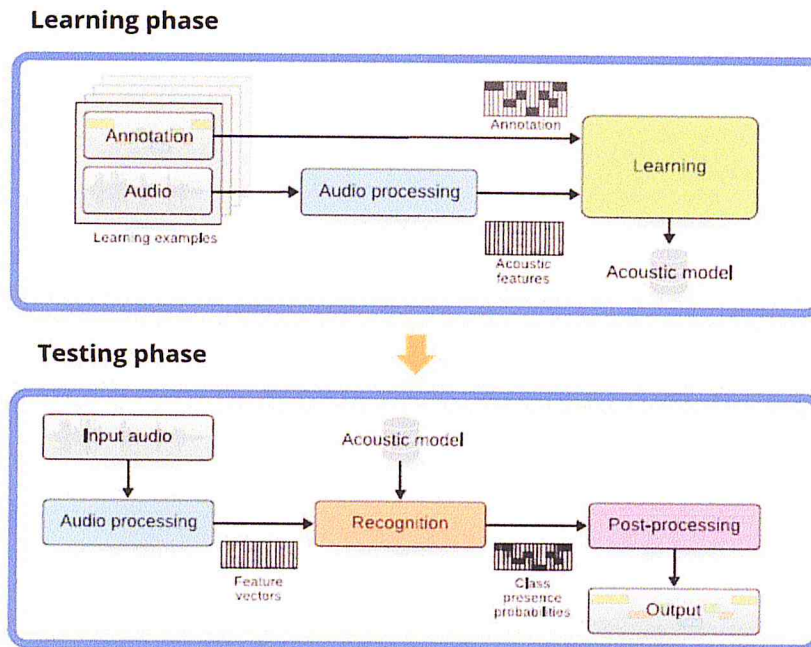


Figure 3.11: The training and testing phase of a SED system [21].

To compare the performance of each SED system, we have examined 7 supervised machine learning classifiers as described in Table 3.5. We have implemented the first 5 classifiers in Python using the Scikit learn library [52], whereas for the CNN classifier, we have employed Keras and Tensorflow libraries. It is worth underscoring that we have performed a feature scaling to our feature vectors using Scikit learn preprocessing standard scaler before training our models in order to make sure that our feature takes similar range of values.

Table 3.5: Machine learning setup.

Classifier	Parameter	Value
Random Forest(RF_1)	N_estimators	50
	Max_depth	40
Random Forest(RF_2)	N_estimators	200
	Max_depth	40
Adaboost(AB)	N_estimators	50
	Max_depth	40
Gaussian SVM(SVM_1)	Max_iteration	100 000
	Gamma	1.0
Polynomial SVM(SVM_2)	Max_iteration	100 000
	Gamma	1.0
	Order	3.0
CNN_1	#Epochs	400
	Batch size	128
	Dropout rate	0.3
	#Hidden layers	3
	Pool size	(5.2.2)
	Filters	128
	Activation function	Relu
	Threshold	0.5
CNN_2	#Epochs	1000
	Batch size	128
	Dropout rate	0.3
	#Hidden layers	3
	Pool size	(5.2.2)
	Filters	128
	Activation function	Relu
Threshold	0.5	

### 3.8 Conclusion

This chapter introduced the experimental setup defined to conduct our experimental comparisons. We have presented our dataset and the way we split it into training and testing set in order to perform the evaluation of our system. We have also presented the tools that we have used for conducting our experiment and the topology of our SED system. We have highlighted the fact that our experimental study is based different set of spectral features and machine learning models and have exposed their parameters.

# Chapter 4: Experimental results and discussion

## 4.1 Introduction

This chapter discusses the experimental results that we have obtained during our experiments. In Sections 4.2 and 4.3 we analyze and discuss the results of the first and second set of our experiments. In Section 4.4 we talk about the training time of our experimental studies.

## 4.2 First set of experiment

We have conducted extensive experimental comparison among sound analysis methods using 5 types of **event sounds**. We have thoroughly examined 3 different feature extraction techniques: **MFCCs** for the first case study, **Log Mel-band Energy** for the second case study and a combination of **MFCCs +  $\Delta$ MFCCs +  $\Delta\Delta$ MFCCs** for the last case study to examine the performance of the 7 classifiers described in Table 3.5 for each type of the above mentioned features. We have utilized the **1 second segment-based approach** along with the **Fscore** and **Accuracy** metrics to perform the evaluation of our systems and have supported our analysis and discussion with numerous statistical tests.

### 4.2.1 Case study 1

In this case study, we compare the performance of 7 classifiers (SED Systems) using MFCCs. Table 4.1 gives the average accuracy (%) results that we have obtained. The first column represents the event class and the rest of the columns designates the type of classifiers that are used in our experiment. The last row specifies the mean rank of each classifier over all events.

Table 4.1: Average classification accuracy (%) results based on MFCCs.

Event	SVM_1	SVM_2	CNN_1	CNN_2	RF_1	RF_2	AB
Brakes squeaking	83.00 ± 0.28	82.94 ± 0.28	92.43 ± 0.04	92.43 ± 0.04	92.43 ± 0.04	92.43 ± 0.04	<b>92.45 ± 0.04</b>
Car	67.01 ± 0.24	58.71 ± 0.20	69.53 ± 0.07	69.37 ± 0.07	73.80 ± 0.08	<b>74.20 ± 0.08</b>	74.19 ± 0.08
Children	80.03 ± 0.29	80.15 ± 0.29	88.30 ± 0.11	88.07 ± 0.11	89.51 ± 0.11	<b>89.53 ± 0.11</b>	<b>89.53 ± 0.11</b>
People speaking	72.21 ± 0.30	71.51 ± 0.29	77.42 ± 0.18	77.52 ± 0.18	81.11 ± 0.19	<b>81.22 ± 0.19</b>	81.20 ± 0.19
People walking	73.99 ± 0.27	67.82 ± 0.24	76.74 ± 0.10	77.35 ± 0.10	81.02 ± 0.11	<b>81.09 ± 0.11</b>	81.06 ± 0.11
<b>Overall accuracy Instance-based(%)</b>	79.48 ± 0.07	74.13 ± 0.06	75.71 ± 0.07	75.75 ± 0.07	79.28 ± 0.07	79.48 ± 0.07	79.47 ± 0.07
<b>Overall accuracy Class-based(%)</b>	75.25 ± 0.26	72.23 ± 0.25	80.88 ± 0.06	80.95 ± 0.06	83.57 ± 0.07	83.69 ± 0.07	83.69 ± 0.07
<b>Mean ranks</b>	<b>6.20</b>	<b>6.80</b>	<b>4.30</b>	<b>4.30</b>	<b>3.10</b>	<b>1.60</b>	<b>1.70</b>

We statistically compare the performances of these techniques using **Friedman test**. Under the null hypothesis, we have assumed that all classifiers are equivalent and the observed differences are due to chance. **Friedman test** rejects this hypothesis with  $FF = 30.15 > F(6,24) = 30.05$  for  $\alpha = 5 \times 10^{-10}$  ( $FF$  is distributed according to the  $F$  distribution with  $7 - 1 = 6$  and  $(7 - 1) \times (5 - 1) = 24$  degrees of freedom), and therefore confirms the existence of at least one pair of techniques with significantly different performances. We have followed up the previous findings with a **Nemenyi test** at a 10% significance level with the critical value  $q_{0.1} = 2.69$  and the critical difference  $CD = 3.68$ . The results of this test are depicted in Figure 4.1.

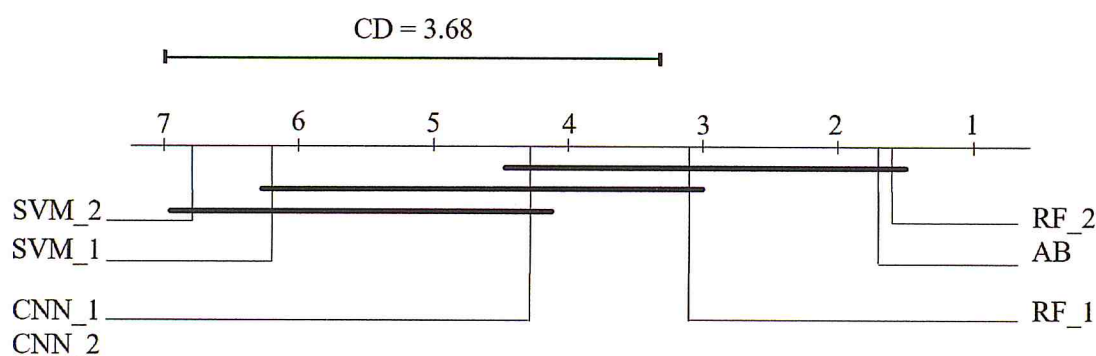


Figure 4.1: Comparison of all systems against each other with the Nemenyi test. Groups of techniques that are not significantly different (at  $\alpha = 0.10$ ) are connected.

The analysis of the previous results can be summarized as follows. **Nemenyi test** indicates that there is no significant difference between SVM classifiers, and the observed differences are merely due to random behavior. We also observe that the ensemble-based classifiers (Random Forest and Adaboost) perform significantly better than SVM. However, data are not sufficient to reach the same conclusion regarding CNN. Most importantly, CNN classifiers strangely show weak performances compared to Random Forest and Adaboost. This behavior is expected since CNN require more data to improve its generalization ability [27], whereas, ensemble-based learners are able to adapt well and learn from **class-imbalanced data** [37], which is the case of SED.

The Box plot depicted in Figure 4.2 represents the distribution of the mean accuracy scores over all events for each SED system. We observe that the ensemble-based approaches have the lowest variance and demonstrate the best performance as the majority of the accuracy scores are between 81% and 89% followed by CNNs. Moreover, the performance of the SVM Polynomial is very weak compared to the rest as it has the largest variance.

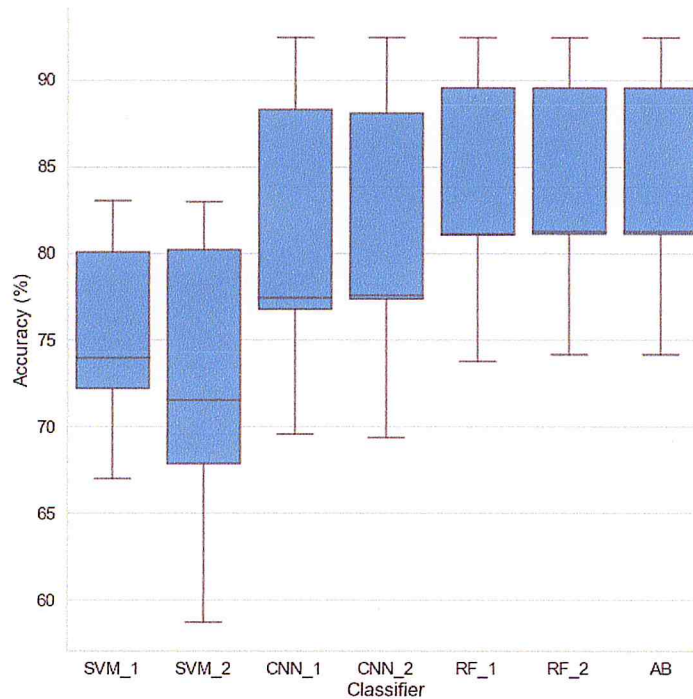


Figure 4.2: The average accuracy scores (%) over all events for each SED system



### 4.2.2 Case study 2

In this case study, we compare the performance of the 7 SED systems using the **Log Mel-band energy features**. Table 4.2 gives the average accuracy (%) results that we have obtained. The first column represents the event class and the rest of the columns designates the type of classifiers that are used in our experiment. The last row specifies the mean rank of each classifier over all events.

Table 4.2: Average classification accuracy (%) results based on Log Mel-band energy.

Event	SVM_1	SVM_2	CNN_1	CNN_2	RF_1	RF_2	AB
Brakes squeaking	<b>92.43 ± 0.04</b>	90.51 ± 0.07	<b>92.43 ± 0.04</b>	<b>92.43 ± 0.04</b>	<b>92.43 ± 0.04</b>	<b>92.43 ± 0.04</b>	92.36 ± 0.04
Car	72.41 ± 0.07	57.92 ± 0.11	67.58 ± 0.07	68.05 ± 0.07	73.97 ± 0.08	<b>74.19 ± 0.08</b>	74.13 ± 0.08
Children	88.86 ± 0.11	85.45 ± 0.11	89.00 ± 0.11	87.92 ± 0.11	89.44 ± 0.11	<b>89.53 ± 0.11</b>	89.51 ± 0.11
People speaking	79.27 ± 0.18	71.31 ± 0.15	77.40 ± 0.17	77.73 ± 0.18	81.15 ± 0.19	<b>81.22 ± 0.19</b>	81.19 ± 0.19
People walking	78.86 ± 0.10	58.42 ± 0.18	76.38 ± 0.10	76.65 ± 0.10	80.91 ± 0.11	<b>81.09 ± 0.11</b>	81.04 ± 0.11
<b>Overall accuracy Instance-based (%)</b>	77.74 ± 0.06	65.63 ± 0.08	74.90 ± 0.06	75.15 ± 0.07	79.32 ± 0.07	<b>79.47 ± 0.07</b>	79.42 ± 0.07
<b>Overall accuracy Class-based (%)</b>	82.37 ± 0.06	72.72 ± 0.07	80.56 ± 0.06	80.55 ± 0.07	83.58 ± 0.07	<b>83.69 ± 0.07</b>	83.65 ± 0.07
<b>Mean ranks</b>	<b>4.00</b>	<b>7.00</b>	<b>5.00</b>	<b>4.80</b>	<b>3.00</b>	<b>1.40</b>	<b>2.80</b>

Following Demsar's recommendations [31], we have first conducted a **Friedman test** to statistically compare the performance of these systems. We have assumed that all systems perform similarly and the observed differences are merely due to chance. **Friedman test** rejects this hypothesis with  $FF = 9.73 > F(7,203) = 8.55$  for  $\alpha = 5.0 \times 10^{-5}$  ( $FF$  is distributed according to the  $F$  distribution with  $7 - 1 = 6$  and  $(7 - 1) \times (5 - 1) = 24$  degrees of freedom), and therefore confirms the existence of at least one pair of techniques with significantly different performances.

Then, we have tested the pairwise significance differences using a **Nemenyi test** at a 10% significance level with the critical value  $q_{0.1} = 2.69$  and the critical difference  $CD = 3.68$ .

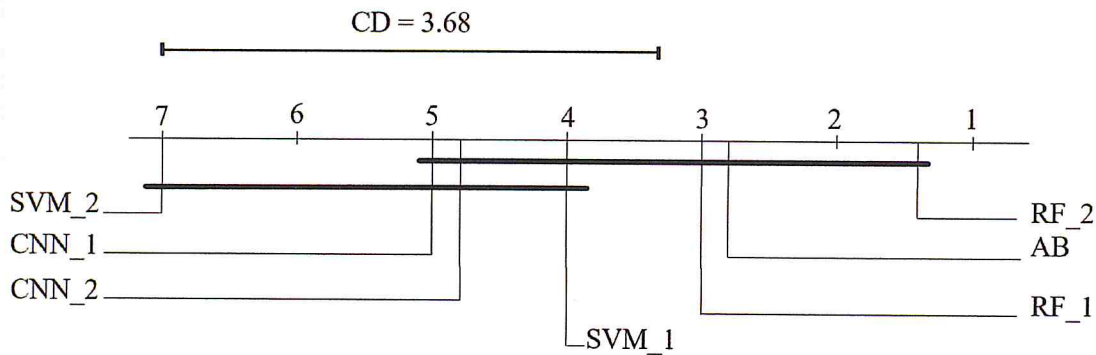


Figure 4.3: Comparison of all systems against each other with the Nemenyi test. Groups of techniques that are not significantly different (at  $\alpha = 0.10$ ) are connected.

The analysis of the test results illustrated by Figure 4.3 can be summarized by three main observations:

- The results of the Ensemble-based learners are in the lead followed by SVM with Gaussian kernel (SVM\_1) and CNNs in the third place. This latter observation confirms our previous assumption i.e. Convolutional Neural Network classifiers may not have been well-trained due to the lack of sound events data.
- Similarly to our previous findings, there is no statistical difference between CNN\_1 and CNN\_2.
- SVM with polynomial kernel (SVM\_2) exhibits very poor performance compared to the gaussian one (SVM\_1). A possible cause of this behavior may be related to the nature of Log Mel Band Energy data and the procedure of projecting samples into high dimensional space (kernel).

The Box plot depicted in Figure 4.4 represents the distribution of the mean accuracy scores (%) over all events for each SED system. We observe that the ensemble-based approaches have the lowest variance and demonstrate the best performance as the majority of the accuracy scores are between 81% and 89% followed by the SVM\_1 with an accuracy score between 79% and 88% and CNNs. Moreover, similarly to our previous findings the performance of the SVM\_2 is still very weak compared to the rest as it has the largest variance.

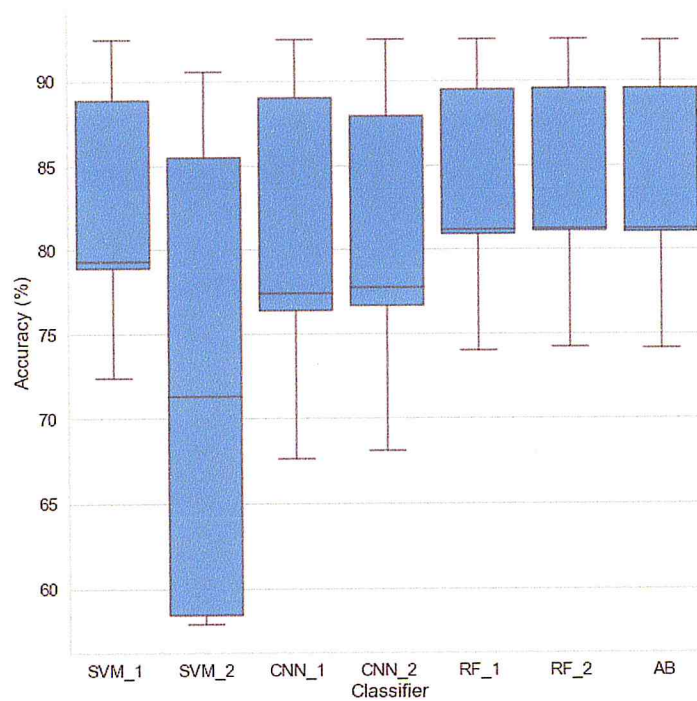


Figure 4.4: The average accuracy scores (%) over all events for each SED system

### 4.2.3 Case study 3

This section is devoted to investigate the influence of  $\Delta\text{MFCC} + \Delta\Delta\text{MFCC}$  on the generalization ability of SVM\_1, SVM\_2, RF\_1, RF\_2, and Adaboost. To this end, we have carried out pairwise comparisons between the aforementioned techniques with and without adding  $\Delta\text{MFCC} + \Delta\Delta\text{MFCC}$  features. Due to its robustness, we have considered using the **Wilcoxon signed-ranks tests**. A summary of this experiment and test statistics is shown in Table 4.3 and 4.4. Row 9 specifies the number of win/tie/loss of the system trained with MFCC+ $\Delta\text{MFCC} + \Delta\Delta\text{MFCC}$  (column highlighted in grey) over the system which uses only MFCC features. Row 10 shows the associated p – value; it is worth noting that p – values  $\leq 0.05$  indicates that the system in the column highlighted in grey is significantly better than the system trained only with MFCC features at 5% significance level i.e. **adding  $\Delta\text{MFCC} + \Delta\Delta\text{MFCC}$  significantly improves the predictive performance.**

Table 4.3: Average classification accuracy (%) results of SVMs using MFCC and MFCC+ $\Delta$ MFCC+ $\Delta\Delta$ MFCC

Event	SVM_1	SVM_1	SVM_2	SVM_2
	MFCC	MFCC + $\Delta$ MFCC+ $\Delta\Delta$ MFCC	MFCC	MFCC + $\Delta$ MFCC+ $\Delta\Delta$ MFCC
Brakes squeaking	83.00 $\pm$ 0.28	92.43 $\pm$ 0.04	82.94 $\pm$ 0.28	92.43 $\pm$ 0.04
Car	67.01 $\pm$ 0.24	74.20 $\pm$ 0.08	58.71 $\pm$ 0.20	74.12 $\pm$ 0.08
Children	80.03 $\pm$ 0.29	89.53 $\pm$ 0.11	80.15 $\pm$ 0.29	89.53 $\pm$ 0.11
People speaking	72.21 $\pm$ 0.30	81.22 $\pm$ 0.19	71.51 $\pm$ 0.29	81.26 $\pm$ 0.19
People walking	73.99 $\pm$ 0.27	81.09 $\pm$ 0.11	67.82 $\pm$ 0.24	81.05 $\pm$ 0.11
<b>Overall accuracy</b> <b>Instance-based (%)</b>	79.48 $\pm$ 0.07	79.48 $\pm$ 0.07	74.13 $\pm$ 0.06	79.45 $\pm$ 0.07
<b>Overall accuracy</b> <b>Class-based (%)</b>	75.25 $\pm$ 0.26	83.69 $\pm$ 0.07	72.23 $\pm$ 0.25	83.68 $\pm$ 0.07
<i>W/T/L</i>		5/0/0		5/0/0
<i>p - value</i>		0.044		0.044

The results shown in Table 4.3 indicate that training SVMs with MFCC+ $\Delta$ MFCC+ $\Delta\Delta$ MFCC considerably improves the generalization ability with  $p - value \leq 0.044$ ; whereas, introducing additional features did not demonstrate any improvement in the performance of ensemble-based learners as depicted in Table 4.4.

Table 4.4: Average classification accuracy (%) results of ensemble learners using MFCC and MFCC+ $\Delta$ MFCC+ $\Delta\Delta$ MFCC.

Event	RF_1	RF_1	RF_2	RF_2	AB	AB
	MFCC	MFCC + $\Delta$ MFCC+ $\Delta\Delta$ MFCC	MFCC	MFCC + $\Delta$ MFCC+ $\Delta\Delta$ MFCC	MFCC	MFCC + $\Delta$ MFCC+ $\Delta\Delta$ MFCC
Brakes squeaking	92.43 $\pm$ 0.04	92.43 $\pm$ 0.04	92.43 $\pm$ 0.04	92.43 $\pm$ 0.04	92.45 $\pm$ 0.04	92.43 $\pm$ 0.04
Car	73.80 $\pm$ 0.08	74.01 $\pm$ 0.08	74.20 $\pm$ 0.08	74.20 $\pm$ 0.08	74.19 $\pm$ 0.08	74.20 $\pm$ 0.08
Children	89.51 $\pm$ 0.11	89.47 $\pm$ 0.11	89.53 $\pm$ 0.11	89.53 $\pm$ 0.11	89.53 $\pm$ 0.11	89.53 $\pm$ 0.11
People speaking	81.11 $\pm$ 0.19	81.09 $\pm$ 0.18	81.22 $\pm$ 0.19	81.22 $\pm$ 0.19	81.20 $\pm$ 0.19	81.22 $\pm$ 0.19
People walking	81.02 $\pm$ 0.11	80.85 $\pm$ 0.11	81.09 $\pm$ 0.11	81.09 $\pm$ 0.11	81.06 $\pm$ 0.11	81.09 $\pm$ 0.11
<b>Overall accuracy</b> <b>Instance-based (%)</b>	79.28 $\pm$ 0.07	79.31 $\pm$ 0.07	79.48 $\pm$ 0.07	79.48 $\pm$ 0.07	79.47 $\pm$ 0.07	79.48 $\pm$ 0.07
<b>Overall accuracy</b> <b>Class-based (%)</b>	83.57 $\pm$ 0.07	83.57 $\pm$ 0.07	83.69 $\pm$ 0.07	83.69 $\pm$ 0.07	83.69 $\pm$ 0.07	83.69 $\pm$ 0.07
<i>W/T/L</i>		1/1/3		0/5/0		3/1/1
<i>p - value</i>		0.59		1		0.28

Based on these observations, we believe that adding  $\Delta\text{MFCC}+\Delta\Delta\text{MFCC}$  features provides a better representation of the sound events. More specifically, the projection of these features into a higher dimensional space using a kernel trick has improved both the decision boundary and margins determined by SVM; which was not possible using only MFCCs. From this experiment, we can derive two lessons:

- $\Delta\text{MFCC}+\Delta\Delta\text{MFCC}$  features have a major impact on the generalization ability of SVM-based systems.
- Adding  $\Delta\text{MFCC}+\Delta\Delta\text{MFCC}$  do not have any significant influence on the performance of ensemble learning classifiers.

#### 4.2.4 Summary of results

In order to investigate the results of our first set of experiment, we present in Figure 4.5 the F-score of the 7 SED systems for each sound event. We can observe that all systems show very low detection rates of the event “brakes squeaking” with an Fscore value  $< 6\%$ . This is due to the fact that the numbers of instances of this event is very low (as shown in Table 3.1), which is not sufficient for learning an effective model. We can also note that CNN-based systems exhibit very poor performance in case of “brakes squeaking” and “children”. This behavior is expected since CNN model requires more data than the other classifiers to improve its performance. Furthermore, because the numbers of instances of the three remaining events are higher comparing to the above mentioned events, we can observe a better performance with an Fscore value  $\leq 37$  for people speaking followed by an Fscore value  $\leq 48$  for people walking and Fscore value  $\leq 50$  for car.

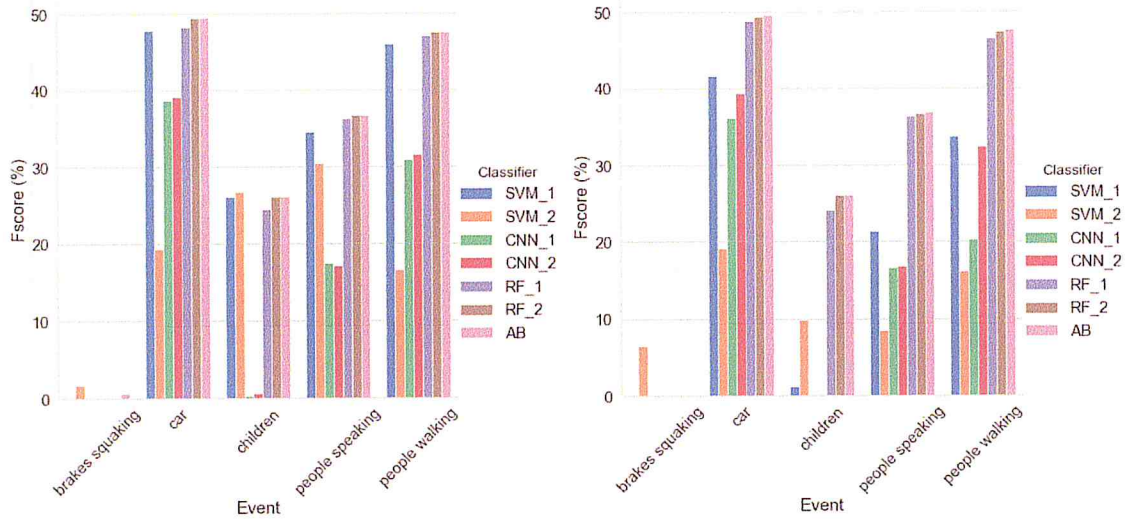


Figure 4.5: Overall Fscore results (%) of SED systems for 1<sup>st</sup> case study (left), the 2<sup>nd</sup> (right).

Table 4.5: Overall Fscore (%) results for each case study.

Case studies		SVM_1	SVM_2	CNN_1	CNN_2	RF_1	RF_2	AB
Case study 1	Overall Fscore	45.0 ± 0.17	23.59 ± 0.08	32.61 ± 0.15	33.18 ± 0.14	44.17 ± 0.17	45.0 ± 0.17	<b>45.01 ± 0.17</b>
	Instance-based							
Case study 2	Overall Fscore	30.88 ± 0.16	18.98 ± 0.11	17.45 ± 0.08	17.69 ± 0.07	31.2 ± 0.14	31.93 ± 0.15	<b>32.06 ± 0.15</b>
	Class-based							
Case study 3	Overall Fscore	34.91 ± 0.15	18.67 ± 0.12	28.06 ± 0.13	33.9 ± 0.14	44.34 ± 0.17	44.99 ± 0.17	<b>45.07 ± 0.17</b>
	Instance-based							
Case study 3	Overall Fscore	19.58 ± 0.09	12.05 ± 0.06	14.6 ± 0.08	17.72 ± 0.07	31.15 ± 0.14	31.93 ± 0.15	<b>32.03 ± 0.15</b>
	Class-based							
Case study 3	Overall Fscore	45.00 ± 0.09	<b>45.14 ± 0.06</b>	26.78 ± 0.08	32.33 ± 0.07	44.22 ± 0.14	45.0 ± 0.15	45.01 ± 0.15
	Instance-based							
Case study 3	Overall Fscore	31.93 ± 0.09	<b>32.03 ± 0.06</b>	15.35 ± 0.08	17.32 ± 0.07	31.33 ± 0.14	31.93 ± 0.15	31.94 ± 0.15
	Class-based							

Table 4.5 summarizes the overall F-scores (%) obtained during the first set of our experiments. We can observe that Adaboost demonstrates the best detection performance in both of case study 1 and 2. However in case study 3 SVM\_2 demonstrates a better detection performance which is expected as discussed Section 4.2.3.

Based on the results of our experimental enquiries, we can derive the following lessons. :

- The SED system based on the ensemble learning classifiers (RF\_2 and AB) along with MFCCs achieve the best detection performance with an overall class-based accuracy of 83% and an overall instance-based Fscore of 45%.
- SVMs along with MFCCs +  $\Delta$ MFCCs +  $\Delta\Delta$ MFCs achieve a significant improvement in the detection performance with an overall class-based accuracy of 83% and an overall instance-based Fscore of 45%.
- The lack of sound event data influences negatively the performance of CNNs.
- The MFCCs and MFCCs +  $\Delta$ MFCCs +  $\Delta\Delta$ MFCs features significantly improve the predictive performance of SED systems compared to the log Mel-band energy features.

### 4.3 Second set of experiment

In this case study, we have investigated the effect of the number of features on the generalization performance of Random Forest (RF\_2) classifier. Specifically, we have invoked a feature selection approach, namely mRMR in order to automatically determine the optimal 40% , 60% and 80% sets of 120 extracted MFCC+ $\Delta$ MFCC+ $\Delta\Delta$ MFCC. However due to the lack of data in our dataset, this techniques did not demonstrate any improvement in the performance our SED system as the selected features were not sufficient for learning an effective model. We believe that applying this feature selection technique along with data augmentation can lead to better performance.

### 4.4 Training time

The GPU requirement for SVMs training and testing was enormous. **The training and testing time per each event** took approximately **7 hours** for SVM\_1 and approximately **4 hours** for SVM\_2. Moreover, in the case of RF\_1, RF\_2 and AB classifiers it took approximately **seven hours** to **train and test a single fold** whereas it took **1 hour** to **train and test all of the 10 folds** for CNN\_2 and approximately **30 minutes** for the CNN\_1 classifier.

## CONCLUSION

### 1. Contributions and summary of experimental findings

The primary goal of this thesis was to conduct an **empirical analysis and comparisons among monophonic Sound Event Detection systems**. To this end, we have carried out **two set of experiments** to analyze their behavior using a combination of DCASE 2017 and DCASE 2016 datasets. First, we have examined 3 different feature extraction techniques (MFCCs, Log Mel-band Energy and MFCCs +  $\Delta$ MFCCs +  $\Delta\Delta$ MFCCs) and 4 classification paradigms (SVM, CNN, Random Forest and Adaboost), while varying their parameters. We have chosen one second segment based F-score and accuracy as our performance evaluation metrics and have founded our analysis and discussion with numerous statistical tests. From this experimental study, we can derive the following conclusions:

- The SED system based on the ensemble learning classifiers (RF\_2 and AB) along with MFCCs is the best in term of computational cost and detection performance with an overall class-based accuracy of 83% and an overall instance-based Fscore of 45%. Our reasoning on this is based on the fact that the ensemble learners are able to adapt well and learn from class-imbalanced data which is the case of SED.
- The SED systems based on SVMs along with MFCCs +  $\Delta$ MFCCs +  $\Delta\Delta$ MFCs achieve a significant improvement in the detection performance with an overall class-based accuracy of 83% and an overall instance-based Fscore of 45%. We believe that the projection of these features into a higher dimensional space using a kernel trick has improved both the decision boundary and margins determined by SVM. However the computational cost of these classifiers is enormous.
- The MFCC and MFCCs +  $\Delta$ MFCCs +  $\Delta\Delta$ MFCCs features significantly improve the predictive performance of SED systems compared to the log Mel-band energy features. This is due to the fact that the MFCCs extracted from a mixture of sounds are more relevant



and representative compared to log Mel-band energy. We believe that the log Mel-band energy features may represent well an individual sound event.

- The lack of sound event data influences negatively the performance of CNNs.

In the second set of experiments, we have investigated the effect of the number of features on the generalization performance of Random Forest classifier through invoking a feature selection approach, namely mRMR. However, from the experimental findings, we have noticed that due to the lack of data in this field of research, the selected features were not sufficient for learning an effective model. We believe that applying this technique along with data augmentation can yield better performance.

## **2. Limits and Future work**

This thesis has revealed several interesting areas for improvement. Based on the insights gained from the experimental findings, we have concluded that the lack of data can negatively affect the performance of SED systems. A natural extension of this work would be to exploit data augmentation techniques along with feature selection for a possible improvement in the behavior of such systems and analyze their effect on neural networks. Another appealing work direction would be to apply preprocessing techniques to diminish the effect of ambient background noise.

During this work we have faced several difficulties since the overlapping sound events in the audio recording cannot be well represented by the extracted features. Moreover, the process of framing in the SED area leads to imbalanced data which affect the performance of such systems. Also, even with the use of Nvidia Tesla K80 GPUs, the training time is still enormous.

This field of research is interesting as it contributes directly to the development of smart cities. We have acquired knowledge and many skills throughout the past 8 months, such as: fundamentals of Machine Learning and key steps for conducting proper Machine Learning experiments. We have also learned analyzing experimental findings based on statistical tests. Moreover, we have mastered Python and have discovered Google Collaboratory platform that we will continue using for future machine learning projects.

- [10] Cakır, E., Parascandolo, G., Heittola, T., Huttunen, H., & Virtanen, T. (2017). Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6), 1291-1303.
- [11] Adavanne, S., & Virtanen, T. (2017). A report on sound event detection with different binaural features. *arXiv preprint arXiv:1710.02997*.
- [12] Clavel, C., Ehrette, T., & Richard, G. (2005). Events detection for an audio-based surveillance system. In *2005 IEEE International Conference on Multimedia and Expo* (pp. 1306-1309).
- [13] Mesaros, A., Heittola, T., Eronen, A., & Virtanen, T. (2010). Acoustic event detection in real life recordings. In *2010 IEEE 18th European Signal Processing Conference* (pp. 1267-1271).
- [14] Lu, R., & Duan, Z. (2017). Bidirectional GRU for sound event detection. *Detection and Classification of Acoustic Scenes and Events*.
- [15] Jeong, I. Y., Lee, S., Han, Y., & Lee, K. (2017). Audio event detection using multiple-input convolutional neural network. *Detection and Classification of Acoustic Scenes and Events (DCASE)*.
- [16] Brown, G., Pocock, A., Zhao, M. J., & Luján, M. (2012). Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. *Journal of machine learning research*, 13(Jan), 27-66.
- [17] Battiti, R. (1994). Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on neural networks*, 5(4), 537-550.
- [18] Yang, H. H., & Moody, J. (2000). Data visualization and feature selection: New algorithms for nongaussian data. In *Advances in neural information processing systems* (pp. 687-693).
- [19] Peng, H., Long, F., & Ding, C. (2005). Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (8), 1226-1238.
- [20] Mesaros, A., Heittola, T., & Virtanen, T. (2016). TUT database for acoustic scene classification and sound event detection. In *2016 IEEE 24th European Signal Processing Conference (EUSIPCO)* (pp. 1128-1132).

- [21] Virtanen, T., Plumbley, M. D., & Ellis, D. (Eds.). (2018). *Computational analysis of sound scenes and events*. Springer International Publishing.
- [22] Manolakis, D. G., & Ingle, V. K. (2011). *Applied digital signal processing: theory and practice*. Cambridge University Press.
- [23] Gold, B., Morgan, N., & Ellis, D. (2011). *Speech and audio signal processing: processing and perception of speech and music*. John Wiley & Sons.
- [24] Babae, E., Anuar, N. B., Abdul Wahab, A. W., Shamshirband, S., & Chronopoulos, A. T. (2017). An overview of audio event detection methods from feature extraction to classification. *Applied Artificial Intelligence*, 31(9-10), 661-714.
- [25] Michael Peitler .(2016). Acoustic event detection of general sounds , Graz University of Technology
- [26] Azencott, C. A. (2018). *Introduction au Machine Learning*. Dunod.
- [27] Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern classification*. Second edition. N'ju-Jork.
- [28] Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.
- [29] Kuncheva, L. I. (2004). *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons.
- [30] Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7), 1895-1923.
- [31] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan), 1-30.
- [32] Garcia, S., & Herrera, F. (2008). An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9(Dec), 2677-2694.
- [33] García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10), 2044-2064.
- [34] Japkowicz, N., & Shah, M. (2011). *Evaluating learning algorithms: a classification perspective*. Cambridge University Press.

- [35] Avidan, S. (2001). Support vector tracking. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001* (Vol. 1, pp. I-I).
- [36] Chu, F., & Wang, L. (2005). Applications of support vector machines to cancer classification with microarray data. *International journal of neural systems*, 15(06), 475-484.
- [37] Alpaydin, E. (2009). *Introduction to machine learning*. MIT press.
- [38] Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1), 119-139.
- [39] Sehgal, A., & Kehtarnavaz, N. (2018). A convolutional neural network smartphone app for real-time voice activity detection. *IEEE Access*, 6, 9017-9026.
- [40] Vemula, H. C. (2018). *Multiple Drone Detection and Acoustic Scene Classification with Deep Learning* (Doctoral dissertation, Wright State University).
- [41] El Akadi, A., El Ouardighi, A., & Aboutajdine, D. (2008). A powerful feature selection approach based on mutual information. *International Journal of Computer Science and Network Security*, 8(4), 116.
- [42] Cheng, G., Qin, Z., Feng, C., Wang, Y., & Li, F. (2011). Conditional Mutual Information-Based Feature Selection Analyzing for Synergy and Redundancy. *Etri Journal*, 33(2), 210-218.
- [43] Peng, H., & Long, F. (2004). An efficient max-dependency algorithm for gene selection. In *36th Symposium on the interface: Computational Biology and Bioinformatics* (Vol. 57, p. 65).
- [44] Meyer, P. E., Schretter, C., & Bontempi, G. (2008). Information-theoretic feature selection in microarray data using variable complementarity. *IEEE Journal of Selected Topics in Signal Processing*, 2(3), 261-274.
- [45] Gencoglu, O., Virtanen, T., & Huttunen, H. (2014). Recognition of acoustic events using deep neural networks. In *2014 22nd European Signal Processing Conference (EUSIPCO)*(pp. 506-510).
- [46] Mesaros, A., Heittola, T., & Virtanen, T. (2016). Metrics for polyphonic sound event detection. *Applied Sciences*, 6(6), 162.

- [47] Giannoulis, D., Benetos, E., Stowell, D., Rossignol, M., Lagrange, M., & Plumbley, M. D. (2013). Detection and classification of acoustic scenes and events: An IEEE AASP challenge. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* (pp. 1-4).
- [48] Temko, A., Nadeu, C., & Biel, J. I. (2007). Acoustic event detection: SVM-based system and evaluation setup in CLEAR'07. In *Multimodal Technologies for Perception of Humans* (pp. 354-363). Springer, Berlin, Heidelberg.
- [49] Lutz, M. (2010). *Programming Python: powerful object-oriented programming*. " O'Reilly Media, Inc."
- [50] Langtangen, H. P., & Langtangen, H. P. (2011). *A primer on scientific programming with Python* (Vol. 6). Berlin/Heidelberg: Springer.
- [51] McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference* (Vol. 8).
- [52] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct), 2825-2830.
- [53] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)* (pp. 265-283).
- [54] Chollet, F. (2018). *Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. MITP-Verlags GmbH & Co. KG.
- [55] Liu, H., & Motoda, H. (Eds.). (2007). *Computational methods of feature selection*. CRC Press.

