الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Master

Filière Électronique

Spécialité Instrumentation

présenté par

Benkhaled Fahim

Detection and Classification of Audio Events for Environmental Sound Analysis

Proposé par: Ykhlef Farid

Supervisé par: Ykhlef Farid et Ykhlef Fayçal

Année Universitaire 2018-2019

Acknowledgements

To quote Isaac Newton "If I have seen further it is by standing on the shoulders of Giants”

Many thanks to family and loved ones, teachers, workers and everyone who contributed to this field, none of this would have been possible without your shoulders.

ملخص: بدافع من الحاجة المتزايدة في الآونة الأخيرة للكشف والتصنيف الدقيق للأحداث و المحيط الصوتي، في العديد من المجالات المختلفة مثل الأمن ، وإكتشاف العيوب في الآلات ، والمنازل الذكية والمراقبة الصحية ، يهدف هذا العمل إلى تصميم نموذج للأحداث الصوتية المثيرة للإهتمام ، من خلال تطبيق الأساليب الشائعة المستخدمة بإستعمال الأدوات الموجودة في مجال معالجة الإشارات لاستخراج المعالم ومجال التعلم الآلي للتدريب كذلك في ميدان الإحصائيات لتجهيز المعالم.

كلمات المفاتيح: كشف الحدث الصوتي ؛ إستخراج المعالم ؛ التعلم الالي ؛ معالجة الإشارات.

Résumé: Motivé par le besoin croissant récent de détection et de classification précises des événements sonores et de l'environnement, dans une multitude de domaines tels que la sécurité, la détection des pannes dans les machines, les maisons intelligentes et la surveillance de la santé, ce travail vise à modéliser les événements audio d'intérêt, en appliquant des méthodes communes en utilisant les boîtes à outils existantes dans les domaines du traitement du signal pour l'extraction de caractéristiques, de l'apprentissage automatique pour la formation et des statistiques pour le prétraitement des caractéristiques.

Mots clés: Détection d'évènements audio; Extraction des caractéristiques; Apprentissage automatique; Traitement de signal

Abstract: Motivated by the more recent growing need for accurate detection and classification of sound events and environment, in a plethora of domains like security, fault detection in machinery, smart homes and health monitoring, this work aims to model audio events of interest, by exerting common methods using existing toolboxes in the fields of signal processing for feature extraction, machine learning for the training and statistics for feature preprocessing.

Keywords: Audio event detection; Feature extraction; Machine learning; Signal processing.

List of acronyms and abbreviations

A

ADC, Analog to Digital Converter *7*

AI, Artificial Intelligence *2*

C

CSV, Comma Seperated Values *36*

CV, Cross Validation *31*

D

DCT, Discrete Cosine Transform *17, 44*

DFT, Discrete Fourier Transform *8, 14, 15, 17*

DSP, Digital Signal Processing *7*

E

etc *5, 14, 29, 37*

F

FFT, Fast Fourier Transform *15, 16*

FIR, Finite Impulse Response *8*

G

GMM, Gaussian Mixture Model *23*

H

Hz, Hertz *3, 4, 9, 16*

I

i.e, id est (in other words) *5, 8, 18, 23, 24, 33*

K

kHz, Kilo Hertz *3, 36*

K-NN, K-Nearest Neighbors *23*

L

log, Logarithm *14, 16, 17*

LPC, Linear Predictive Coding *14*

M

MCR, Mean Crossing Rate *13*

MFCC, Mel Frequency Cepcstrum Coefficient *44*

O

OVO, One Versus One *26*

OVR, One Versus the Rest *25*

R

RBF, Radial Basis Function *28, 47, 49*

RMS, Root Mean Square *13*

S

SER, Sound Event Recognition *9*

SVM, Support Vector Machine *20, 23, 24, 25, 27, 34, 40, 46, 47, 49*

Z

ZCR, Zero Crossing Rate *12, 13, 40, 44, 46*

Contents

List of Figures

List of Tables

Introduction

The world witnessed a rapid development in the consumer electronic devices in the last few decades. This affected the interaction of humans with the electronic devices in a major way. For instance, computers are not giant, bulky machines that are only used by the scientists for specific purposes anymore. They are smaller, more powerful, easier to use, and the vast amount of the world's population are using them daily for leisure or work purposes. The introduction of smartphones, pocket-size computers with phone capabilities, has led to an even tighter bond between the humans and the computers.

In this direction, one of the goals is to make devices that can recognize and understand the things and events happening around them without any user input, and then perform some operations based on their understanding.

Acoustic Event Detection (**AED**) is denoted as the recognition of any general individual sound events in audio, requiring estimation of onset. One of the senses that we use most while interacting within our physical context is hearing. Humans are very good at interpreting and assigning meanings to the sounds. On the other hand, computers still cannot offer reliable accuracy for this task. AED differs from another well-studied audio information retrieval task called Automatic Speech Recognition (**ASR**), in the sense that the aim is not to map the speech audio into words/phonemes, but to map non-speech audio to their corresponding semantic labels and offset for distinct sound event instances and identification of the sound

Our aim is to implement a system of audio detection and classification, the thesis work focuses on algorithms for both feature extraction and classification. The developed systems are tested and evaluated through existing common methods in recent literature. A large part

of the work has to be concerned with the problem of acoustic event classification, since detection relies heavily on classification.

This thesis will be structured as follows:

1. **Audio Features Extraction Techniques**
2. **Machine Learning**
3. **Fundamentals of Audio Event Detection**
4. **Implementation under Python**

We will end this dissertation by defining the various research perspectives that will be interesting to continue this line of work.

Chapter 1 Signal Analysis and Features Extraction

1.1 Introduction

For machine learning to be successful, we need a language to describe everyday things that is sufficiently powerful to capture the similarities and differences between them and yet is computationally easy to manage. It has also been studied in field psychology and philosophy that humans perceive the world around as attribute-value pairs. In early research in Artificial Intelligence (**AI**) this notion became a common way of representing knowledge, one can think of it as a spread sheet with columns representing attributes and cells as values.

We (humans), absorb all sorts of information around us on a daily basis, in many situations, discerning between different attributes is almost instinctive, without much effort or thought, for instance, one can easily tell the difference between let's say sound of water dripping and a car horn, because, as one grows up the brain develops to learn what the distinct differences between these two sounds are. To incorporate the same learning experience in a computer (so we can make predictions afterward), one must have an understanding of: sound, how humans perceive sounds and find out what creates the differences between sound A and B, in a quantifiable reliable manner.

In this chapter, we lay grounds for different audio features (attributes) and the methodology for extracting them as well as signal processing tools that make that a possibility.

1.2 Audio Signal Classification

To start off, we need to discuss the nature of sound, its taxonomy and human perception of sound.

1.2.1 Sound

From a physical viewpoint, it is a wave motion through a medium, this motion propagates by a series of condensations and rarefactions created within the transmitting medium [1].

From a subjective viewpoint, sound is a sensation produced through stimulation of the cochlea, specifically, the hair-like nerve cells, which generate electrical impulses that get transmitted to the brain, where it's decoded and perceived as sound [2].

Furthermore, sounds are divided into groups:

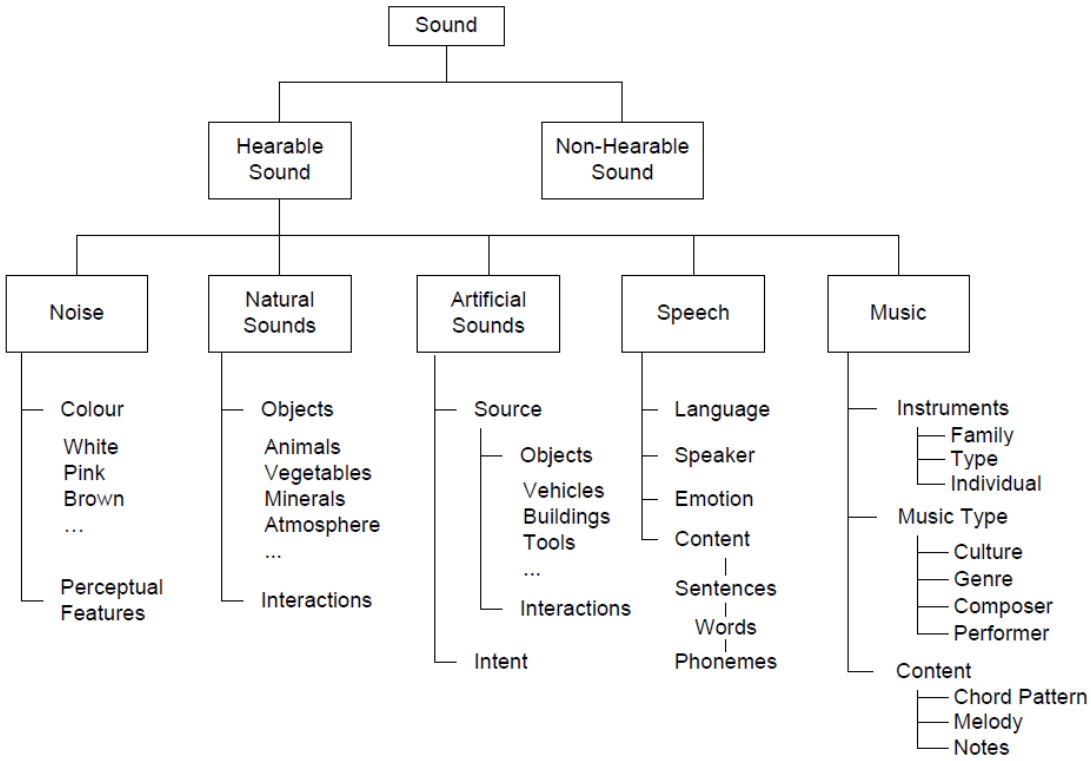

Figure 1 Taxonomy of sound [3]

- **Non-Hearable Sounds**

Consists of Ultrasound (greater than 20 kHz) and infrasound (lesser than 20 Hz) or sounds that are within the audible range, however too quiet to be perceived [4].

- **Hearable Sounds**

Sounds that lay within the audible frequency interval (20 Hz – 20 kHz). In addition they must be loud enough for that particular frequency **Figure *2*** shows this in more detail [4]**.**

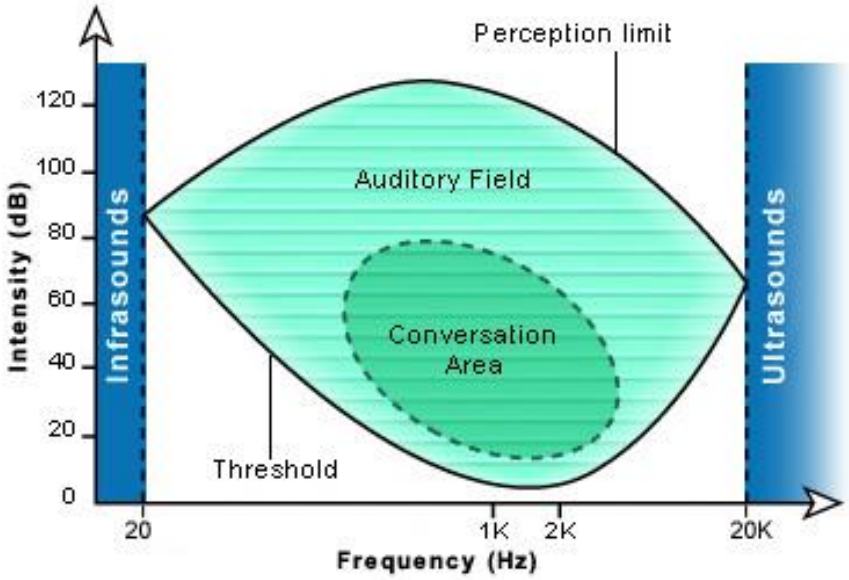

Figure 2 Human auditory field: Frequency-Intensity curves [4]

Hearable sounds are divided into subgroups, depending on the source that generates them:

a Noise

Noise is defined as any undesired signal that interferes with the information-bearing signal, there's several categories of noise a signal could be subjugated to, that would degrade the latter in quality, such as: acoustic, electromagnetic, electrostatic and processing noises [5].

In addition to that, depending on the frequency of the noise or time characteristics a noise can be arranged into several categories:

- White noise: a purely random uncorrelated noise process with a theoretical equal power at all frequencies [5].
- Coloured noise: non-white noise or any wideband noise characterized by a non-flat spectrum such as pink and brown noise. Pink noise for instance has a $\frac{1}{f}$ frequency dependence and a constant power per octave, brown noise on the other hand has a frequency distribution of $\frac{1}{f^2}$ [5] [6].
- Narrowband noise: a noise identifiable by its narrow bandwidth, most notable form is 50/60 Hz main's power [5].

b Natural Sounds

Natural sounds are the ones that are not generated by humans or created through human influence, they're caused by nature and the natural world that includes animals, the earth, the weather and water etc. They can be classified depending on the object that created the sound, however, it's also important to keep in mind that the objects can interact with and another making newer sounds [6].

c Artificial Sounds

We can view these as the opposite of what natural sounds are, i.e., human made sounds or human-influenced, excluding speech and music. The reason for not including the two in this category, is because they have multiple sub-classes that it's better to just put them into their own separate category. Artificial sounds, to give a few examples, are made by cars, buildings and machinery. In practice, the source of the sound is what we use for the classification feature, in addition we can include the intent of certain sounds, for instance, a sound of a telephone ringing or a siren signifies something [6].

d Speech

Speech can be described as human made sounds through the vocal tract for the purpose of communicating, including speech that's recorded or computer synthesized. There are several sub-categories that we can divide speech into, such as language, gender, emotional content, the subject matter, we can even go deeper and classify per word basis and phonemes.

e Music

Music can be defined as human made sounds using instruments, including the human body. There are multiple ways we can categorize music, like whether the music is made by one or many instruments (monophonic or polyphonic). Monophonic music (or polyphonic for that matter) then branches into family of the instrument(s) that is (are) being used (brass, string, percussion, voice, etc.) we can go further and classify by the type of instrument(s) (tuba, trombone, etc.). Then there's also sub-classes of content such as culture of origin, composer and genre. As with speech, we can also reduce our classification to most basic building blocks like notes.

1.3 Signal Representations

A lot of effort went into finding the proper representations in the field of acoustics, which permits the extraction of important information from audio signals.

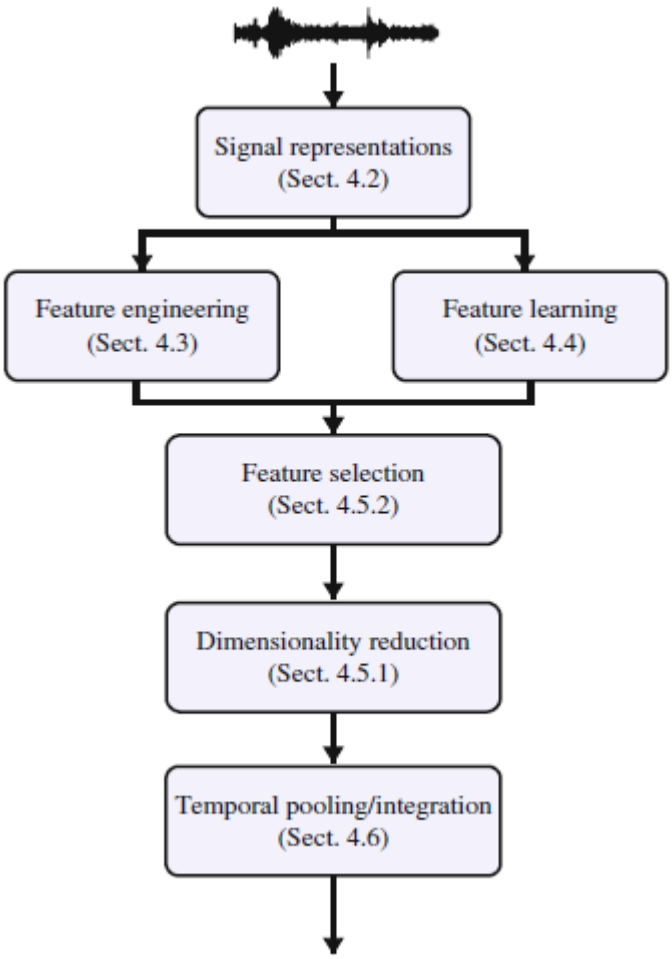

Figure 3 Standard feature extraction process *[7]*

1.3.1 Signal Acquisition and Preprocessing

First step towards event audio detection is capturing the actual audio signal data, storing and preprocessing it, this part focusses on these steps.

a Signal acquisition

In the following section, we define $\boldsymbol{a}$ as a continuous signal at time $\boldsymbol{t}$ to have an amplitude $\boldsymbol{a}(\boldsymbol{t})$. Signal $\boldsymbol{a}$ may represent for example the analog electrical current waveform created by a microphone's coil vibration.

So, to process the analog signal $\boldsymbol{a}$ we have to convert it to discrete time by sampling it, plus into discrete amplitudes through quantization. The reason we want discrete values is so

we can compute the signal, and processors can only represent values with limited precision, add to that the storage constraints.

The discretization in time is referred to as Nyquist-Shannon sampling. The signal $\boldsymbol{a(t)}$ is represented by a fixed amount of **N** values $\boldsymbol{a(n)}$ (samples) per unit of time. The sampling rate $\boldsymbol{f_s}$ is the frequency at which the original analog $\boldsymbol{a(t)}$ is sampled into, producing the $\boldsymbol{a(n)}$ values. $\boldsymbol{T_s}$ is the sampling period, the relation between the discrete time index $\boldsymbol{n}$ and continuous time $\boldsymbol{t}$ is as follows:

$$\boldsymbol{T_s = \frac{1}{f_s}} \qquad \text{Equation 1}$$

as:

$$\boldsymbol{t = nT_s} \qquad \text{Equation 2}$$

The sampling theorem ensures that original analog signal can be reconstructed from a finite set of $\boldsymbol{N}$ samples, the condition below must be met

$$\boldsymbol{f_s \geq 2f_h} \qquad \text{Equation 3}$$

where $\boldsymbol{f_h}$ is the highest frequency in $\boldsymbol{a(t)}$ also known as Nyquist frequency. In practice an analog low pass filter is applied to $\boldsymbol{a(t)}$ to ensure proper sampling of any type of input.

Subsequently, quantization is applied by converting the continuous values to discrete fixed amplitudes $\boldsymbol{x(n)}$ where each $\boldsymbol{a(n)}$ value is mapped to the nearest fixed value. These fixed values are specified by the Analog to Digital Converter (ADC) one characteristic of an ADC is its precision in bits, typically in Digital Signal Processing (DSP) b=16 or b=24 are used and quantized values are stored as integer or floating point format, as for music and speech analysis b=32 bits floating points then scaled to [-1;+1] range. The number of possible sample values is given by $\boldsymbol{2^b}$ [8] [9].

b Preprocessing

Before sound features are extracted, several preprocessing steps needs to be applied to the signal in time or frequency domain. To give a few examples:

- **Down-Mixing**: Conversion of multi-channel audio to a single channel (mono) in order to reduce redundant feature by linearly averaging all the channels signals into one $\boldsymbol{x_0}$, C is the number of channels:

$$\boldsymbol{x_0(n) = \frac{1}{C}\sum_{C=1}^{C} x_c(n)} \qquad \text{Equation 4}$$

- **Pre-emphasis**: Typically, consists of applying a high pass first order digital filter i.e. a finite Impulse response (**FIR**) filter, for the purpose of amplifying frequency bands which carry important information [7] [8].

$$\boldsymbol{y[n] = x[n] - \alpha x[n-1]} \qquad \text{Equation 5}$$

Other pre-processing steps can also be performed such as noise reduction and echo cancellation [8].

1.3.2 Time-Frequency Representations

Before any analysis, the signal is commonly converted to frequency domain using Discrete Fourier Transform (**DFT**) we assume $\boldsymbol{x(n)}$ to be periodic. The spectrum $\boldsymbol{X(m)}$ is defined by:

$$\boldsymbol{X(m) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)\, e^{\frac{-j2\pi nm}{N}}} \qquad \text{Equation 6}$$

This gives us a representation which we can plot on a linear frequency scale. There are other spectra representations which can have a non-linear magnitude or frequency scaling, or a combination of linear and nonlinear on either axes.

It is often desirable to search for information in specific frequency bands, which is the motivation behind the many scales that spawned, these scales vary in terms of: logarithmic or perceptual laws, the number of bands, the filters used and the overlap length between them [9] [7] [8].

One of the more widely used non-linear frequency scale is the mel-scale

- **Mel-scale**: It aims to mimic the psychological sensation of heights of harmonics

B is the mel-scale conversion from Hz to mels and its inverse B^{-1} are given by:

$$\boldsymbol{B(f) = 1125 \ln\left(1 + \frac{f}{700}\right)}$$ Equation 7

$$\boldsymbol{B^{-1}(b) = 700\left(\exp(\frac{1}{1125}) - 1\right)}$$ Equation 8

b is the frequency in mels

There are also other slight variations of B and B^{-1} expressions [7].

Other scales include:

- Bark-scale
- Gamma-tone filters
- Critical bands
- Constant-Q transform

1.4 Feature Extraction

The purpose of this step is to reduce the audio signal into a form that describes the more important information about the sound event. A measure of a good feature is the ability to discriminate between different classes with ease, while keeping the variation within the same class to a minimum, plus the resiliency to external effects such as noise. Commonly, frame based features are used. Extracted from sequential short-time windowed frames, usually **25—60 ms** in size.

Sound event classes differ from one to another, and each one have a more optimal set of features that describes them. For instance, in ASR applications it is common to rely mainly on just frame-based features, however for Sound Event Recognition (**SER**) uses, these frame-

based features are not enough on their own, so, it is common to add supplementary features to better capture the information [10].

1.4.1 Windowing

Let us consider a signal $\boldsymbol{x(n)}$ with $\boldsymbol{n} \in [\boldsymbol{0}, \boldsymbol{N}[$ divided into $\boldsymbol{K}$ short, overlapping frames $\boldsymbol{x_k(\hat{n})}$ ($\boldsymbol{k} \in [\boldsymbol{0} \dots \boldsymbol{K-1}[$) of $\boldsymbol{N_f}$ (frame size in samples) and $\hat{\boldsymbol{n}}$ is the discrete sample index, relative to a single frame $\hat{\boldsymbol{n}} \in [\boldsymbol{0} \dots \boldsymbol{N_f - 1}[$ the start index of the kth frame $\boldsymbol{n_{start,k}}$ in $\boldsymbol{x(n)}$ is described by the relation

$$\boldsymbol{n_{start,k} = k.N_f^T} \qquad \text{Equation 9}$$

$\boldsymbol{N_f^T}$is the frame period in samples, the end index is given as:

$$\boldsymbol{n_{start,k} = k.N_f^T + N_f} \qquad \text{Equation 10}$$

The overlap percentage defined as $\boldsymbol{O_f}$ where $\boldsymbol{L_f}$ is the frame size in seconds and $\boldsymbol{T_f}$is the frame period (also known as hop length or frame step) in seconds

$$\boldsymbol{O_f = \frac{L_f - T_f}{L_f}} \qquad \text{Equation 11}$$

Overlapping is used to maintain continuity within frames [11].

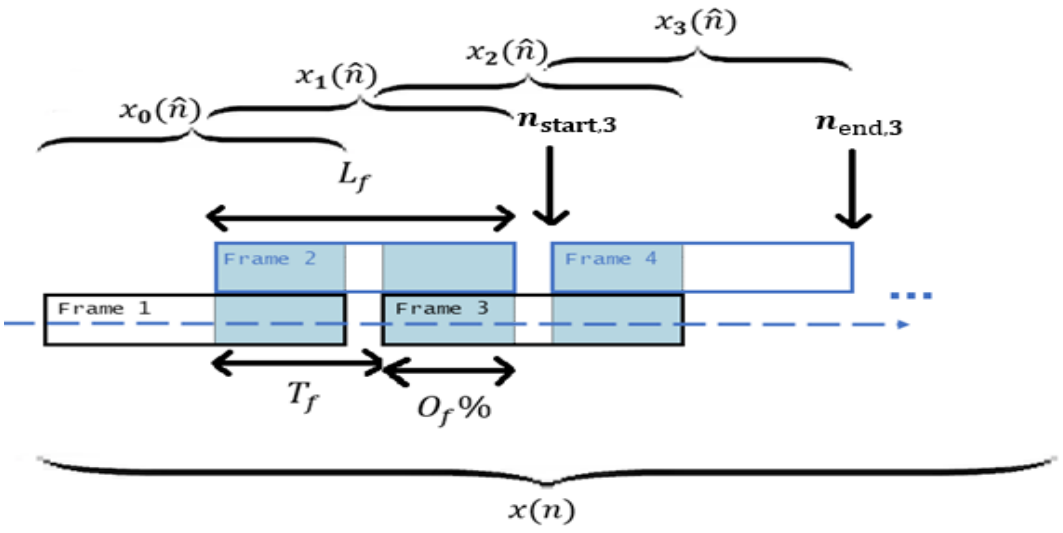

Figure 4 Representation of the framing process [12]

The excess frames towards the end of $x(n)$, are ignored, since we assume that the signal is more than just a few frames, the discarded samples won't contribute much to the final model [8].

1.4.2 Window Functions

Framing is simply put is a multiplication of $\boldsymbol{x(n)}$ with window function $\boldsymbol{w(n)}$ there are multiple window functions, their usage depends on the type of analysis performed later on (spectral or time) and the application at hand (recognition of speech, music or general sounds..) [8].

- **Rectangular window**:

  A constant function best suited for time analysis, efficient in computational power, defined as follows [8]:

$$\boldsymbol{w_r(n) = 1\ for\ n = 0 \ldots N-1} \qquad \text{Equation 12}$$

- **Hanning window**:

  Also known as Hann-window, it is defined for $\boldsymbol{n \in [0, N-1]}$ as:

$$\boldsymbol{w_{Han}(n) = 0.5\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right)} \qquad \text{Equation 13}$$

  This window is suitable for spectral analysis due to the fact that the side lobes in the spectrum roll off by around 18 dB per octave plus it is fitting for applications in which $\boldsymbol{x(n)}$ needs to be reconstructed from spectra [8].

- **Hamming window**:

  A modified version of a Hanning window, except that it doesn't reach zero at the edges. It is the most commonly used window in spectral analysis especially for speech, it reduces the amplitude of the first side lobe in the spectrum significantly, defined for $\boldsymbol{n \in [0, N-1]}$ [8].

$$w_{Ham}(n) = \alpha - \beta \cos\left(\frac{2\pi n}{N-1}\right)$$ Equation 14

Now there are different features that belong to different domains, as shown in the figure below

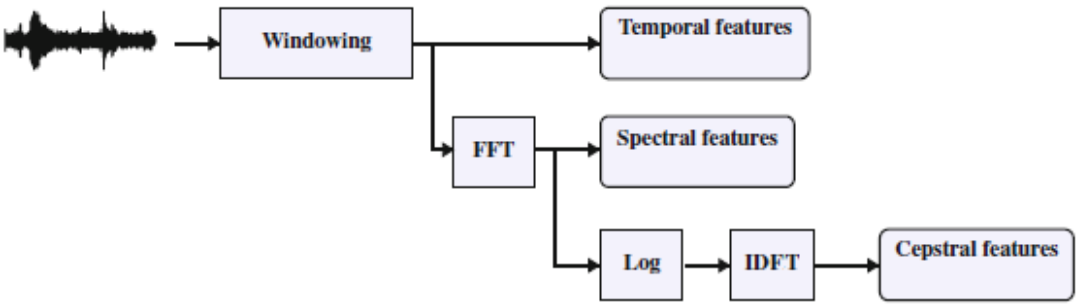

Figure 5 Simplified block diagram for extracting features in different domains *[7]*

In the following sections, we are introducing a variety of features in different domains

1.4.3 Time Domain Features

a Zero and mean crossing rate

Zero-Crossing Rate (ZCR) simply describes the number of sign changes of $x(n)$ per unit time

A sign change occurs whenever:

$$x(n-1)x(n) < 0$$ Equation 15

or

$$x(n-1)x(n+1) < 0 \text{ and } x(n) = 0$$ Equation 16

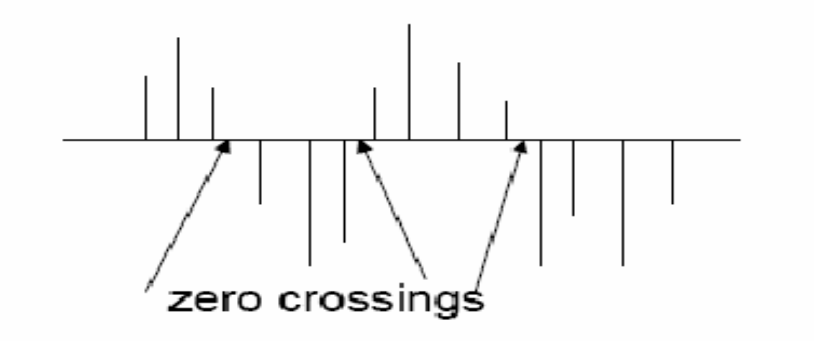

Figure 6: Zero crossings in a signal [13]

Mean Crossing Rate (MCR) is similar to ZCR in the sense that: it is the rate of changes between the mean value $\boldsymbol{\mu_x}$ of $\boldsymbol{x(n)}$ crossing from below to above or vice-versa [8].

with

$$\boldsymbol{\mu_x = \frac{1}{N}\sum_{n=0}^{N-1} x(n)}$$ Equation 17

A high ZCR or MCR implies a high frequency content, it's also important to note that the two are very sensitive to additive noise [8].

b Energy

One basic temporal feature is energy, assuming that the audio signal doesn't have a DC offset. The normalized form of energy E for the signal $x(n)$ is [8]:

$$\boldsymbol{E = \frac{1}{N}\sum_{n=0}^{N-1} x^2(n)}$$ Equation 18

There are also other forms for signal energy that can be used

- Root Mean Square (**RMS**) energy

$$\boldsymbol{E_{rms} = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1} x^2(n)}}$$ Equation 19

It is also a reliable tool for silence detection [7]**.**

- Log Energy

c Amplitude

For this temporal feature, maximum and minimum amplitudes are used [8].

***d* Autocorrelation coefficients**

We can view it as the signal spectral distribution in the time domain. Usually only the first K coefficients are kept [7].

$$R(k) = \frac{\sum_{n=0}^{N-k-1} x[n]x[n+k]}{\sqrt{\sum_{n=0}^{N-k-1} x^2[n]}\sqrt{\sum_{n=0}^{N-k-1} x^2[n+k]}}$$ Equation 20

k is the time delay

1.4.4 Spectral Features

Frequency domain features are extracted typically from the transformed time signal, typically a Fourier transform. Common spectral features are:

a Spectral envelope

Can be viewed as the boundary within which the spectrum is contained, we can approximate it using Linear Predictive Coding (**LPC**)

b Spectral moments

This one describes multiple characteristics of the spectral shape, like the spectral centroid, spectral width, spectral asymmetry and spectral flatness.

Other spectral features include:

- spectral slope
- spectral roll-off
- spectral flux
- alpha ratio

  etc.

1.4.5 Cepstral Features

Before we can delve into cepstral features we need to explain what a cepstrum is.

- **Cepstrum:** The cepstrum definition by Bogert, Healy, and Tukey, given as the inverse Fourier transform (inverse DFT in our case) of the log of magnitude spectrum of the signal, was first motivated by echo detection, then became widely used as a pitch indicator. Cepstral analysis consists of applying a homomorphic transformation turning a convolution into a sum, which allows the separation of the filter from the source, for instance we can separate the vocal tract transfer function from the excitation. The independent variable for the cepstrum is called quefrency, and the process of linear filtering in that domain is called liftering [14].

  The cepstrum c[n] is given by:

$$\boldsymbol{c[n] = DFT^{-1}\{\ \ log|\ DFT\{x[n]\}\ |\ \ \}} \qquad \text{Equation 21}$$

Cepstral features are among the most successful acoustic features in ASR, they have also been exploited in other domains such as acoustic geo-sensing, music mood recognition and acoustic gunshot detection. In the following section we're presenting the main cepstral features [8].

a) MFCC

MFCC are the most common cepstral coefficients, used for both speech analysis and sound scene analysis, it is inspired by the human auditory system. Coefficients are obtained through the following process:

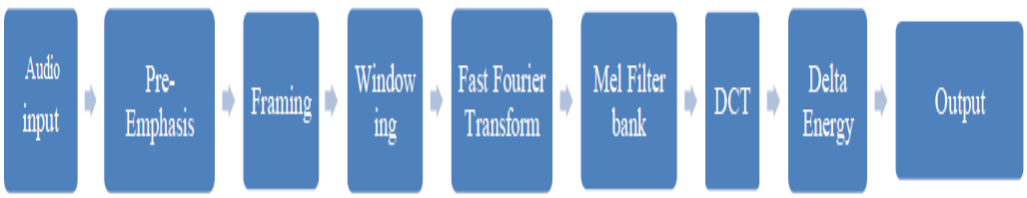

Figure 7 MFCC block diagram

Since we have already discussed the first 4 steps previously, we're only going to touch on the remaining steps

- Fast Fourier Transform (**FFT**): A more efficient algorithm to calculate the DFT, significantly reducing the calculation time, it converts the time domain signal to frequency domain. If a frame size is not the power of two the typical thing to do is to apply zero-padding, which adds samples of 0 values increasing the frame length to the next higher power of two [7].
- Mel Filter bank: 20-40 triangular bandpass filters with increasing bandwidths, and a 50% overlap are applied to the spectrum, this is motivated by the fact that: human perception of small frequency changes, is indiscernible. The effect gets magnified as we go higher in frequencies. The filter banks typically start at 20 Hz, each filter endpoint to the left and right matches the center of the two adjacent filters.

  This step serves to reduce the number of features and help smooth out the magnitude spectrum [8] [15].

  We define a filterbank of M filters (m=1,2...M), the triangular filters functions are given by:

  Where k is the discrete frequency and $f[m]$ represent the boundary points, which are uniformly spaced in the mel-scale, it is given by:

$$\boldsymbol{H_m[k]} = \begin{cases} \mathbf{0} & \boldsymbol{k < f[m-1]} \\ \dfrac{(\boldsymbol{k - f[m-1]})}{(\boldsymbol{f[m] - f[m-1]})} & \boldsymbol{f[m-1] \le k \le f[m]} \\ \dfrac{(\boldsymbol{f[m+1] - k})}{(\boldsymbol{f[m+1] - f[m]})} & \boldsymbol{f[m] \le k \le f[m+1]} \\ \mathbf{0} & \boldsymbol{k > f[m+1]} \end{cases} \quad \text{Equation 22}$$

$$\boldsymbol{f[m]} = \left(\frac{\boldsymbol{N}}{\boldsymbol{F_s}}\right) \boldsymbol{B^{-1}}\left(\boldsymbol{B(f_1)} + \boldsymbol{m}\frac{\boldsymbol{B(f_h) - B(f_1)}}{\boldsymbol{M+1}}\right) \quad \text{Equation 23}$$

f_l and f_h are the lowest and highest frequencies of the filterbank in Hz

F_s sampling frequency in Hz

N size of the FFT

B and B^{-1} are given by Equation **7** and Equation **8**

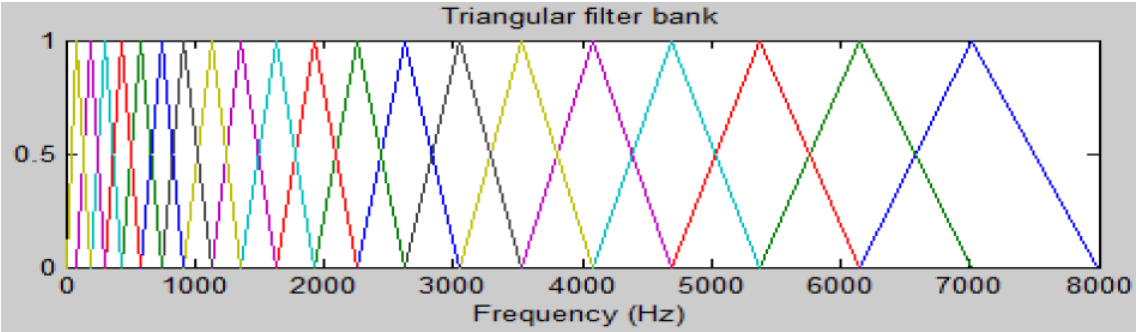

Figure 8 Mel filterbanks [15]

Then we sum the energies in each filter and take the log-energies (natural logarithm)

$$S[m] = ln \sum_{k=0}^{N-1} (|X[k]|^2 H_m[k]), \qquad 0 \leq m < M$$ Equation 24

- Discrete Cosine Transform (**DCT**): The transformation gives us Mel Frequency Cepstrum Coefficients, there are 26 coefficients, and we usually keep around 13 of the first coefficients. One reason for this step is to decorrelate the overlapping mel filterbanks [13].

Note that we mentioned previously that the cepstrum is obtained by applying DFT^{-1} to the log of the DFT of the signal, however we can alter this step by using DCT type II, or type III (also known as the inverse DCT) [8] [9]. DCT of the M filters is given by:

$$c[n] = \sum_{m=0}^{M-1} S[m] \cos\left(\frac{\pi n\left(m + \frac{1}{2}\right)}{M}\right) \qquad 0 \leq n < M$$ Equation 25

b) Delta and Delta-Delta MFCC

Delta values describes the change in MFCCs from one frame from to another while delta-delta is the change between frames of the delta values, these features are called dynamic cepstral features. The first derivative is commonly implemented as least-squares approximation. The delta coefficients d_t are given by:

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2}$$ Equation 26

The first row is filled with zeroes, c is the mel coefficient, t is the current frame index and N is typically 2

1.4.6 Statistical Functionals

In order to process segments with variable length and get rid of the dependency of the feature vector dimensionality on the segment length, statistical descriptors can be applied to the previously mentioned features, common descriptors are [8]:

a Mean

Describes the average of series of samples

$$\mu = \frac{1}{N}\sum_{n=0}^{N-1} s(n)$$ Equation 27

b Standard deviation

$$\sigma = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}(s(n) - \mu)^2}$$ Equation 28

c Maximum and Minimum

Describes the largest and smallest values in series of samples

$$S_{max} = max(s(n)) \quad \text{and} \quad S_{min} = min(s(n))$$ Equation 29

d Percentile

The j^{th} percentile P_j is defined as the value s below which j percent of all the values $s(n)$ are, i.e., for j percent of the values in $s(n)$ the following is true: $s(n) < P_j$.

The series $s(n)$ is sorted in ascending order, the j^{th} percentile can be found in the sorted set of values at the index n_j.

$$n_j = floor\left[\frac{j}{100}(N-1) + 0.5\right]$$ Equation 30

e Median

Median is the middle value in a sorted series.

1.5 Conclusion

So far, we have discussed sound categories and a variety of preprocessing and feature extraction techniques, this is a crucial first step in every acoustic detection system, this entire chapter boils down to collecting and summarizing audio data. Selectively stacking these features together makes for a robust AED.

Chapter 2 Basic Machine Learning Techniques

2.1 Introduction

The term machine learning refers to the automated detection of meaningful patterns in data. In the past couple decades, it has become a common tool in almost any task that requires information extraction from large data sets. We are surrounded by a machine learning based technology: search engines learn how to bring us the best results, anti-spam software learns to filter our email messages. Digital cameras learn to detect faces, as well as intelligent personal assistance applications on smart-phones can learn to recognize voice commands.

One common feature of all of these applications is that: in contrast to more traditional uses of computers, in these cases, due to the complexity of the patterns that needs to be detected, a human programmer cannot provide an explicit, fine detailed specification of how such tasks should be executed. Taking example from intelligent beings, many of our skills are acquired or refined through learning from our experience (rather than following explicit instructions given to us). Machine learning tools are concerned with endowing programs with the ability to “learn” and adapt.

Roughly speaking, learning is the process of converting experience into expertise or knowledge. The input to a learning algorithm is training data, representing experience, and the output is some expertise, which usually takes the form of another computer program that can perform some task.

The goal of this chapter is to go through the Machine learning landscape and lay out the basic concepts upfront for the chapter that follows. We will focus more on a single machine learning technique, successfully used in recent literature which is Support Vector Machine (SVM) [16].

2.2 Types of Machine Learning Algorithms

Machine learning algorithms are organized into taxonomy, based on the desired outcome of the algorithm. Common algorithm types include:

- Supervised learning where the algorithm generates a function that maps inputs to desired outputs. One standard formulation of the supervised learning task is the classification problem: the learner is required to learn (to approximate the behavior of) a function which maps a vector into one of several classes by looking at several input-output examples of the function.
- Unsupervised learning which models a set of inputs: labeled examples are not available.
- Semi-supervised learning which combines both labeled and unlabeled examples to generate an appropriate function or classifier.
- Reinforcement learning where the algorithm learns a policy of how to act given an observation of the world. Every action has some impact in the environment, and the environment provides feedback that guides the learning algorithm.
- Transduction similar to supervised learning, but does not explicitly construct a function: instead, tries to predict new outputs based on training inputs, training outputs, and new inputs.
- Learning to learn where the algorithm learns its own inductive bias based on previous experience [17].

2.2.1 Supervised Machine Learning

The majority of practical machine learning uses supervised learning. Supervised learning is where you have input variables X and an output variable Y and you use an algorithm to learn the mapping function from the input to the output.

$$\boldsymbol{Y} = \boldsymbol{f}(\boldsymbol{X}) \qquad \text{Equation 31}$$

The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (Y) for that data.

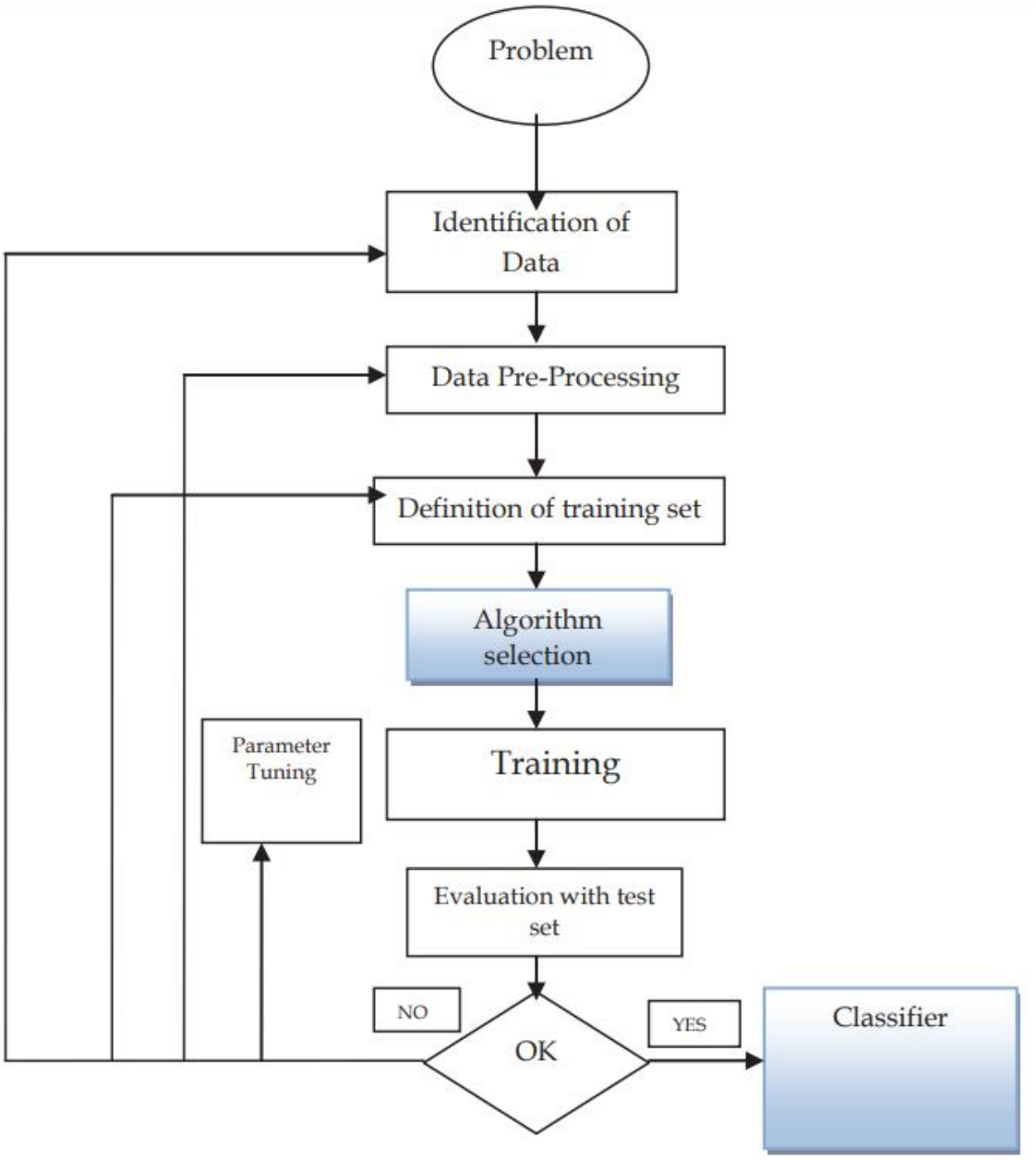

Figure 9 Process of supervised machine learning [17]

It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers, the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

2.2.2 Supervised Machine learning algorithm types

Supervised learning problems can be grouped into regression and classification problems.

- Classification: A classification problem is when the output variable is a category, such as red or blue or disease and no disease.
- Regression: A regression problem is when the output variable is a continuous value, such as dollars or weight.
- Some common types of problems built on top of classification and regression include recommendation and time series prediction respectively.

Some popular examples of supervised machine learning algorithms are:

- Linear regression for regression problems.
- Random forest for classification and regression problems.
- Support vector machines for classification problems.

2.2.3 Linear classifier

In machine learning, the goal of classification is to group items that have similar feature values, into groups. Timothy et al in 1998 stated that a linear classifier achieves this by making a classification decision based on the value of the linear combination of the features. If the input feature vector to the classifier is a real vector X, then the output score is:

$$y = f(\vec{w}.\vec{x}) = f\left(\sum_j w_j x_j\right)$$ Equation 32

where $\vec{w}$ is a real vector of weights and f is a function that converts the dot product of the two vectors into the desired output. The weight vector $\vec{w}$ is learned from a set of labelled training samples. Often f is a simple function that maps all values above a certain threshold to the first class and all other values to the second class. A more complex f might give the probability that an item belongs to a certain class.

For a two-class classification problem, one can visualize the operation of a linear classifier as splitting a high-dimensional input space with a hyperplane: all points on one side

of the hyperplane are classified as positive classes, while the others are classified as negative classes. A linear classifier is often used in situations where the speed of classification is an issue, since it is often the fastest classifier, especially when $\vec{x}$ is sparse. However, decision trees can be faster. Also, linear classifiers often work very well when the number of dimensions in $\vec{x}$ is large, as in document classification, where each element in $\vec{x}$ is typically the number of counts of a word in a document (see document-term matrix). In such cases, the classifier should be well regularized.

There exists multiple classifiers, to mention a few:

- K-Nearest Neighbor (**K-NN**)
- Gaussian Mixture Model (**GMM**)
- Random Forest
- Support Vector Machine(**SVM**)

Each classifier is more fitting depending on the task at hand.

2.2.4 Support Vector Machine

To give a little background, SVM was first introduced by Vapnik and Lerner in 1963, they proposed that the optimal hyperplane is the one that separates the training examples with the widest margin, from the 1960s to 1990s, Vapnik and Chervonenkis developed the Vapnik-Chervonenkis theory, which justifies the maximum margin principle from a statistical point of view. However, SVM really exploded in the 90's when it was proven to be incredibly accurate in recognizing handwritten digits [18].

A Support Vector Machine as stated by Luis et al (Luis Gonz, 2005) performs classification by constructing an N-dimensional hyper plane that optimally separates the data into two categories.

SVM models are a close cousin to classical multilayer perceptron neural networks. In fact, an SVM model using a sigmoid kernel function is equivalent to a two layer, perceptron neural network.

In the parlance of SVM literature, a predictor variable is called an attribute, and a transformed attribute that is used to define the hyperplane is called a feature. The task of

choosing the most suitable representation is known as feature selection. A set of features that describes one case (i.e., a row of predictor values) is called a vector. So the goal of SVM modelling is to find the optimal hyperplane that separates clusters of vector in such a way that cases with one category of the target variable are on one side of the plane and cases with the other category are on the other size of the plane. The vectors near the hyper plane are the support vectors. The figure below presents an overview of the SVM process.

a Two-Dimensional Hyperplanes

Before considering N-dimensional hyper planes, let's look at a simple 2-dimensional example. Assume we wish to perform a classification, and our data has a categorical target variable with two categories. Also assume that there are two predictor variables with continuous values. If we plot the data points using the value of one predictor on the X axis and the other on the Y axis we might end up with an image such as shown below. One category of the target variable is represented by rectangles while the other category is represented by ovals.

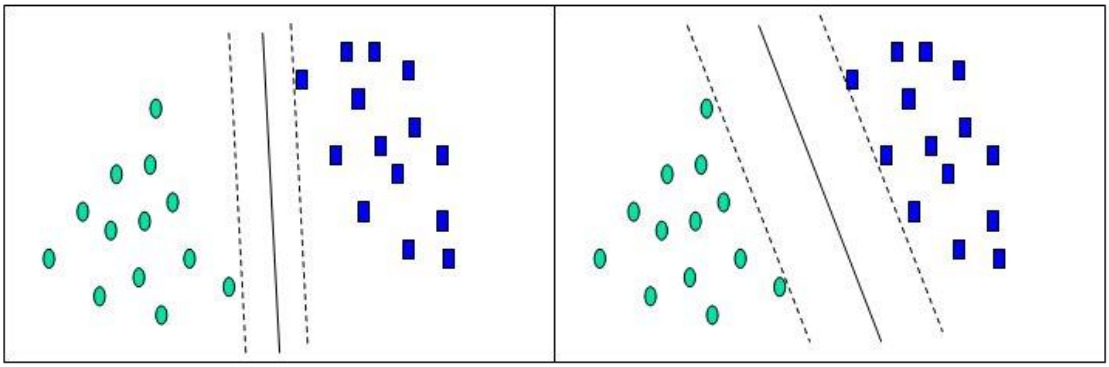

Figure 10 Variations of separating hyperplanes [17]

In this idealized example, the cases with one category are in the lower left corner and the cases with the other category are in the upper right corner; the cases are completely separated. The SVM analysis attempts to find a 1-dimensional hyper plane (i.e. a line) that separates the cases based on their target categories. There are an infinite number of possible lines; two candidate lines are shown above. The question is which line is better, and how do we define the optimal line. The dashed lines drawn parallel to the separating line mark the distance between the dividing line and the closest vectors to the line. The distance between

the dashed lines is called the margin. The vectors (points) that constrain the width of the margin are the support vectors. The following figure illustrates this.

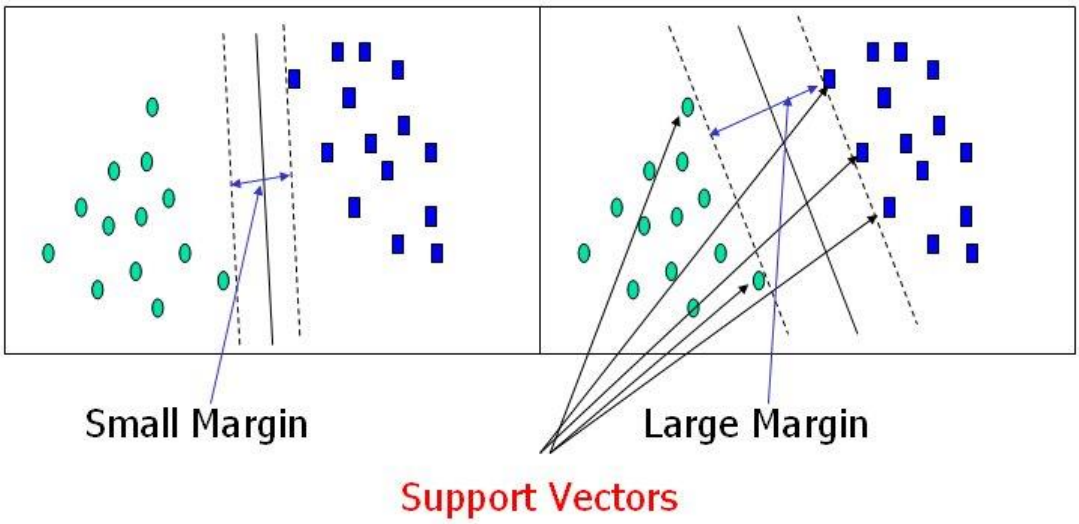

Figure 11 Difference of margin length between different hyperplanes [17]

An SVM analysis (Luis Gonz, 2005) finds the line (or, in general, hyperplane) that is oriented so that the margin between the support vectors is maximized. In the figure above, the line in the right panel is superior to the line in the left panel. If all analyses consisted of two-category target variables with two predictor variables, and the cluster of points could be divided by a straight line, life would be easy. Unfortunately, this is not generally the case, so SVM must deal with:

(a) More than two predictor variables.

(b) Separating the points with non-linear curves.

(c) Handling the cases where clusters cannot be completely separated.

(d) Handling classifications with more than two categories.

b Multi-Dimensional Hyperplane

The support vector machine is fundamentally a binary classifier. In practice, however, we often have to tackle problems involving $K > 2$ classes. Various methods have therefore been proposed for combining multiple two-class SVMs in order to build a multiclass classifier.

One commonly used approach (Vapnik, 1998) is to construct K separate SVMs, in which the k^{th} model is trained using the data from class C_k as the positive examples and the data from the remaining K − 1 classes as the negative examples. This is known as the **one-versus-the-rest (OVR)** approach.

Another approach is to train K(K−1)/2 different 2-class SVMs on all possible pairs of classes, and then to classify test points according to which class has the highest number of votes, such approach is called **one-versus-one (OVO)** [19].

2.2.5 Kernels

Kernel methods refer to a class of techniques that employ positive definite kernels. This mathematical trick is motivated by the need to separate between samples of different classes that regular linear separation doesn't handle very well, At an algorithmic level, its basic idea is quite intuitive: implicitly map objects to high-dimensional feature spaces and then directly specify the inner product there, in other words, it adds more dimensions and warps the shape of the data points so it becomes more separable [18].

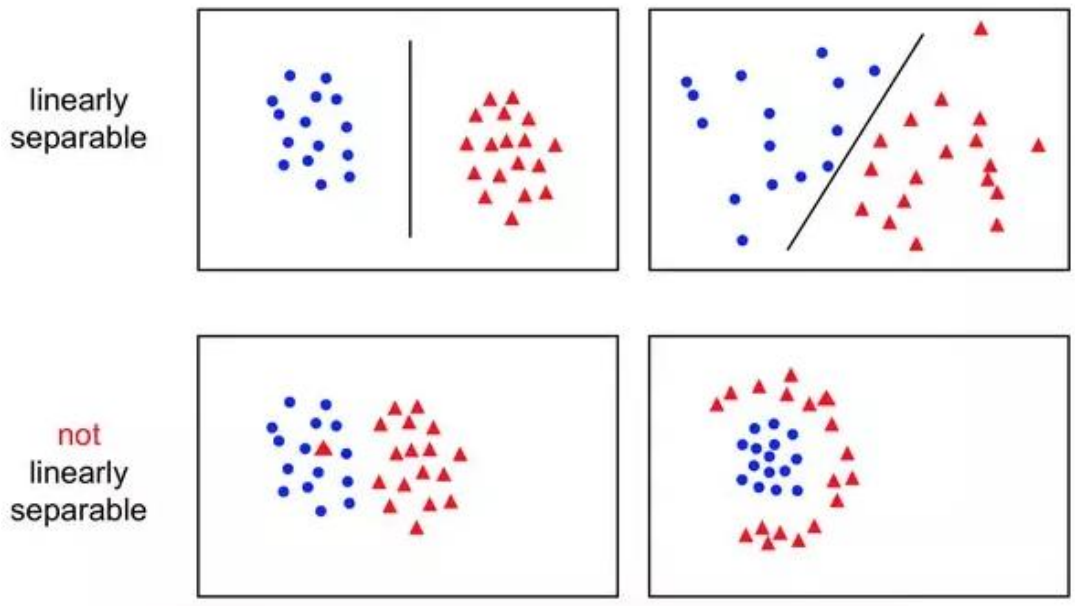

Figure 12 Demonstration of separable and non-separable classes [20]

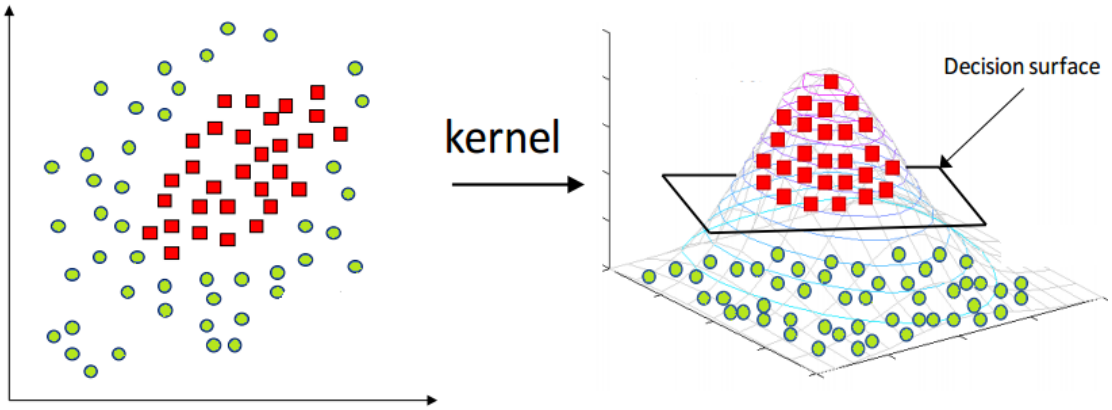

Figure 13 Visualization of a kernel trick [21]

2.2.6 Kernel types

Here's example representations of each kernel's separation of 3 different classes with an SVM classifier [22]:

- **Linear**

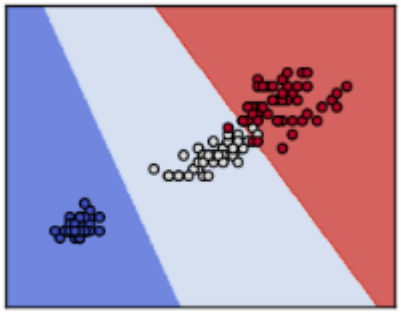

Figure 14 Linear kernel [23]

- **Polynomial**

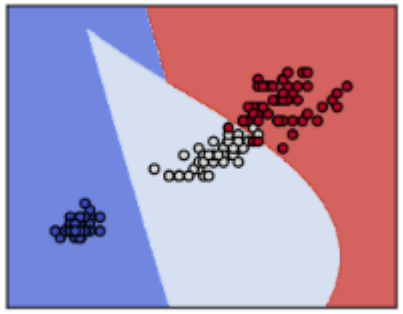

Figure 15 Polynomial (degree 3) kernel [23]

- **Radial Basis Function (RBF)**

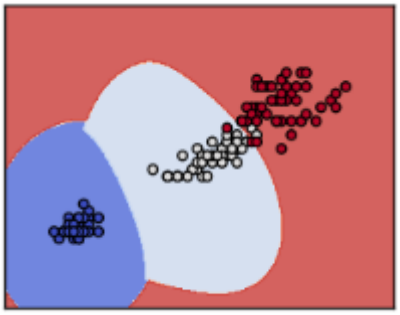

Figure 16 RBF kernel [23]

- **Sigmoid**
- etc.

2.2.7 Gamma and C Hyperparameters

A model can be adjusted to fit input data using certain parameters, which allows to tweak how far the influence of a single training example reaches (**Gamma**) as well as control underfitting and overfitting (**C** parameter). High gamma values means a single data point influence is small and constrained while low values, extends the influence farther, it is specific to RBF, sigmoid and polynomial kernels only. As for C also known as the penalty parameter, when it's larger the model overfits, and it underfits when C is smaller, so, it's important to find a balance.

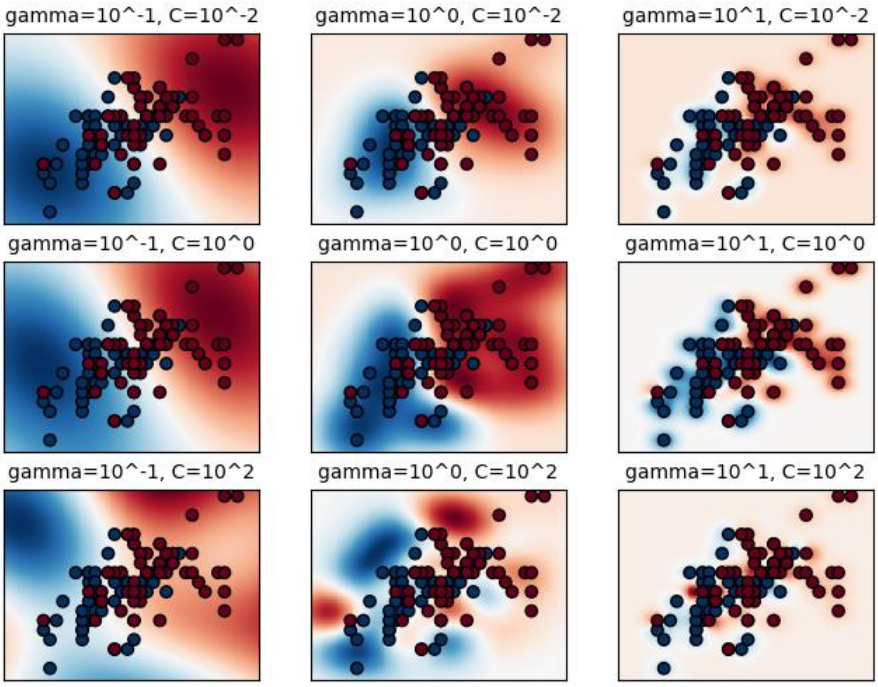

Figure 17: The impact of different gamma and C combinations on classification boundaries [24]

2.2.1 Curse of dimensionality

This rather emotive term is used to describe difficulties associated with the feasibility of density estimation in many dimensions. Adding a dimension stretches the points across that dimension making them further apart. In a multi-dimensional space, samples drift further apart when the dimensionality becomes too large, as a result, the process will be computationally expansive [25].

To help reduce the number of dimensions needed and bring down all features to the same level of magnitude, feature normalization and selection methods are utilized, the more common ones are as follows:

a Z-score Normalization (Standard Scaler)

Features are rescaled so that they'll have the properties of a standard normal distribution. Given by:

$$X_{norm} = \frac{x - \mu}{\sigma}$$ Equation 33

where μ and σ are the mean and standard deviation of a feature respectively [26].

b Min-Max Normalization

An alternative approach, is to scale data to a fixed range usually [-1,1] or [0,1] defined by the relation [27]:

$$X_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad \text{Equation 34}$$

a Variance Threshold

This feature selection method allows to only keep features with high variance discarding features below a specified threshold, since the ones with high variance contain more useful information [28].

2.2.2 Unsupervised Machine Learning

Unsupervised learning is where you only have input data X and no corresponding output variables. The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data. These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data. Unsupervised learning problems can be further grouped into clustering and association problems.

- Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- Association: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy A also tend to buy B.

Some popular examples of unsupervised learning algorithms are:

- k-means for clustering problems.
- Apriori algorithm for association rule learning problems.

2.3 Classification Evaluation

Once we have designed a classifier, we like to know how well the system can do the job or quantify the quality of the system performance.

The accuracy of a machine learning classification algorithm is one way to measure how often the algorithm classifies a data point correctly. Accuracy is the number of correctly predicted data points out of all the data points. Sometimes, this quantity is expressed as a percentage rather than a value between 0 and 1. The accuracy of a model is often assessed or estimated by applying it to test data for which the labels (Y values) are known. The accuracy of a classifier on test data may be calculated as number of correctly classified objects/total number of objects. Accuracy is directly related to error rate, such that:

$$\boldsymbol{Accuracy} = \mathbf{1} - \boldsymbol{Error\ Rate} \qquad \text{Equation 35}$$

2.3.1 Hold Out Method (Train Test Split)

If the data set at hand is large, we may divide it in two parts; use one for training and hold out the other for testing, typically, one third for testing and two thirds for training, hence the name hold-out method. This is a popular method to assess the system's performance. In most cases the data are limited in size; thus a hold-out method is ad-hoc in the sense of which subset is held out for testing. The performance evaluation via this method depends on how the data are separated.

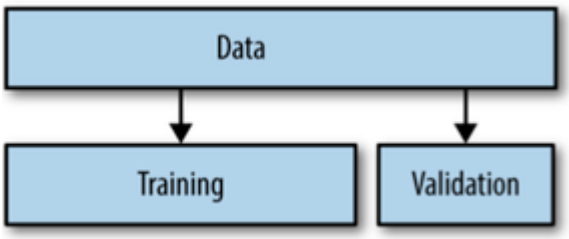

Figure 18: Train and test splitting

2.3.1 K-Fold Cross-validation (CV)

Cross-validation is a process for creating a distribution of pairs of training and test sets out of a single data set. In cross validation the data is partitioned into **k** subsets, $S_1 \ldots\ldots S_k$, each called a fold. The folds are usually of approximately the same size. The learning algorithm is then applied **k** times, for **i = 1** to **k**, each time using the union of all subsets other than S_i as the training set and using S_i as the test set [18].

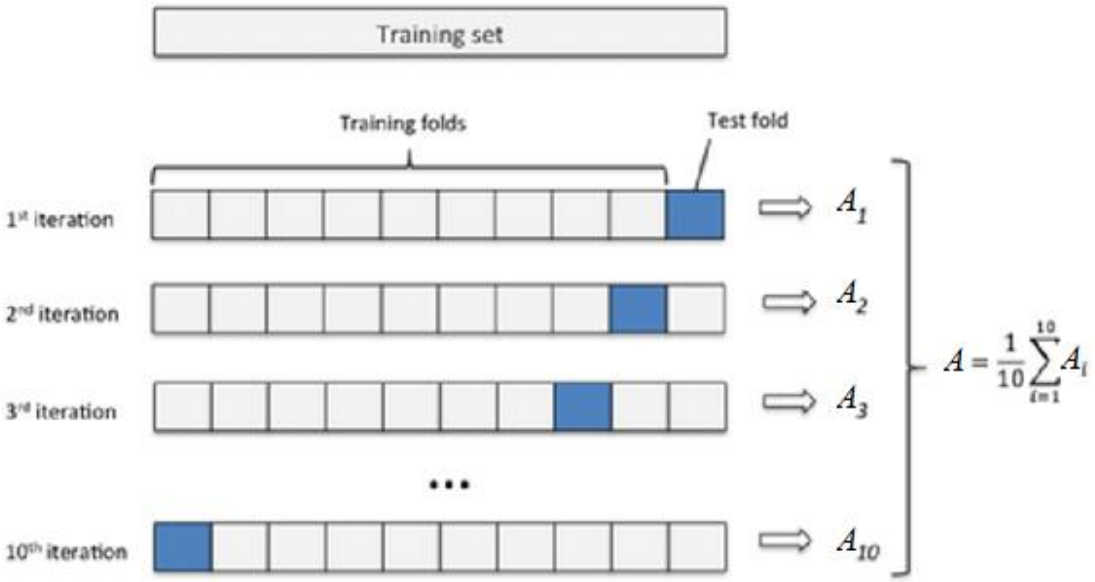

Figure 19 10 Fold cross-validation method [29]

where A_i is the accuracy for a particular iteration and A is the average accuracy.

2.3.2 Confusion matrix

A confusion matrix summarizes the classification performance of a classifier with respect to some test data. It is a two-dimensional matrix, indexed in one dimension by the true class of an object and in the other by the class that the classifier assigns. **Table *1*** presents an example of confusion matrix for a three-class classification task, with the classes A, B and C. The first row of the matrix indicates that 13 objects belong to the class A and that 10 are correctly classified as belonging to A, two misclassified as belonging to B and one as belonging to C.

Actual class	Assigned class			
		A	B	C
	A	10	2	1
	B	0	6	1
	C	0	3	8

Table 1: Example of a 3 class confusion matrix [18]

2.4 Audio Event Detection

Audio Event Detection or AED for short, is a relatively recent discipline, it consists of processing acoustic signals and converting them into symbolic descriptions corresponding to a listener's perception of the different sound events present in the signals and their sources i.e. The determination of both the identity of sounds and their position in time. One common approach consists of using a binary classifier, training it to audio events (positive class) and background noise (negative class), then, a sliding window on the audio signals is used for detection, this is known as detection-by-classification, this is the main reason why Audio Event Classification is so intertwined with AED [30] [31].

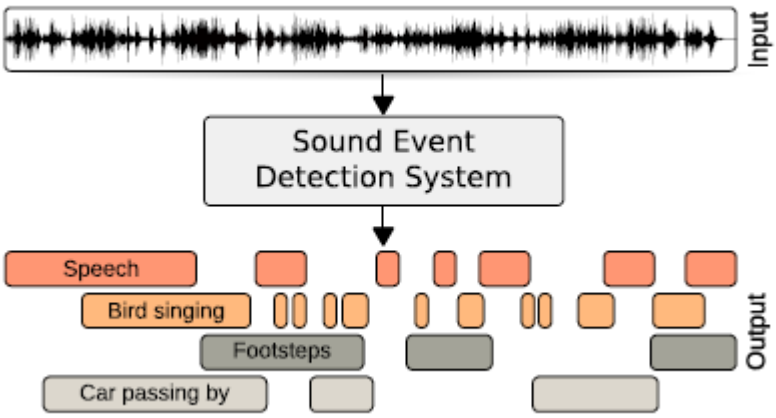

Figure 20 Sound event detection: finding temporal positions and textual labels [7]

2.5 Conclusion

To summarize, we discussed classification categories and existing algorithms focusing mainly on SVM combined with kernel tricks as well as dimensionality reduction and evaluation technics. This sets up the basis for Detection and Classification in conjunction with extracted features we've seen in the first chapter, allowing us to implement it under a computer.

Chapter 3 Detection and Classification of Audio Events : Experiment Results

3.1 Introduction

So far, we have discussed audio features and methods which allows to separate, classify and detect input data. In what follows we're tackling the procedures and results of audio events and scenes feature extraction, detection and classification by training extracted features with an SVM algorithm. ***Figure 21*** encapsulates the general plan and the way it was implemented under python.

.

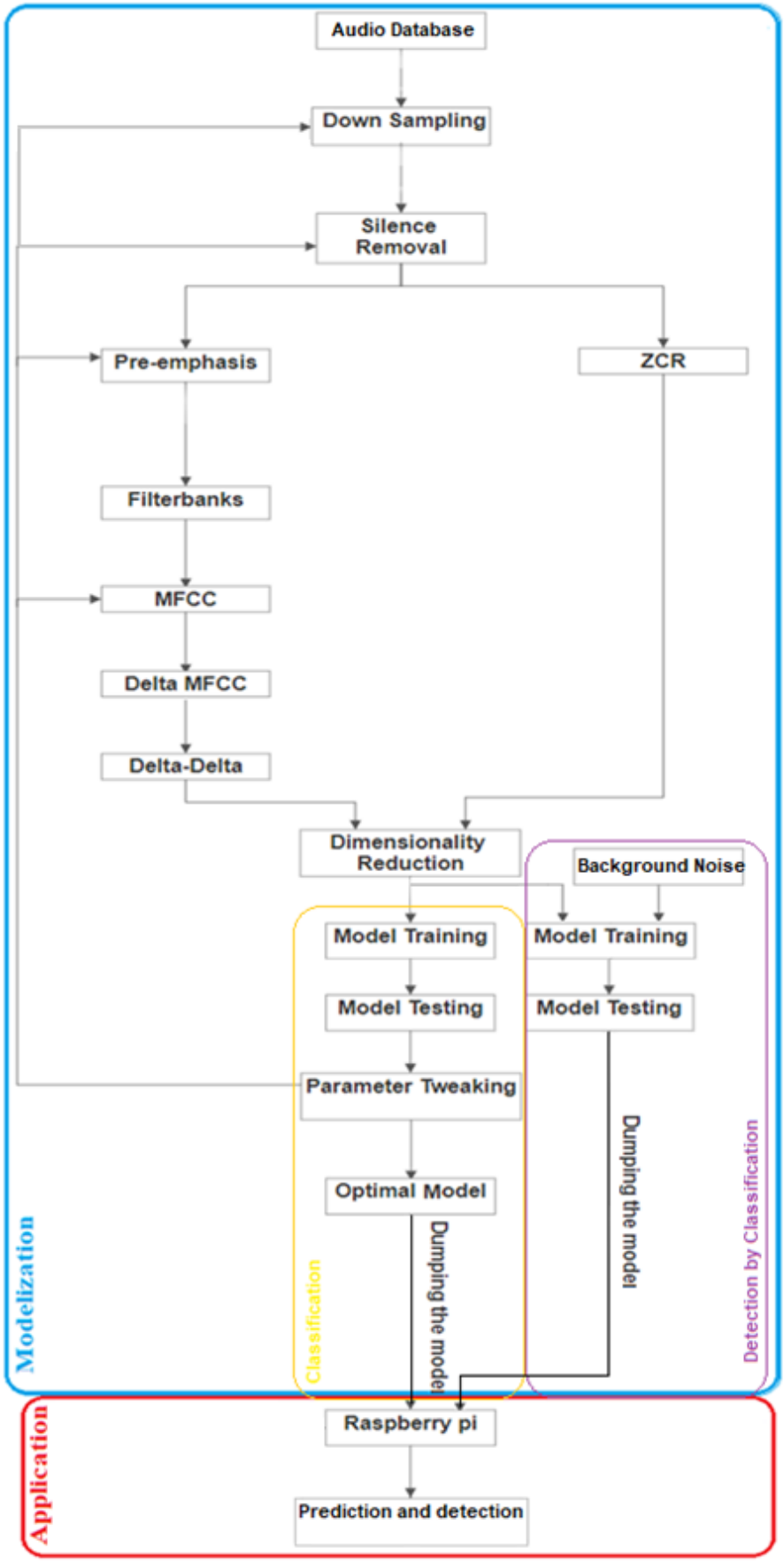

Figure 21 Explanatory diagram of classification and detection task

3.2 Audio Dataset (Database)

For the classification aspect, Environmental Sound Classification-50 database or ESC-50 for short, is used. It consists of 2000 labeled environmental recordings with 50 classes in total, each class is given 40 wav mono recordings lasting 5 seconds in length each, sampled at 44.1 kHz, containing a variety of common sounds such as: Human made sounds produced through speech or human activity, as well as natural sounds ranging from animal sounds to wind and rain.

For the application, we split this database in half, in order to save the processing time during training and feature extraction phases, this is due to the limited computation resources at hands.

The metadata is stored in sheet of a Comma Separated Values (**CSV**) file, which contains the name of each audio wav file as well as the corresponding label for it.

The reasons for picking this particular database are:

- The small size in terms of number of classes and audio files, hence a relatively shorter processing time (seconds to a few minutes)
- Decent variety of sound events
- Minimalistic background noise present in the recordings
- High sample rate, which gives better results when down-sampled

As for detection by classification, the entirety of previously mentioned database is used, in combination with background audio recorded using a crude microphone.

In order to have a balance between the positive and negative classes each recording lasts 5 seconds in length, the process is done 1000 times ie: about 83 minutes of background noise capturing in a relatively noisy environment that includes sounds such as: people talking, kids shouting, cars going by, etc. While also making sure none of the audio events we want to detect, exist in the 83 minute recording. ESC-50 is treated as a positive class and the microphone background as a negative class [32].

3.3 Preprocessing and Feature Extraction

3.3.1 Preprocessing

The audio files have silent portions, are unfiltered and have high sample rate, so, preprocessing them is the first step taken towards ensuring the data is filtered from unwanted extra information that do not contribute or degrade the model's performance. Preprocessing operations applied to the .wav files are:

a Silence trimming

As shown below in **Figure 22** the silent portions are removed entirely leaving only the information bearing parts, using trial on error approach to pick the optimal power threshold for silence detection, and individually inspecting wav file plots in audacity.

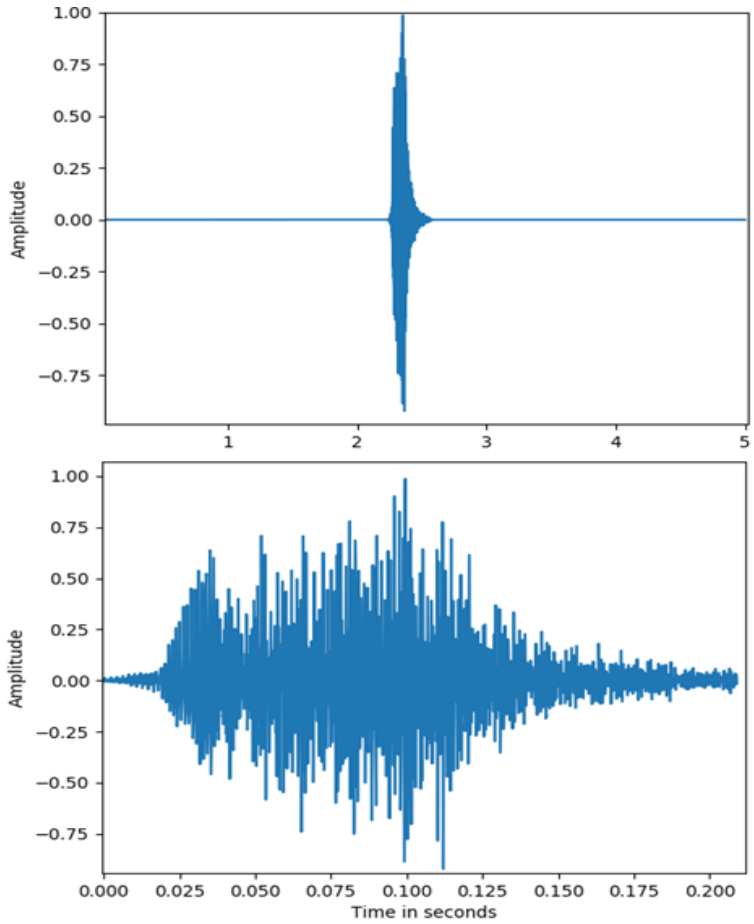

Figure 22: Before (top) and after (bottom) silence removal

b Pre-emphasis

Now, in order to balance the frequency spectrum between lower and higher frequencies, we apply a pre-emphasis filter to the audio files, with $\boldsymbol{\alpha = 0.97}$ the value again, was picked through a trial on error approach by testing out different values between 0.95 and 0.97 and keeping the one that gives out better accuracy during the testing. Result of the filtering is shown below.

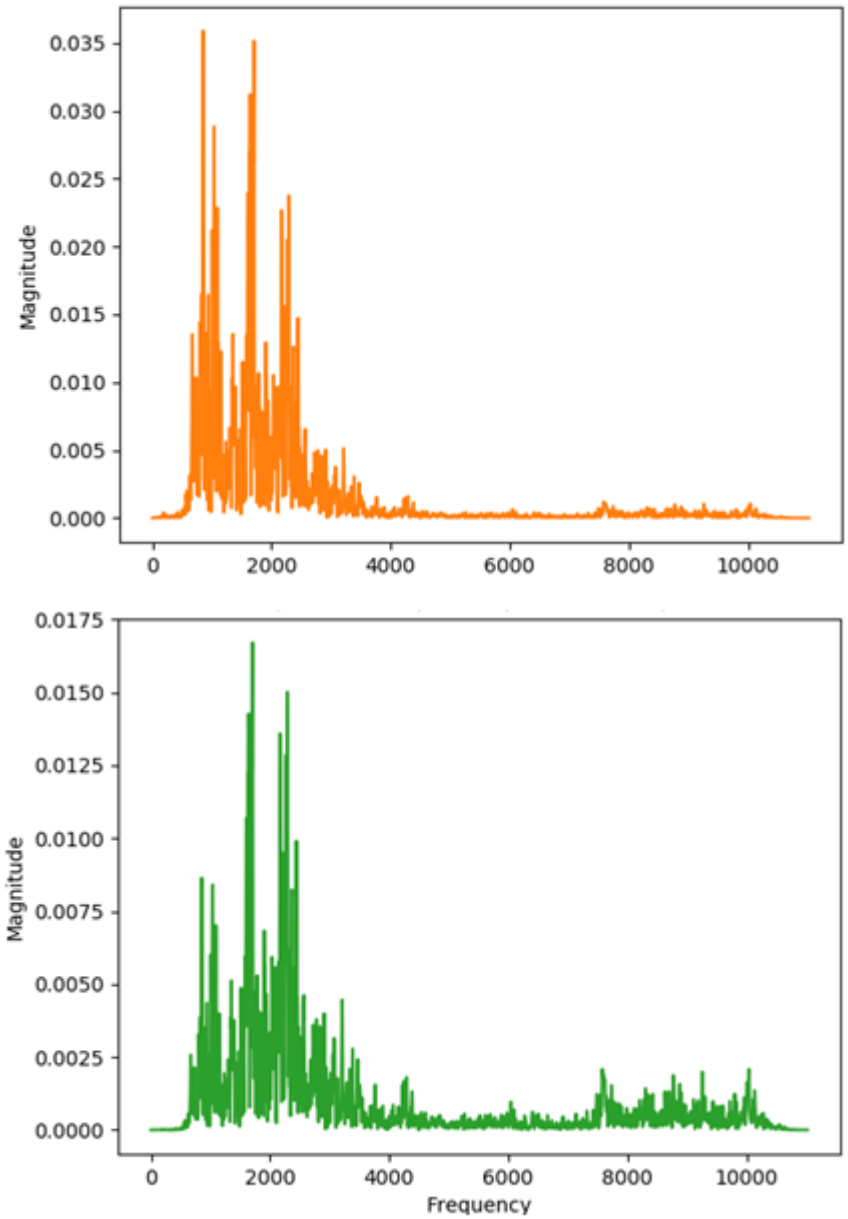

Figure 23: Spectrum before (top) and after (bottom) pre-emphasis

c Down-sampling

A high sampling frequency introduces more redundant information that's unnecessary, costly in processing and doesn't improve the model's final score, the figure below shows some of that information loss when signal gets down-sampled.

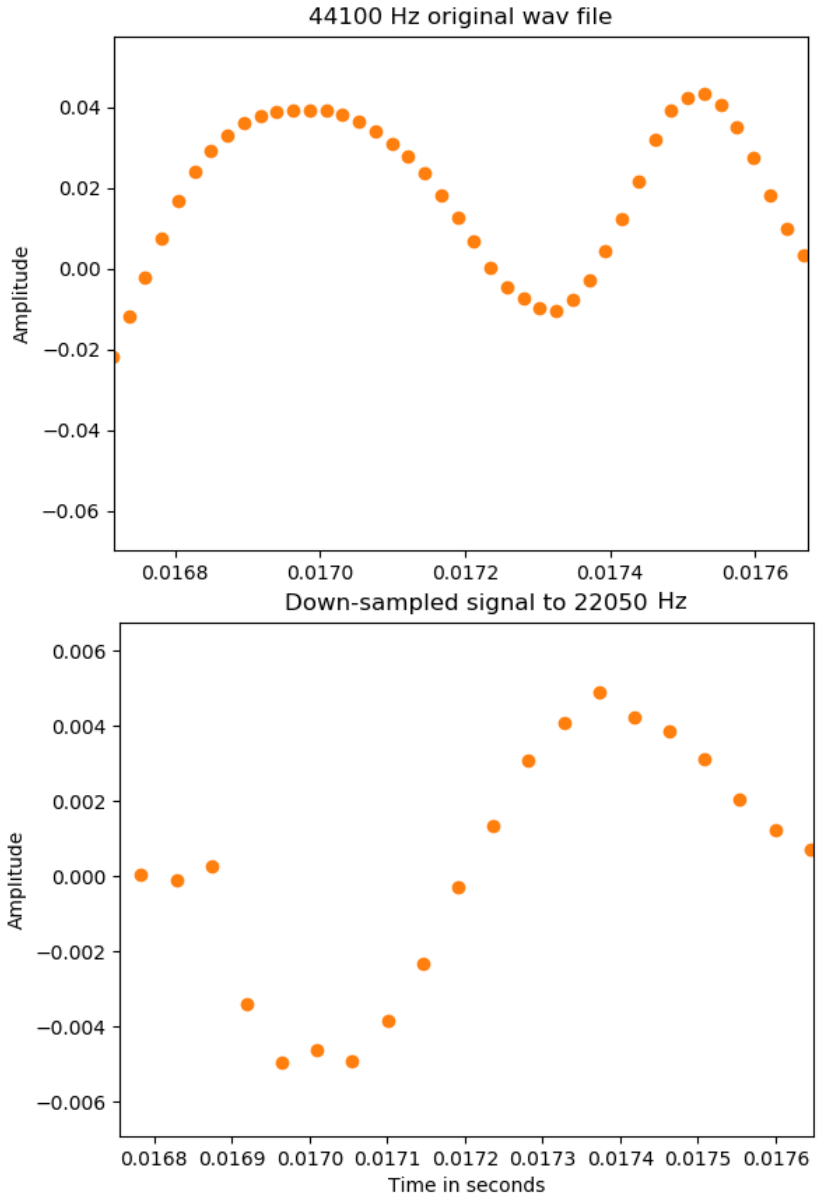

Figure 24 Effects of down-sampling the signal

3.3.2 Feature Extraction

Now that we have pre-processed the audio input we can move on to extracting features, the following features are used for training the model.

a Zero Crossing rating

This adds a time domain information, more precisely, how often the signal crosses x=0 axis in a single frame, the frame size taken is 25ms sample with a hop length of 10ms samples (default parameters). Modifying these two parameters to add more coefficients, doesn't seem to be beneficial whatsoever in terms of final accuracy, ZCR results is shown in the figure below.

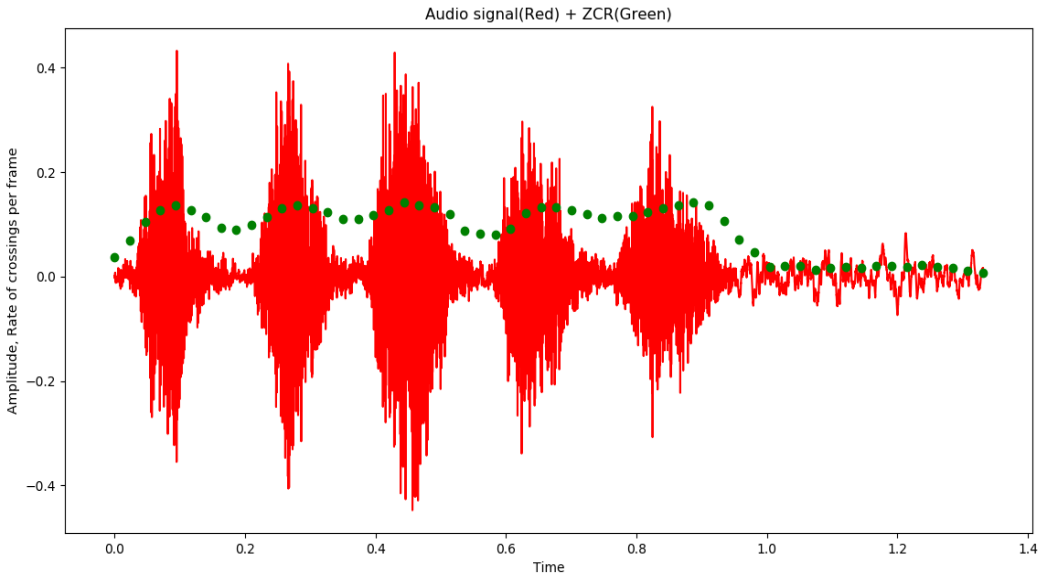

Figure 25: ZCR with respect to the audio signal

We can see that the higher the frequency content in a frame, the higher ZCR is, although there's variation from one audio file to another in terms of how obvious this correlation is.

b MFCC

MFCC is one of the more important feature that is relied upon in this application, which works better with an SVM classifier.

Python Speech Features library simplifies the process of obtaining the MFCC features, by bundling all the necessary steps and parameters tweaking into one function [33].

The first step is to frame the signal, initially the default built in parameters are used with the exception of using a hamming window instead of a rectangular one, with the frame size=25ms, and a step of 1ms which means a 60% overlap.

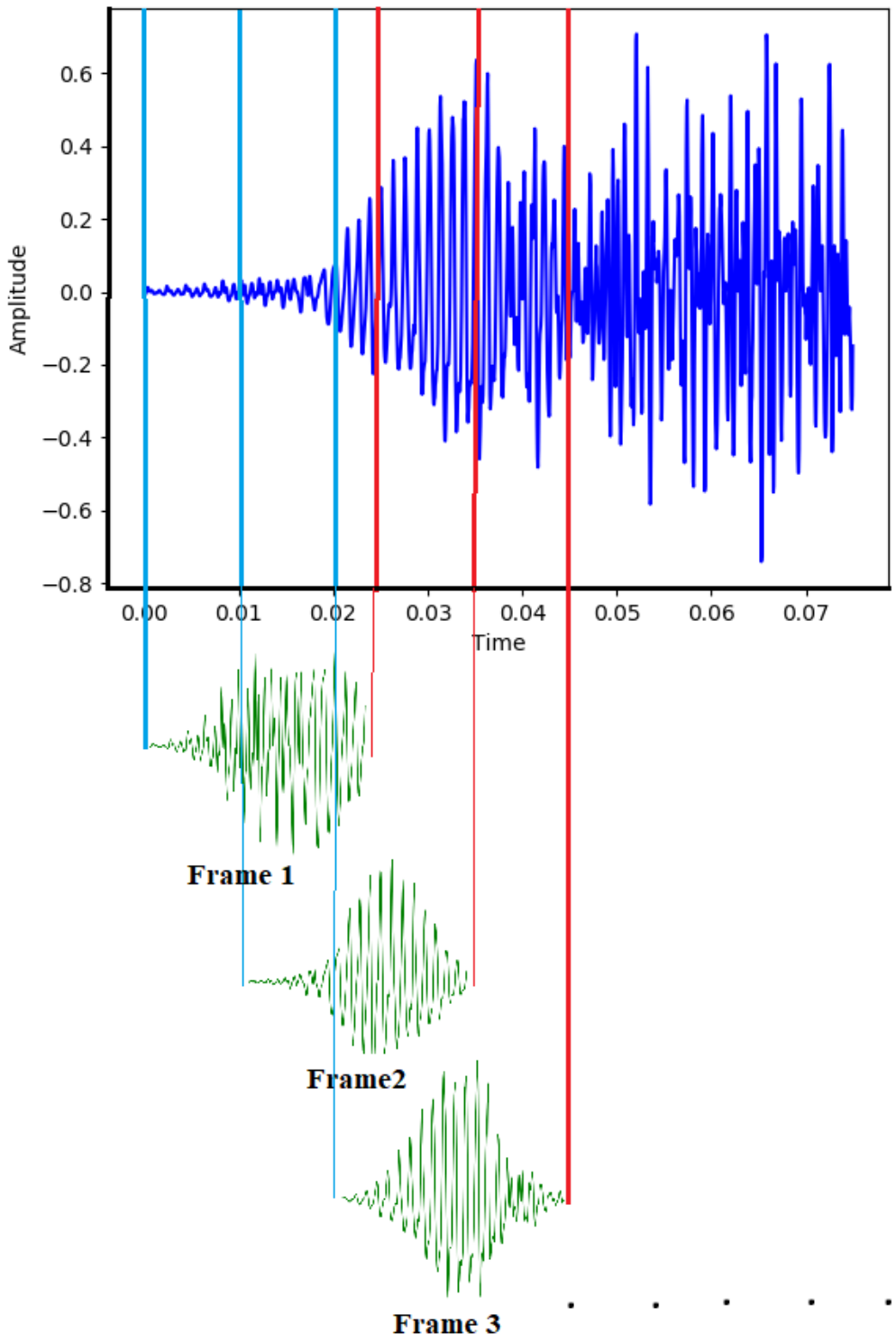

Figure 26: Visualization of framing process

Then the power spectrum for each frame is calculated

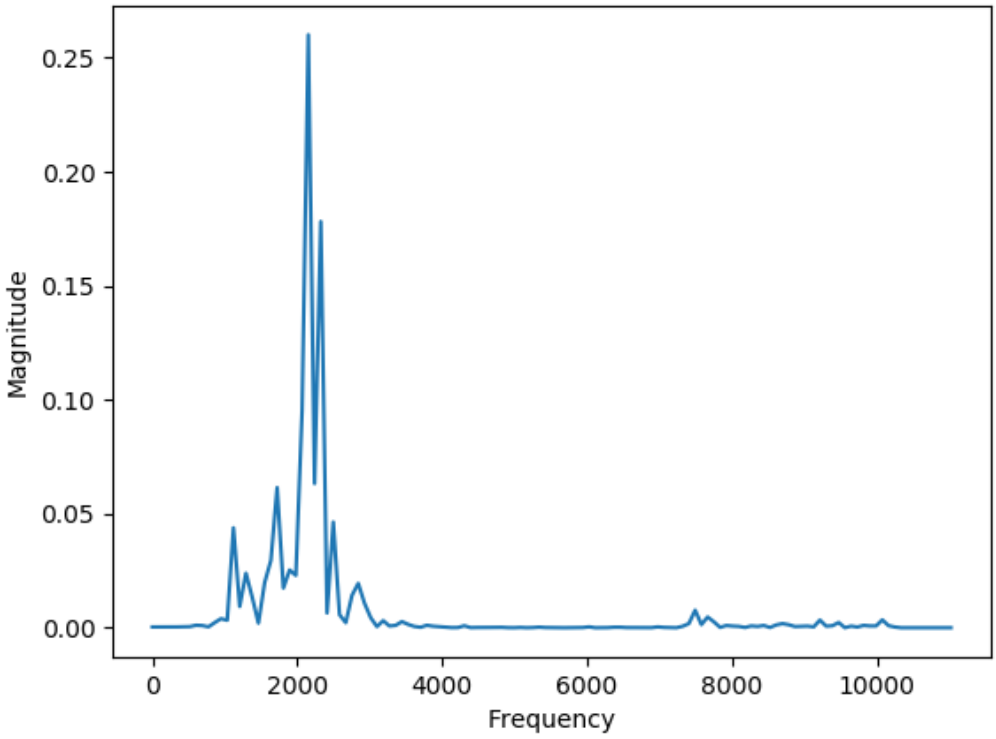

Figure 27: Power spectrum of a single frame

then we apply triangular filters to this power spectrum as such:

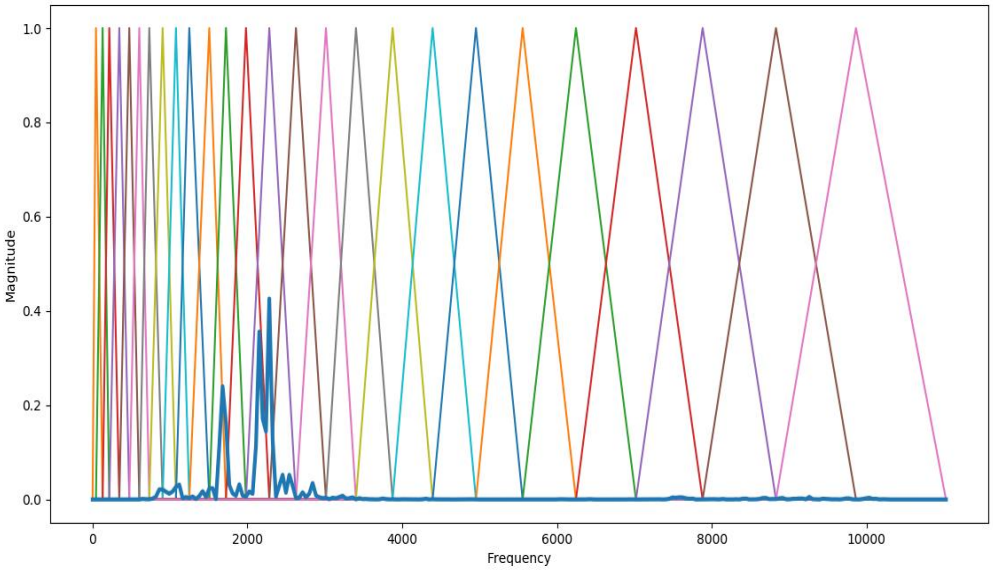

Figure 28: Mel filterbanks

Total energy in each frame is stored separately by summing the total energy, then the filterbank energies are obtained by a dot product of each mel filterbank with the power spectrum, and apply natural logarithm to the resulting filterbank energies and the total

energy. The number of mel filterbanks used is 26 by default, this results in a 2D matrix of size N_Frames x M_Filters each row holds a feature vector.

Then, we decorrelate these overlapping filterbanks by a DCT type-II transform, and finally lifter the matrix of cepstra increasing the magnitude of the high DCT coefficients. The resulting MFCC feature matrix is: first column is replaced with the log total energy, for our case, using 18 cepstrum coefficients is the most optimal case.

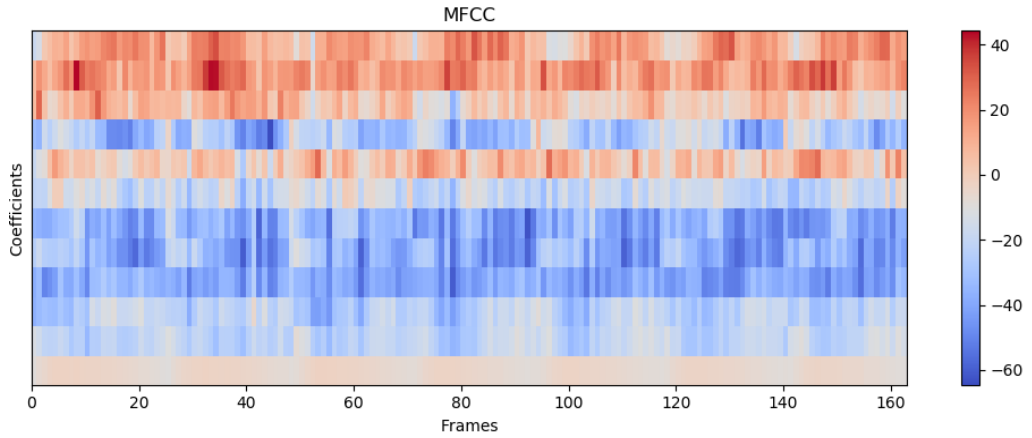

Figure 29: MFCC spectrogram

c Delta and Delta-Delta

18 delta features and delta-delta features each are computed from the MFCC feature vectors.

3.3.3 Dimensionality Reduction

At this point, there is a total of 54 features for each frame:

- 18 MFCC features
- 18 Delta MFCC features
- 18 Delta-Delta features
- 1 ZCR feature

This wouldn't be a problem except that each trimmed audio file has a lot of frames (up to a few hundreds), and each file is different from the other in the number of frames, so this creates two problems. The first, is that training with a large number of feature vectors is time consuming, the second issue is: we'll end up with certain classes having more training examples than other classes. In order to combat this, we employ two solutions.

a Statistical Descriptors

This reduces every feature for all frames into a single combination of 6 statistical descriptors:

- Mean
- Maximum
- Minimum
- Standard Deviation
- Skew
- Median

Then, for each audio file we stack these descriptors horizontally in a single vector as such:

Wav file	MEAN_MFCC_Coeff1	STDV_MFCC_Coeff1	SKW_MFCC_Coeff1	MAX_MFCC_Coeff1	MIN_MFCC_Coeff1	MED_MFCC_Coeff1	...	MEAN_ZCR	STDV_ZCR	SKW_ZCR	MAX_ZCR	MIN_ZCR	MED_ZCR
Dog1.wav	-1.2	-17.6	-43.5	2.1	-11.3	10.7	...	0.12	0.02	-1.3	0.1	0.05	0.1
Dog2.wav	-1.2	-17.6	-43.5	2.2	-11.3	10.7	.	0.11	0.02	-1.3	0.12	0.11	0.1
.....													
Church Bells40.wav	-2.4	-2.8	-29.7	-23.4	-16.2	-17.1	...	0.18	0.03	0.87	0.2	0.02	0.14

Table 2 Matrix of extracted features

This produces a total of 330 features for each audio sample.

b Min Max Scaling

In order to have a better performance in terms of training time and accuracy, we use this particular scaling transformation which scales each feature to a particular range [0,1] in this case.

c Variance Threshold

Leaving out redundant and feature vectors enhances processing time, by inspecting feature vectors variances and comparing how many features gets removed by that, allows us to select a variance threshold interval.

3.4 Model Training

SVM is the machine learning algorithm of choice for prediction and detection tasks, using the scikit-learn python library [34]. Recently, the Support Vector Machine (SVM) paradigm has proved highly successful in a number of classification tasks. As a classifier that discriminates the data by creating boundaries between classes rather than estimating class conditional densities, it may need considerably less data to perform accurate classification. This is one of the main reason the SVM classifier is initially chosen in this thesis as the main classification technique. Other advantages include:

- Versatile: different Kernel functions can be specified for the decision function.
- Effective in high dimensional spaces.
- Memory efficient.

Best gamma, C and kernel parameter combination is set using an exhaustive search over specified different values:

C and Gamma = [0.001, 0.01, 0.1, 1, 10]

Kernel = [rbf,linear] polynomial kernel was excluded due to very slow performance.

3.5 Classification Results

In this section, we're presenting the results of parameter tweaking and selection by comparing accuracy scores. A **10** fold cross validation is used to evaluate the average score each time, with the exception of the confusion matrix which is obtained by randomly splitting one third of the database for testing and two thirds for training. An SVM with a combination of RBF kernel are used for the classification for all results.

3.5.1 Downsampling

Sampling rate (kHz)	8	16	22.05	32	44.1
Accuracy %	79.3	83.8	81.9	81.3	78.2

Table 3: Accuracy with respect to sampling rate

As we can see, there is an increase in not only classification accuracy (to a certain point), but also time it takes to extract features when decreasing the sampling frequency. This is attributed to the fact that: lower sampling frequencies means less data points taken from the signal as a result less data to process add to that the time it takes to down-sample a signal. **44.1 kHz** was the fastest due to no down-sampling being done, **16 kHz** is the frequency of choice because it's faster and have gotten a boost in accuracy from it.

3.5.2 Silence Removal

Through experimentation and careful examination of **50** randomly picked waveform plots, **12-38 dB** interval showed that truncation of silent portion was more or less acceptable, **22 dB** threshold point seem to ensure no significant amount of information was lost and less noise present during the trimming process of our data, plus it yields one of the best classification results as shown below.

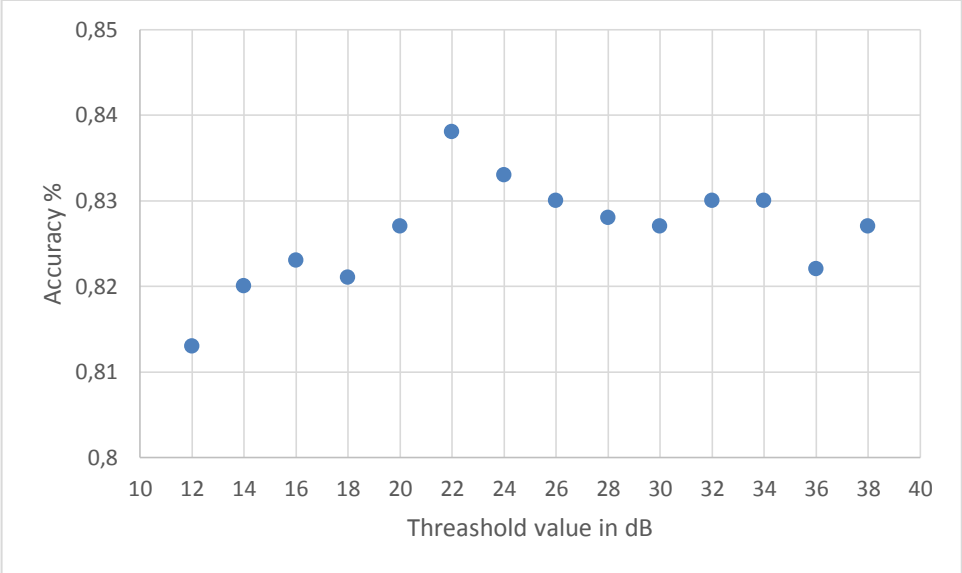

Figure 30: Accuracy with respect to silence threshold

3.5.3 Pre-emphasis

Varying Pre-emphasis coefficient made very little difference in terms of accuracy, only recommended values **0.95-1** were logged, a higher pre-emphasis coefficient implies a boost of influence of higher frequencies. The difference in classification accuracy we get is presented below:

Pre-emphasis coefficient	0.95	0.96	0.97	0.98	0.99	1
Accuracy %	83.7	83.6	83.8	83.6	83.4	83.2

Table 4:Accuracy with respect to pre-emphasis coefficient

3.5.4 MFCC, Delta and Delta-Delta Coefficients

These three are the backbone of all features used, so, it is no surprise to find out that adding more coefficients results in a substantial increase in accuracy of the classification, the curve follows an exponential growth then settles down at around **81 – 83** % accuracy

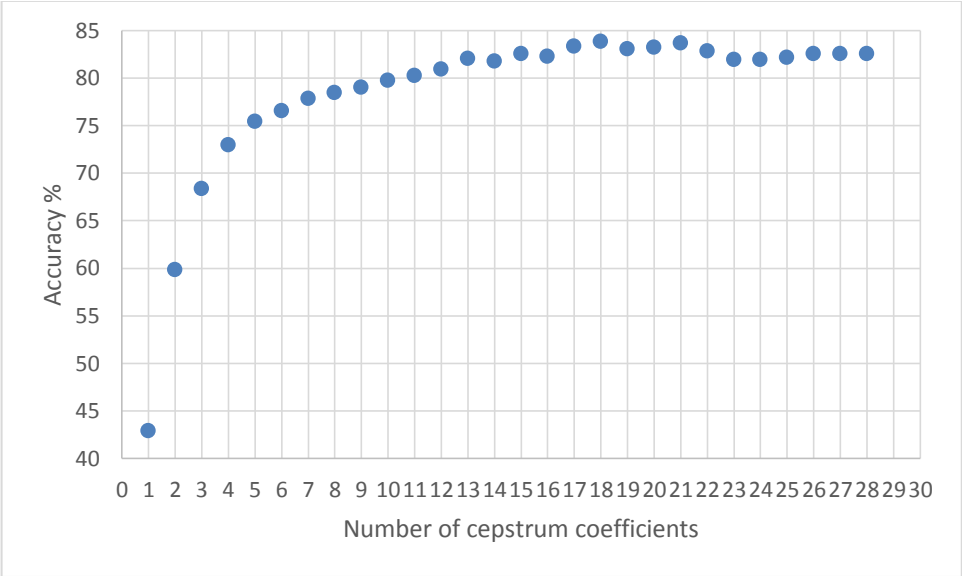

Figure 31: Accuracy with respect to number of coefficients

18 Coefficients are kept, there's very little improvement in accuracy when taking coefficients beyond that.

3.5.5 Feature Selection

Table below shows impact of removing features below different variance threshold points on accuracy, **0.16** gave out best results.

Variance Threshold	0.02	0.04	0.06	0.08	0.1	0.12	0.14	0.16	0.18	0.2
Accuracy %	83.3	83.1	82.5	82.9	83.4	83.2	83.7	84.1	83	82.8
Number of Features kept	254	244	227	214	203	197	189	183	175	171

Table 5: Accuracy with respect to different variance thresholds

3.5.6 SVM Kernels

Two kernel types were tested: linear and RBF. RBF gave the best results with a slightly higher training time than linear kernel, results of finding the optimal C and gamma values that fit each class are shown below.

- RBF kernel

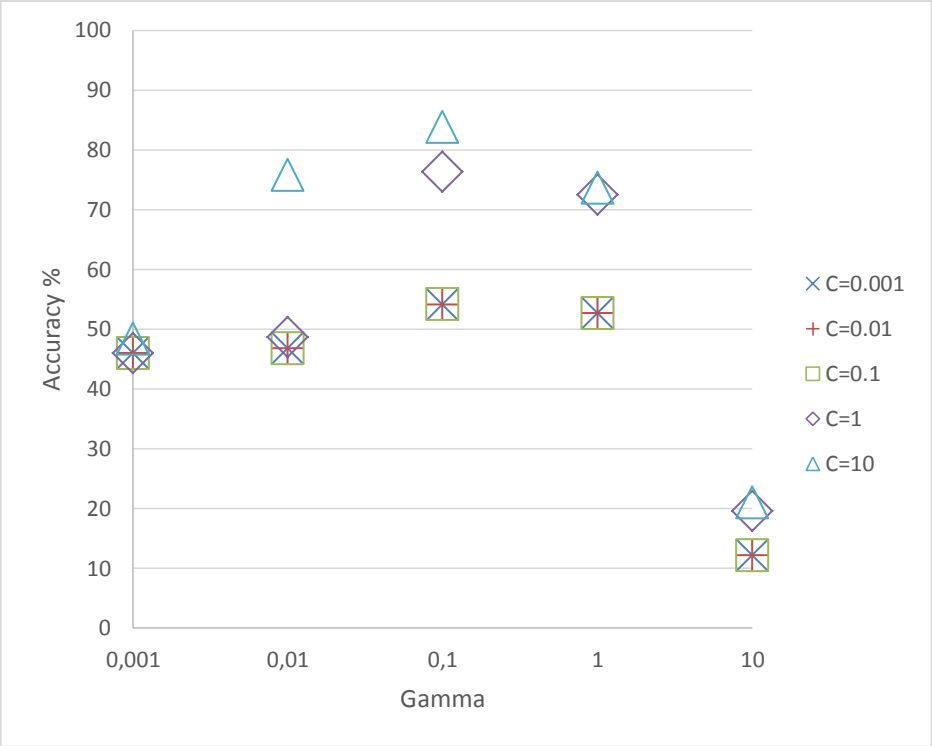

Figure 32: Accuracy with respect to C and Gamma

- Linear kernel

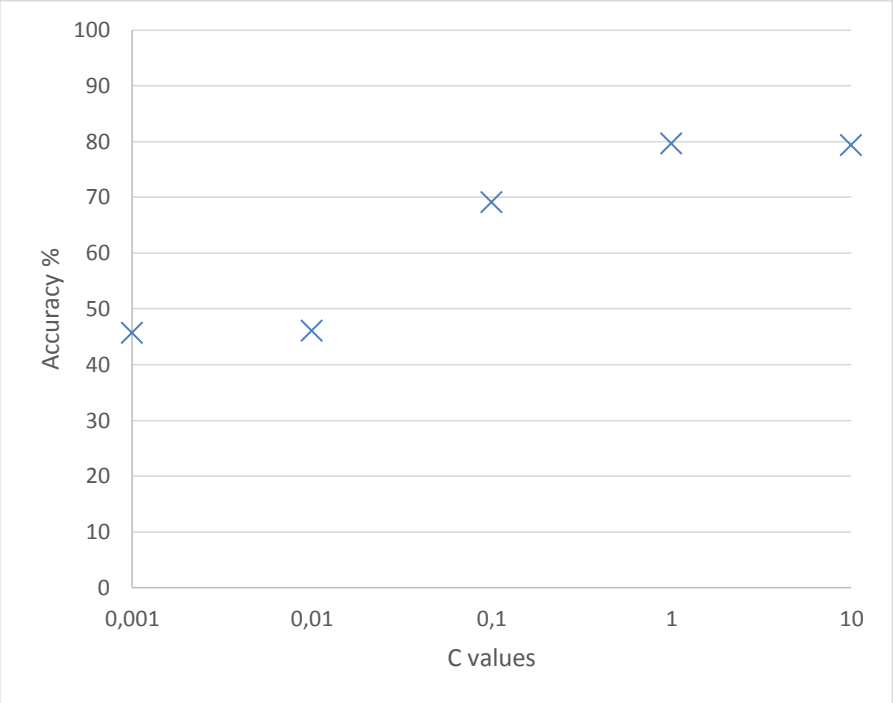

Figure 33: Accuracy with respect to C

As we can see, gamma=0.1, C=10, kernel='rbf' combination yielded the best accuracy. If one desires to trade off accuracy for faster training time C=1 and kernel='linear' is a decent compromise.

3.5.7 Confusion matrix

To show how well each class performed during testing, a confusion matrix is put in place.

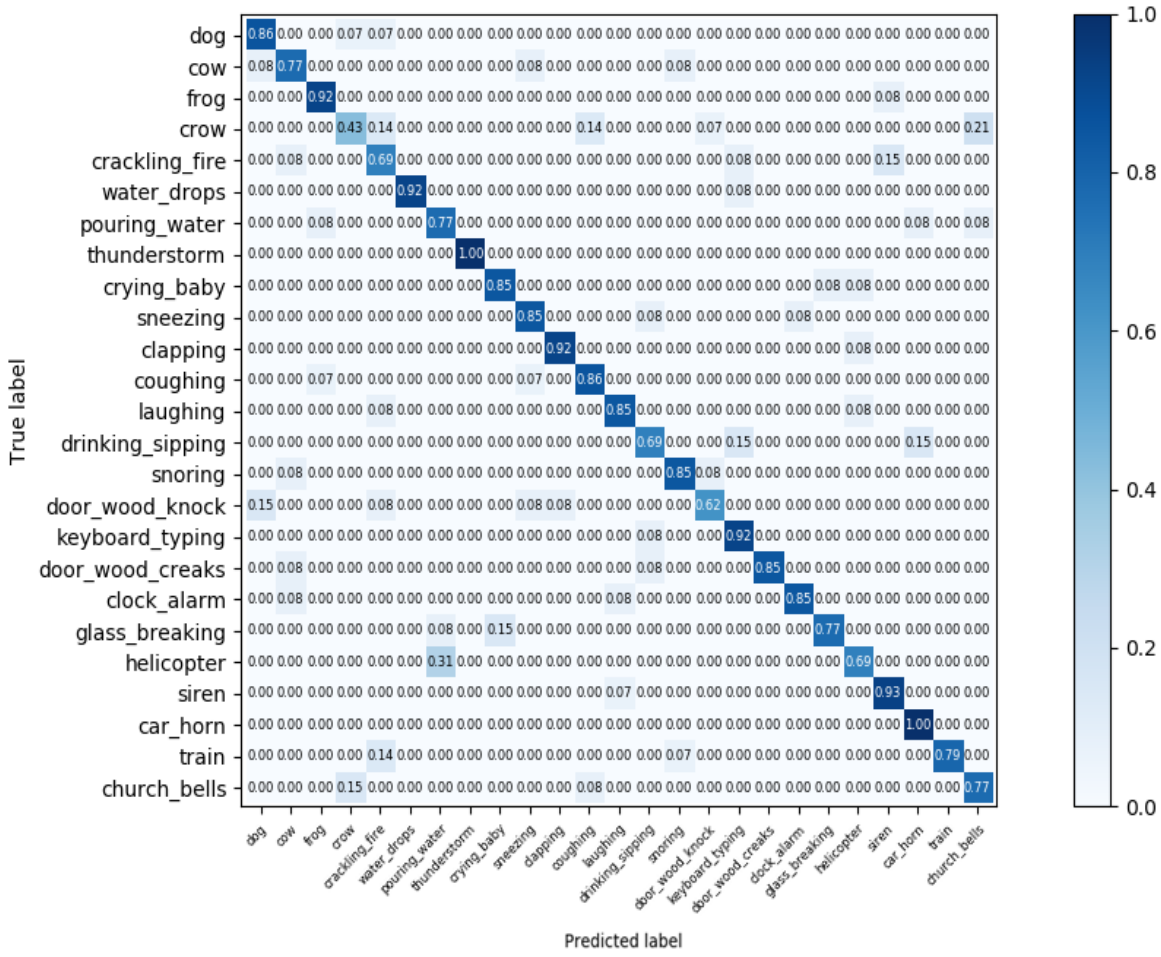

Figure 34: Confusion matrix

Two classes performed perfectly with no misclassifications which are: car horn and thunderstorm. Others, did not fare as good such as crow sounds, door knocking and helicopter going as low as 43% in accuracy.

3.6 Detection Result

Now we turn our focus to the result of the detection process, parameter tweaking is skipped at this stage and we use optimal ones gathered from classification phase. For evaluation purposes a newly formed testing audio clip captured directly from different YouTube videos containing all 25 sound classes. By sliding a window of length 1 second and an overlap of 0.1 second, first we detect whether the processed window event/not event when an event is detected it uses the model from previous section to try and predict the label for it.

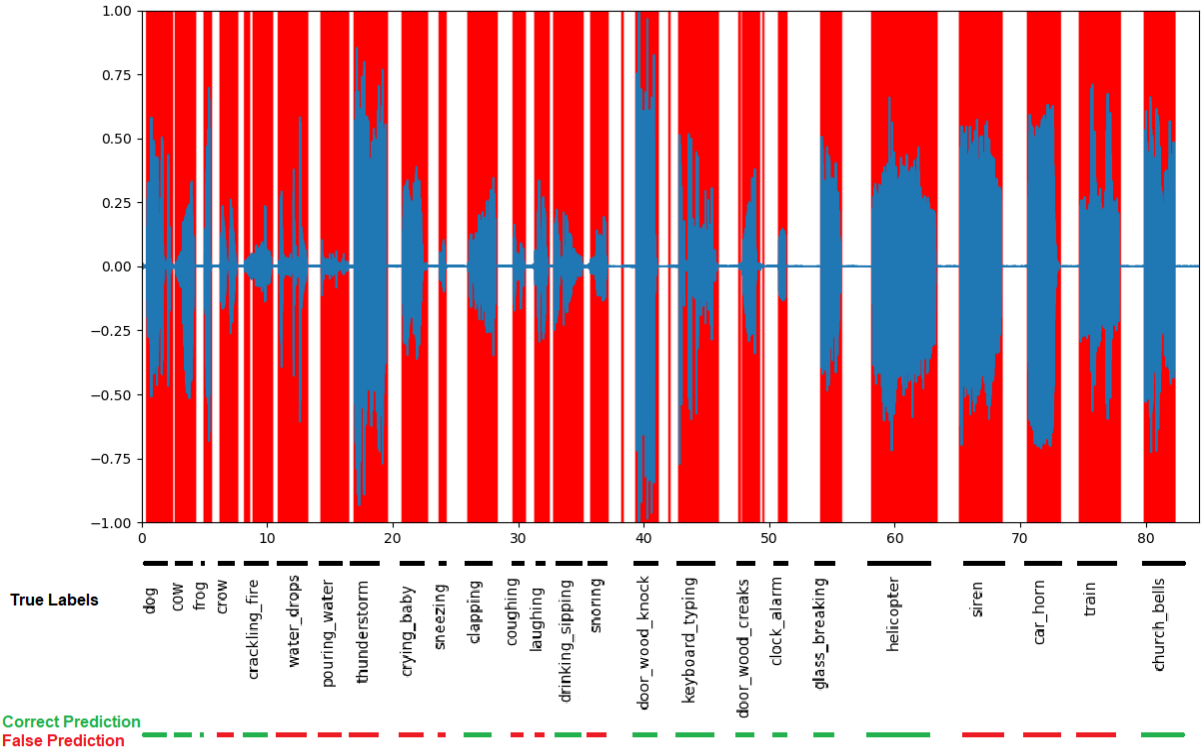

Figure 35 Audio event detection result

13 out of the 25 audio segments were correctly predicted, which is a significant decrease in accuracy, however the detection was way more precise in discerning between background noise and actual events.

3.7 Conclusion

In this chapter, we presented an implementation of audio detection and classification starting by describing the technics used and highlighting parameters that need to be adjusted. The results of the system were evaluated based on evaluation approaches cited in the literature of audio event detection.

The results obtained during the tests were satisfactory. However, testing the system in a new environment showed that it's far from ideal and require a larger dataset or alternative learning methods.

The main advantage of the method conducted lies in its ease of application and low consumption of memory.

Conclusion

In this work, we have carried out classification and detection using support vector machine trained on both cepstral and temporal features that we extracted after reducing them to 6 statistical descriptors, this proved to be very effective as shown with the classification results especially when compared to regular frame based training, which takes way more time and processing power. As for detection, we've shown how well SVM can perform in detecting timestamps for the events in a low background noise environment, while leaving more room for improvement in the approach taken as we saw with the classification of detected events from random YouTube clips.

This work allowed us to introduce ourselves to a very extensive and evolving research topic. It allowed us to familiarize ourselves with machine learning, audio features and event detection technics.

By successfully implementing and evaluating a system of classification and detection, we were able to achieve the objectives set at the start.

The developed system has limitations and restrictions, the main ones being not handling overlapping audio events very well and the system performs poorly to new different possible variations of the audio events, due to the small dataset.

As perspectives to this research work, we suggest, in the first place, the implementation of other unsupervised deep learning algorithms which should help in accounting for the different variations of the audio events and handling the overlapping sounds more effectively.

References

1. A. L. Albert, ELECTRICAL COMMUNICATION, New York; London: J. Wiley & Sons; Chapman & Hall, 1940, 1950.

2. B. Kostek, Perception-Based Data, Springer-Verlag Berlin Heidelberg, 2005.

3. J. W. Dennis, "Sound event recognition in smart environments Using Spectrogram Image Processing," in *2015 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, Bucharest, Romania, 2014.

4. R. Pujol, "cochlea," 06 06 2018. [Online]. Available: http://www.cochlea.org/en/hear/human-auditory-range. [Accessed 18 08 2019].

5. S. V.Vaseghi, Advanced Digital Signal Processing and Noise Reduction, John Wiley & Sons, LTD, 2001.

6. D. Gerhard, "Audio Signal Classification: History and Current Techniques," Regina, Saskatchewan, CANADA, 2003.

7. M. D. P. D. E. Tuomas Virtanen, Computational Analysis of Sound Scenes and Events, Springer, 2017.

8. D. F. Eyben, Real-time Speech and Music Classification by Large Audio Feature Space Extraction, Munich, Germany: Springer International Publishing, 2015.

9. A. A. H.-W. H. Xuedong Huang, Spoken Language Processing: A Guide to Theory, Algorithm and System Development, Prentice Hall PTR, 2001.

10. J. W. Dennis, "Sound Event Recognition in Unstructured Environments Using Spectrogram Image Processing," Jurong West, Singapore, 2014.

11. A. T. P. S. U. Koustav Chakraborty, "Voice Recognition Using MFCC Algorithm," *International Journal of Innovative Research in Advanced Engineering (IJIRAE),* vol. 1, no. 10, p. 4, 2014.

12. "mathworks," mathworks, [Online]. Available: https://www.mathworks.com/help/audio/ref/mfcc.html.

13. G. BachuR, "1 Separation of Voiced and Unvoiced using Zero crossing rate and Energy of the Speech Signal," 2008.

14. R. W. S. Lawrence Rabiner, Introduction to Digital Speech Processing, USA: Now Publishers, 2007.

15. P. R. Parwinder Pal Singh, "An Approach to Extract Feature using MFCC," *IOSR Journal of Engineering,* vol. 4, no. 8, pp. PP 21-25, 2014.

16. S. S.-S. Shai Ben-David, Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press, 2014.

17. Y. Zhang, New Advances in Machine Learning, InTech, 2010.

18. C. W. G. I. Sammut, Encyclopedia of Machine Learning and Data Mining, Springer, 2017.

19. C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

20. nalamidi, "github," 2018. [Online]. Available: https://github.com/nalamidi/Breast-Cancer-Classification-with-Support-Vector-Machine/blob/master/Breast%20Cancer%20Classification.ipynb. [Accessed 02 09 2019].

21. G. Zhang, "medium," medium, 11 11 2018. [Online]. Available: https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d. [Accessed 01 10 2019].

22. "scikit-learn," [Online]. Available: https://scikit-learn.org/stable/modules/svm.html. [Accessed 15 09 2019].

23. "dataaspirant," [Online]. Available: https://dataaspirant.com/2017/01/25/svm-classifier-implemenation-python-scikit-learn/.

24. "scikit-learn," [Online]. Available: https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html.

25. B. Stilman, Computer Intensive Methods in Control and Signal Processing, Birkhäuser Basel, 1997.

26. R. M. H. Selim Aksoy, "Feature normalization and likelihood-based similarity measures for image retrieval," *Pattern Recognition Letters - Special issue on image/video indexing and retrieval,* vol. 22, no. 5, pp. 563 - 582, 2001.

27. "scikit-learn," scikit-learn, [Online]. Available: https://scikitlearn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html. [Accessed 22 09 2019].

28. L. S. Moreno, "Machine detection of emotions: Feature Selection," 2016.

29. G. A. N. S. N. P. Juan Buhagiar, "Automatic Segmentation of Indoor and Outdoor Scenes from Visual Lifelogging," *Applications of Intelligent Systems,* vol. 310, 2018.

30. A. Waibel, Computers in the Human Interaction Loop, Springer, 2009.

31. P. K. F. K. M. M. R. M. I. M. A. M. Huy Phan, "What Makes Audio Event Detection Harder than Classification?," *Proceedings of the 25th European Signal Processing Conference (EUSIPCO),* 2016.

32. K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," 01 05 2015. [Online]. Available: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/YDEPUT. [Accessed 01 05 2019].

33. "python-speech-features," [Online]. Available: https://python-speech-features.readthedocs.io/en/latest/.

34. "scikit-learn," [Online]. Available: https://scikit-learn.org/stable/.

35. V. B. S. E. G. R. Romain Serizel, Acoustic Features for Environmental Sound Analysis, Springer, 2017.

36. "practicalcryptography," 2013. [Online]. Available: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/. [Accessed 20 8 2019].

37. E. Micheli-Tzanakou, Supervised and unsupervised pattern recognition, CRC Press, 1999.