Urban Sound Event Classification for Audio-Based Surveillance Systems

João Pedro Duarte Galileu

MSc Thesis

Supervisor: Prof. Dr. João Manuel Ribeiro da Silva Tavares

U. PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Master's Degree in Mechanical Engineering

January 2020

À minha família

Abstract

With the ever-growing population in urban centers, new challenges to the policing of the cities arise. The concept of safe cities is now, more than ever, worth investing in due to the reduced costs of new methods of surveillance. The advent of artificial intelligence allowed for new ways to mitigate the crime-related problems in urban environments and even help in the investigation after a crime has been committed.

With this project, its author pretends to study the solutions to these problems in audio-based surveillance and develop a prototype of an audio-based surveillance system capable of automatically detecting, classifying and registering a sound event. To achieve this goal, machine learning algorithms are implemented after pre-processing the stream of audio in real-time. The extracted audio features fed to the machine learning models are Mel-Frequency Cepstrum Coefficients (MFCC).

The dataset used is the UrbanSound8K audio dataset, comprised of 8732 audio samples of urban sounds from the following classes: Air Conditioner, Car Horn, Children Playing, Dog Barking, Drilling, Engine Idling, Gunshot, Jackhammer, Siren and Street Music. The machine learning algorithms used are Logistic Regression, Support Vector Machines, Random Forests, k-Nearest Neighbors and Artificial Neural Networks.

The algorithm that performed the best was the Artificial Neural Network, when paired with the use of 40 Mel-Frequency Cepstral Coefficients as input features. Along with identifying the class of sound, a log of the identified sounds is provided, allowing the user to retrieve the information of when the sound events took place.

In the making of this project, it was possible to see the potential of the use of machine learning algorithms to detect and classify any given sound, as long as the right dataset is used to train the models with. Also, it was observed that the quality of the microphone used to capture the stream of audio can have a significant impact in the predictions made by the machine learning models due to the low SNR. Despite this, a working prototype was able to correctly identify some of the classes present in the dataset. Thus, these algorithms present a viable alternative to traditional audio-surveillance systems, operated by humans.

Keywords: Audio Surveillance, Machine Learning, Multiclass Classification, Mel-Frequency Cepstral Coefficients, Real-time Sound Classification.

Classificação de Eventos Sonoros Urbanos para Sistemas de Áudio-Vigilância

Resumo

Com a crescente população nos centros urbanos, surgem novos desafios no que ao policiamento das cidades diz respeito. Agora, mais do que nunca, vale a pena investir no conceito de cidades seguras, devido aos reduzidos custos de novos métodos de vigilância. O surgimento da inteligência artificial permitiu novas maneiras de mitigar os problemas relacionados com o crime em ambientes urbanos e até mesmo ajudar na investigação depois de um crime ter sido cometido.

Com este projeto, o seu autor pretende estudar as soluções para esses problemas na vigilância baseada em áudio e desenvolver um protótipo de um sistema de vigilância baseado em áudio capaz de detectar, classificar e registar automaticamente um evento sonoro. Para atingir esse objetivo, os algoritmos de *machine learning* são implementados após o pré-processamento do áudio em tempo real. As caraterísticas do áudio extraídas fornecidas aos modelos de *machine learning* são Mel-Frequency Cepstral Coefficients (MFCC - Coeficientes Cepstrais de Frequência Mel).

O conjunto de dados usado é o conjunto de dados de áudio UrbanSound8K, composto por 8732 amostras de áudio de sons urbanos das seguintes classes: Ar Condicionado, Buzina de Carro, Crianças a Brincar, Cão a Ladrar, Berbequim, Motor de Carro, Tiro, Martelo Pneumático, Sirene e Música de Rua. Os algoritmos de *machine learning* usados são: Regressão Logística, Máquinas de Vetores de Suporte, Florestas Aleatórias, k-Nearest Neighbors e Redes Neurais Artificiais.

O algoritmo que apresentou o melhor desempenho foi a Rede Neural Artificial, quando emparelhada com o uso de 40 MFCCs como caraterísticas de entrada. Juntamente com a identificação da classe de som, é fornecido um registo dos sons identificados, permitindo ao utilizador recuperar as informações da altura em que os eventos de som ocorreram.

Na elaboração deste projeto, foi possível ver o potencial do uso de algoritmos de *machine learning* para detectar e classificar qualquer som, desde que o conjunto de dados correto seja usado para treinar os modelos. Além disso, observou-se que a qualidade do microfone usado para captar o áudio pode ter um impacto significativo nas previsões feitas pelos modelos de *machine learning* devido à baixa razão sinal-ruído. Apesar disso, um protótipo conseguiu identificar corretamente algumas das classes presentes no conjunto de dados. Assim, esses algoritmos apresentam uma alternativa viável aos sistemas tradicionais de vigilância por áudio, operados por seres humanos.

Palavras-chave: Audio-vigilância, *Machine Learning*, Classificação Multiclasse, Coeficientes Cepstrais de Frequência Mel, Classificação de Som em Tempo Real.

Agradecimentos

Gostaria de agradecer, em primeiro lugar, ao Professor Doutor João Manuel Ribeiro da Silva Tavares a oportunidade de desenvolver uma dissertação na área da análise de eventos sonoros por processos de *machine learning* e pelos seus conselhos e orientação ao longo deste trabalho.

Ao Eng.º Sérgio Jesus, colega e amigo, pelo inestimável incentivo e apoio oferecido ao longo da elaboração deste trabalho.

À minha família pelo apoio, oportunidade e investimento na minha educação e por estarem sempre disponíveis para me ajudar.

À Rafaela, por todos os momentos de conforto, carinho e compreensão nos períodos mais difíceis.

Finalmente, a todos os que me acompanharam na minha jornada na FEUP, amigos e colegas, que de alguma forma contribuíram para o meu crescimento como estudante e como pessoa.

Contents

List of Figures

List of Tables

Abbreviations

ADC	Analog-to-Digital Converter
ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
CSV	Comma-Separated Values
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
FN	False Negative
FP	False Positive
GUI	Graphical User Interface
IFT	Inverse Fourier Transform
kNN	k-Nearest Neighbors
LR	Logistic Regression
MFCC	Mel-Frequency Cepstral Coefficient
MIR	Music Information Retrieval
RF	Random Forest
SED	Sound Event Detection
SNR	Signal-to-Noise Ratio
STFT	Short-time Fourier Transform
SVM	Support Vector Machine
TN	True Negative
TP	True Positive

1 Introduction

This first chapter concerns the clarification of the subject matter of the project of this thesis and the context in which it is inserted. The used methodology, its implementation and the contents of each chapter are also presented.

1.1 Project context and motivation

In the interest of seeking prosperity, stability and social and educational facilities over the last two decades, the destination of choice for citizens and businesses has been urban centers, to the detriment of rural areas. As such, the concentration of population in metropolitan areas has been on the rise [1].

Over half of the world's population already lives in cities. The current estimates of urban population as a percentage of total population in the world sit at around 55% [2], with recent predictions raising that number to 68% by 2050 [3] and to as much as 85% by 2100 [4]. Out of the many outcomes from this trend on our cities, some problems arise, like the impact to the environment due to human activity, the increased stress on systems and infrastructures, the potential reductions in health and quality of life for city dwellers and, most importantly in the context of this thesis, the difficulty of effectively policing and securing public spaces.

As such, technological systems are a much-needed answer to such problems, with a well-established and growing trend of leveraging these solutions when addressing some of the most pressing issues facing urban communities [5]. Surveillance systems that use audio as one of the main sources of input may become a reality soon, due to the low cost when compared to cameras and the robustness to various adverse conditions. It is even possible to use arrays of microphones to detect anomalies in a city and pinpoint the exact location of its source. One example is the array of sensor nodes deployed on New York City streets, Figure 1.1, to analyze the sources of noise pollution in order to combat it.

Surveillance systems are based on one or more sensors able to acquire information from the surrounding environment. Whereas the first generation of surveillance systems implied monitoring activity by a human operator in order to detect anomalous situations or events, recently developed automated systems try to perform this task using computer vision and pattern recognition methodologies. This brings advantages such as cost savings, with the decrease in price of sensors and processing units, and the ability to cope with huge amounts of data originating from tens or even thousands of different sensors per surveillance system, which cannot be handled by human operators [6].

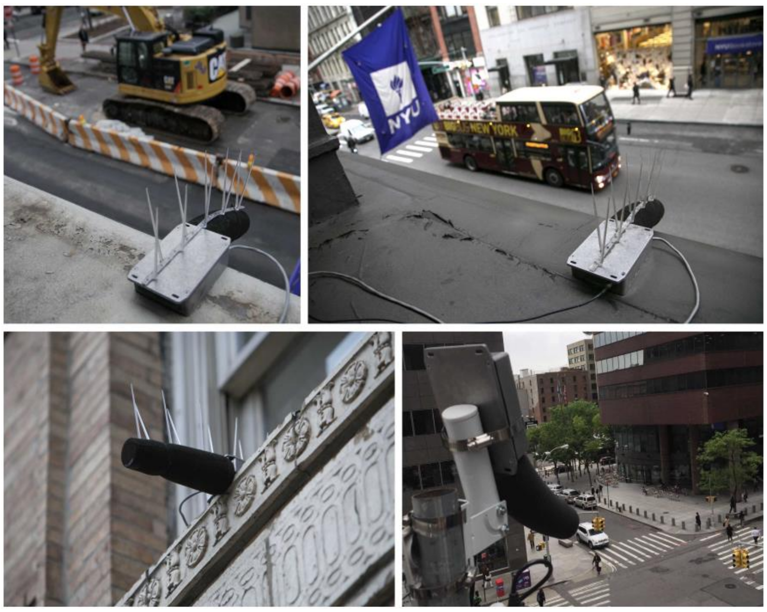


Figure 1.1 - Acoustic sensor nodes deployed on New York City streets [7]

1.2 Project objectives

With the previous issues in mind, the goal of this project is to build a sound event classifier that could be used in a real-time audio surveillance system.

This is done using the audio stream from a microphone, extracting the chosen audio features and feeding them as input in a machine learning model. This model, which needs to be trained with a labeled dataset, classifies a sound event in the given audio stream. Since there are different classes of sound events for the classifier to predict, we are dealing with a multiclass classification problem.

To achieve this goal, various multiclass classification algorithms will be tested and the one with the best performance will be implemented in the real-time solution. With a system as such in place, it can be expanded by adding other classes of sounds to the dataset, depending on the use cases.

The dataset to be used will be the *UrbanSound8K* dataset [8], a collection of 8732 sound excerpts collected from the *Freesound* [9] library of sounds. The audio samples of this dataset will be pre-processed and then fed into various machine learning models for multiclass classification.

1.3 Methodology used in the making of this project

This project had 4 main phases, which are detailed below.

A systematic review on audio surveillance [6] was used as the starting point to help understand what would be the background areas of knowledge needed to understand the subjects at hand. It was soon revealed that a good grasp on Digital Signal Processing (DSP) and *machine learning* methods was necessary.

Since the project involved the testing and use of several machine learning algorithms, the author of this thesis enrolled in an online course meant to give its students a broad understanding of machine learning algorithms and methodologies. After that, other courses, albeit smaller, were used in order to help programming in the *Python*™ programming language and to understand the DSP aspects of the project.

After knowing what tools were available, a total of five different machine learning algorithms were chosen, along with the feature extraction method. At the same time, the search for a dataset to work with begun. The *UrbanSound8K* dataset revealed itself to be one of the most used datasets when testing the performances of multiclass classification models using audio that wasn't composed of speech or music.

In the fourth phase, a prototype of an audio-surveillance system was built with the knowledge acquired in the first three phases, using whatever microphones were available, like the microphone built-in on the laptop used to program the system or the microphones built-in on several pairs of headphones. After achieving satisfactory results, the project was considered complete.

1.4 Thesis structure

This thesis is divided into 5 main chapters that present the development of the project. In the end, there is a list of references that served as the theoretical basis for all the concepts discussed in this document. The contents of the main chapters are as follows:

Chapter 1: The project context and motivation behind it are discussed, as well as its objectives and the structure of the document. The methodology followed in the making of this project is also addressed.

Chapter 2: Some background into Digital Signal Processing (DSP) is given in this chapter, with a special focus on the Mel-Frequency Cepstral Coefficients of audio signals. Along with this, there is the explanation of some of the Machine Learning models used in the Sound Event Detection (SED) and classification field, such as Logistic Regression, Support Vector Machines, Random Forests, k-Nearest Neighbors and Artificial Neural Networks.

Chapter 3: The methodology adopted, and the implementation of the system are discussed in this chapter, with a closer look into how the audio classification system came about. A closer look into a selected sample of each class of sounds is made.

Chapter 4: The results of the implementation of the methods presented in Chapter 3, accompanied by their interpretation and discussion. The results of the multiclass classification are shown, in the form of confusion matrices, for every machine learning model considered and for every amount of MFCCs desired. The real-time implementation results are also showed here.

Chapter 5: The conclusions of this project are presented and reflected upon, exposing the final state of the developed system and its strong points and flaws. Based on that, some options about a future work to be done in continuation of this document are provided.

2 Background

In audio classification, the predictive models operate on audio (digital sound) with tasks ranging from wake-word or speech command detection in Speech Recognition and music genre or artist classification in Music Information Retrieval (MIR) to classification of environmental sounds.

As with most pattern recognition tasks, sound event detection (SED) requires a good representation of input. The current state-of-art in pattern recognition problems applied to audio signals does not allow to draw an ultimate conclusion on the single best feature or the best feature set to be used in detection and classification task, irrespective of the kind of audio sources involved [6]. Regardless, some of the most common domains in audio analysis are the *time,* the *frequency,* the *time-frequency* and the *cepstrum* domains.

Feature extraction is a very important part in analyzing and finding relations between different things. The data provided by audio files cannot be understood by the models directly. To convert them into a format that is both understandable and with low complexity to feed them to machine learning models, it is required to extract the features that represent the data in a compact and useful way.

2.1 Surveillance systems

Surveillance systems can be quite helpful in an urban environment, yet they can also be quite controversial. In this subchapter, a discussion about different aspects of surveillance is presented.

2.1.1 The case for audio surveillance

As an important source of information about urban life, sound has great potential for use in smart city applications. Video cameras and other forms of environmental sensing are beginning to be complemented or even substituted by microphones, because of the ever-growing smart phone penetration and the development of specialized acoustic sensor networks.

Audio stream is generally much less onerous than video stream, in terms of bandwidth, memory storage, and computation requirements due to audio stream's one-dimensional nature (time) as opposed to the three-dimensional nature of video stream (width × height × time) [6].

Microphones are also generally smaller and less expensive than cameras, with the added benefit of being robust to environmental conditions in which video cameras would have a scarce performance [10], such as fog, pollution, rain, and daily changes in light conditions that negatively affect visibility, being also less susceptible to occlusion due to the bigger involved wavelength (many surfaces allow for specular reflections of the acoustic wave, thus permitting us to acquire audio events even when obstacles are present along the direct path, although this can be a drawback in sound localization tasks) and capable of omnidirectional sensing [5, 6].

It should be also added that several audio events important to the surveillance task, such as shouts or gunshots, have little to no video counterpart and that from the psychological point of view, audio monitoring constitutes a less invasive surveillance technology than video monitoring in regards to privacy concerns, so much so that it can be a valid substitute [11].

The combination of these facts encourages both the deployment of a higher number of audio sensors and a more complex signal processing stage [6].

The concept of audio surveillance as part of a *smart city* initiative would involve state-of-the-art sound sensing devices and intelligent software algorithms designed to detect and classify the different sounds that can be heard in an urban environment, producing an array of data that can be used to analyze and interpret many sound related aspects of the city, like noise pollution [12], active altercations and crime scenes detection and even migratory patterns of birds [13].

2.1.2 Machine listening

Audio event recognition, the human-like ability to identify and relate sounds from audio, is a nascent problem in machine perception [14].

Machine listening can be described as the auditory counterpart to computer vision, in that it combines techniques from signal processing and machine learning to develop systems that are able to extract meaningful information from sounds [12].

2.1.3 Urban Soundscape

The term urban soundscape can be referred to as the sound scenes and sound events that are commonly perceived in cities. Soundscapes can vary between cities and even neighborhoods, but yet they still share some qualities that set them apart from other soundscapes, like the rural soundscape, for instance, which primarily contains geophony, comprised by naturally occurring non-biological sounds, such as the sound of wind or rain. Urban environments' soundscapes are predominantly anthrophonic, comprised by the sounds produced by humans, such as human voice, traffic, construction, signals, machines, musical instruments, and so on [5].

The automatic capture, analysis, and characterization of urban soundscapes can facilitate a wide range of novel applications including noise pollution mitigation, context-aware computing, and surveillance. It is also a first step towards studying their influence on and/or interaction with other quantifiable aspects of city-life, including public health, real estate, crime, and education [5].

An open-source tool was developed for soundscape synthesis, able to generate large datasets of perfectly annotated data in order to assess algorithmic performance as a function of maximum polyphony and SNR, for instance, which would be prohibitive at a large scale and precision by using manually annotated data. With a collection of isolated sound events, this tool acts as a high-level sequencer, generating multiple soundscapes from a single probabilistically defined "specification" [12]. This tool could help in the *data augmentation* process, which will be discussed later on this thesis.

2.1.4 Challenges in urban sound monitoring

An audio signal is complex due to the superimposition of multiple audio sources and to the multi-path propagation resulting in echo and reverberation effects [6].

With the number of possible sounds being unlimited and densely mixed, urban environments are among the most acoustically rich sonic environments we could study. The production mechanisms and resulting acoustic characteristics of urban sounds are highly heterogeneous, ranging from impulse-like sounds such as gunshots to droning motors that run non-stop, from noise-like sources like air-conditioning units to harmonic sounds like voice. They include human, animal, natural, mechanical, and electric sources, spanning the entire spectrum of frequencies and temporal dynamics [5].

The complex nature of the interaction between the various sources and the built environment, which is often dense, intricate and highly reflective, creates varying levels of "rumble" in the background. As such, it is not unusual for the sources of interest to overlap with other sounds and to present low signal-to-noise ratios (SNR) which change intermittently over time, tremendously complicating the analysis and understanding of these acoustic scenes [5].

Some audio analysis tasks have a relatively clear delineation between what should be considered a "source of interest" and what should be considered "background" or "noise", as is the case with specific musical instruments and accompaniment in music, or individual speakers against the background in human speech. However, this distinction is far less clear in the case of urban soundscapes, where almost any sound source is a potential source of interest, and many "noise-like" sources such as idling engines or HVAC units can have similar acoustic properties even though their type and function are very different [5].

Urban soundscapes are not composed following top-down rules or hierarchical structures that can be exploited as in the case of speech and most music. However, natural patterns of activity resulting from our circadian, weekly, monthly, and yearly rhythms and cultural cycles abound [5].

2.1.5 Privacy concerns

As sound sensing devices become ubiquitous, intelligent and ever connected, using data science to collect, distribute and analyze data to understand the situation on the ground, anticipate future behavior and drive effective action, the surveillance system comes under scrutiny from a privacy point of view [5]. A recent survey on information privacy concerns, carried out with 1,000 respondents around Europe, has found that the majority of people are not familiarized with the concept of audio monitoring and that they tend to worry about this type of solution on a general level, even though they also become more confident with regards to the solution when thoroughly presented with it and its usage area [15].

One key question pertaining to the issue of privacy remains to this day: will sound surveillance be socially acceptable in the future in private places where the use of video is not (e.g. the bathroom and locker rooms)? [11]. If surveillance, whether audio or video based, of a conceived private place is absolutely necessary, audio-based surveillance may prevail as it is the less invasive mode, seeing as there is no surefire way to identify someone based on the voice alone.

2.2 Digital Signal Processing

According to Crocco et al. [6] and to Serizel et al. [16], Mel-Frequency Cepstral Coefficients (MFCCs) are one of the most used features when feeding a machine learning model for classification. For this reason, they will be studied in this subchapter in order to be able to proceed to the implementation of this feature extraction technique.

On Figure 2.1, we can observe a very summarized version of the feature extraction process of the MFCCs that will be the subject of study of this subchapter.

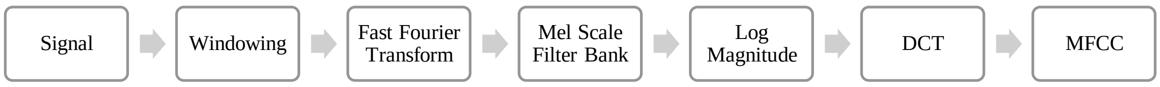


Figure 2.1 - MFCC feature extraction process (Adapted from [17])

2.2.1 Digital Sound Representation

Sound is a physical variation in pressure that propagates through a transmission medium over time. To process the sound with machine learning, it first must be converted to a digital format. The sound, that can be seen as acoustic data, is converted to analog electric signals by a microphone and then digitized using an Analog-to-Digital Converter (ADC), as illustrated in Figure 2.2 [18].

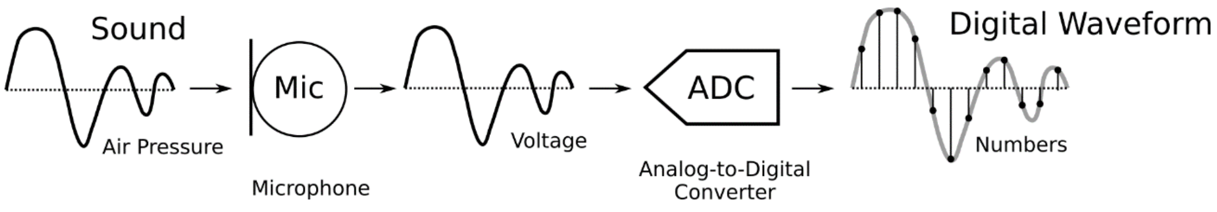


Figure 2.2 - Conversion of sound into a digital representation [18]

Sound waves are digitized by sampling them at discrete intervals of time. If we divide the number of intervals by the time it took to take them, we get a number known as the sampling rate, which is typically of 44.1kHz for CD quality audio, meaning samples are taken 44,100 times per second. If we want to keep a certain value for a frequency when digitizing sound, we must use a sampling rate that is over the double of that frequency. We call this frequency the Nyquist frequency.

Each sample is the amplitude of the sound wave at a particular point in time, where the bit depth determines how detailed the sample will be. This is also known as the dynamic range of the signal (typically 16bit, which means a sample can range from $2^{16} = 65,536$ amplitude values). In the representation on Figure 2.3, we can observe that sound is a one-dimensional sequence of numbers that represent the amplitude values of the sound wave at consecutive points in time, sometimes referred to as a *waveform*, [19].

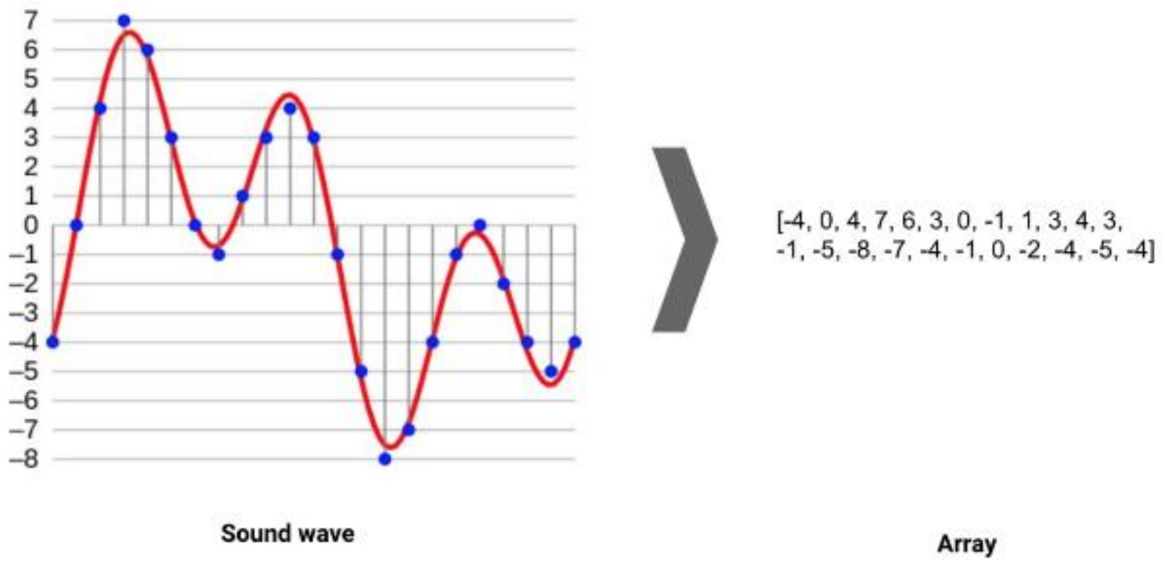


Figure 2.3 - A sound wave, in red, represented digitally, in blue (after sampling and 4-bit quantisation), with the resulting array shown on the right [19]

Although the digitization of sound shown in Figure 2.3 presents the sound as a one-dimension signal, the audio signal can actually be represented as a three-dimensional signal in which the three axes represent time, amplitude and frequency, as shown in Figure 2.4.

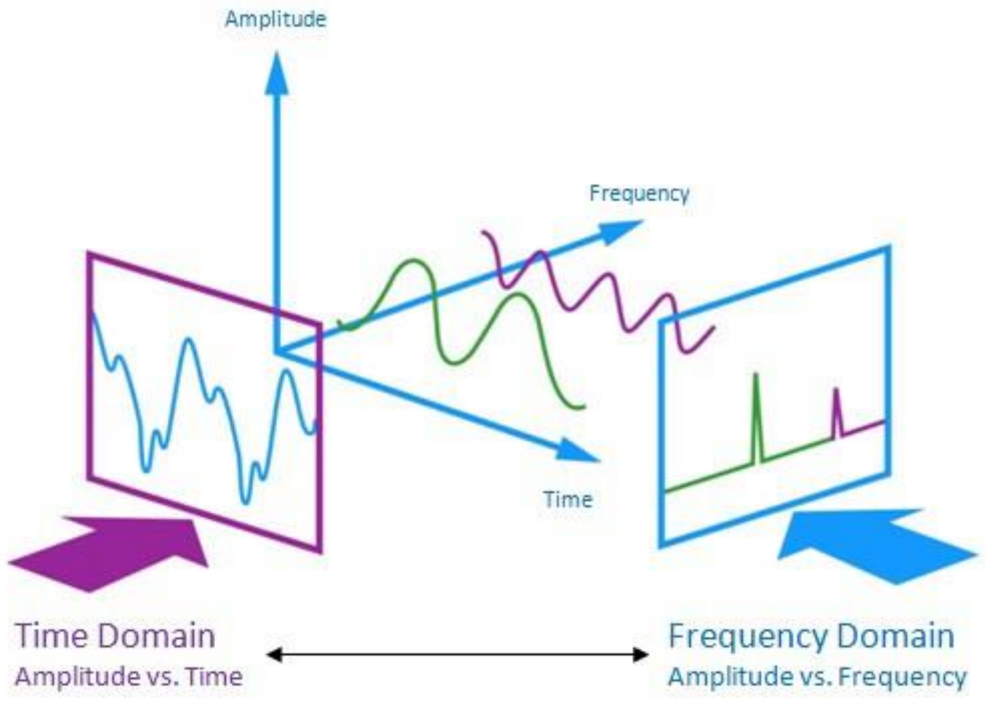


Figure 2.4 - Representation of the audio signal as a three-dimensional entity [20]

Sound signals are usually converted from the time domain to the frequency-domain prior to any analysis. The frequency-domain representation of a signal $x[n]$ on a linear-frequency scale can be obtained with the discrete-time Fourier transform (DFT), presented in Equation 2.1 [16]:

$$X[f] = \sum_{n=-\infty}^{\infty} x[n]\mathrm{e}^{-\mathrm{i}2\pi f n} \tag{2.1}$$

The spectrum $X[f]$ is f_s-periodic in f, with f_s being the sampling frequency. The frequency $f = f_s/2$ represents the Nyquist-frequency [16]. In other words, the spectrum is the frequency domain representation of the time-domain signal [21].

The spectrum $X[f]$ can be transformed back to time domain with the inverse discrete-time Fourier transform (IDFT), as seen in Equation 2.2 [16]:

$$x[n] = \frac{1}{f_s}\int_{-\frac{f_s}{2}}^{\frac{f_s}{2}} X[f]\mathrm{e}^{\mathrm{i}2\pi f n}\mathrm{d}f \tag{2.2}$$

In practice, the spectrum $X[f]$ is approximated by applying the DFT on a windowed frame of length N of the signal $x[n]$. This is referred to as the short-time Fourier transform (STFT) [16]. The fth component of the DFT of the tth frame of $x[n]$ is computed as follows:

$$X[t,f] = \sum_{k=0}^{N-1} w[k]x[tN+k]\mathrm{e}^{\frac{-\mathrm{i}2\pi k f}{N}} \tag{2.3}$$

where $w[k]$ is a window function (e.g. rectangular, Hamming, Blackman, etc.) used to attenuate some of the effects of the DFT approximation and to enforce continuity and periodicity at the edge of the frames. Equation 2.3 is given with a hop between frames equal to the length of the frames (*N*). This means that there is no overlap between consecutive frames. It is common to choose a hop size that is smaller than the frame length in order to introduce overlap that allows for smoother STFT representation and introduces statistical dependencies between frames [16].

The *t*th frame of time domain signal $x[n]$ can be obtained from the discrete spectrum $X[t,f]$ by applying the inverse STFT. Both the STFT and the inverse STFT can be efficiently computed using the fast Fourier transform (FFT) algorithm and the inverse fast Fourier transform (IFFT) algorithm, respectively [16].

2.2.2 Spectrogram

Sometimes, the representation of an audio signal in terms of its frequencies gives a better picture than the signal in the time domain. Even if we lose the time dependency, we get information about what frequencies are present in the wave.

Sound events of interest often have characteristic patterns not just in time (temporal signature) but also in frequency content (spectral signature). Most times, the analysis to the waveform is inconclusive when it comes to identifying the characteristics of the sound event. Therefore, it is common to analyze audio in a time-frequency representation, using something called a *spectrogram*.

Even though the Fourier transform is always a perfectly valid and accurate reconstruction of a representation of the data in the frequency domain, the results of the Fourier transform are only really easily interpretable when the signal maintains some stationarity over time. That stationarity means that the statistical and the spectral characteristics of the signal are not really changing much over time. We know that in the situation of a real world audio surveillance system that is not the case. In fact, it is the changes in the structure of the data over time that we are specifically interested in. Therefore, computing the entire Fourier transform over the entire signal doesn’t make a lot of sense. So we construct the spectrogram of the audio signal using the process shown in Figure 2.5.

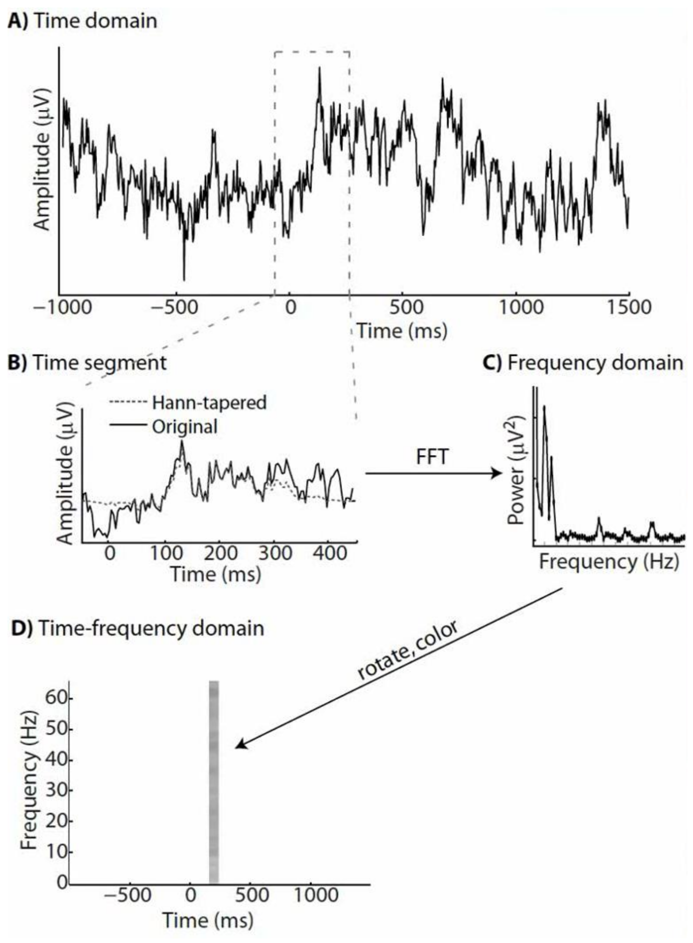


Figure 2.5 - The process of obtaining a spectrogram [22]

Rather than computing the Fourier transform over the entire signal, we just pick a small window and compute the Discrete Fourier Transform (DFT) of the data in that window, ignoring everything else in the signal. The method that allows for an analysis in short time windows (usually about 20 ms) is called the Short Time Fourier Transform (STFT). We do that by using an algorithm called Fast Fourier Transform (FFT), shown in Figure 2.6, which gives us the power spectrum (the frequency content) of the snippet of data. If we rotate the axis such that the frequency axis in now in the *y* axis, and color-code the amplitude of the power, we obtain one line of frequency in the time-frequency plane. The location of the line, called a *bin*, is at the center of the original window.

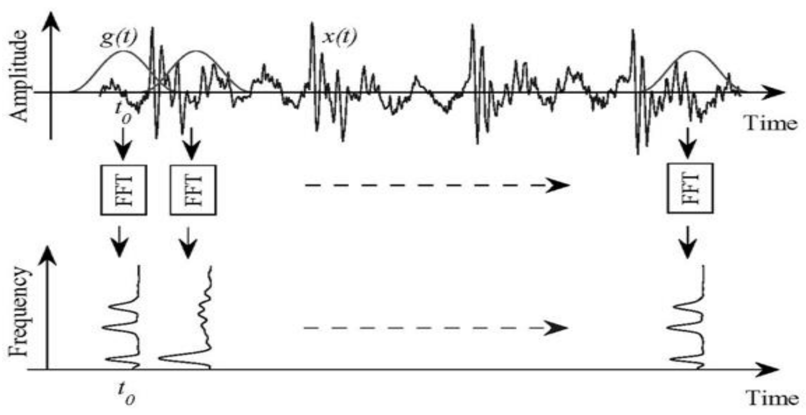


Figure 2.6 - Illustration of the Short Time Fourier Transform [23]

The STFT returns complex numbers describing the phase and magnitude of each frequency bin. A spectrogram is computed by squaring the absolute of the magnitude and discarding the phase information [18].

If we move the window of time, while overlapping it by about 50% over the previous window, we obtain a sequence of lines (or bins) that form what we call the spectrogram of the signal, populating the entire time-frequency plane.

It's also important to taper the data somehow. The reason why we want to do this is to avoid getting artifacts into the edges of the frequency representation, where we cut the windows. We apply a taper to the data to attenuate the signal going out into the edges and this basically avoids edge artifacts. Since we are also attenuating real signal and we're dampening it at the edges, we are also losing signal information and so the way that we account for this lost signal because of tapering is by moving the window in time steps that are smaller than the entire time step. So if we move the window by a hundred milliseconds, for instance, we get less attenuations in the end, where we before had attenuation because that part of the signal is closer to the center.

2.2.3 Cepstrum

The *cepstrum* is formed by taking the log magnitude of the spectrum, followed by an Inverse Fourier Transform (IFT). This results in a signal that's neither in the frequency domain (because of the Inverse Fourier Transform) nor in the time domain (because of the log magnitude prior to the Inverse Fourier Transform). The domain of the resulting signal is called the *quefrency* [21]. This process is detailed in Figure 2.7.

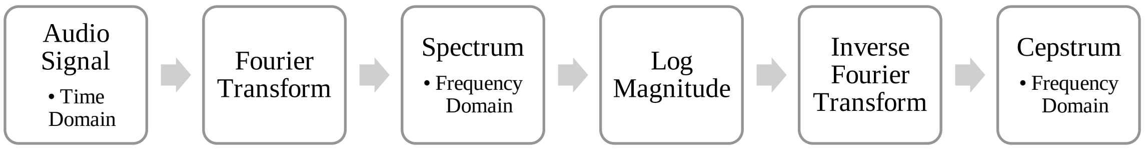


Figure 2.7 - Transformation from Audio Signal into Cepstrum (Adapted from [21])

2.2.4 Mel scale

The human ear resolves frequencies non-linearly across the audio spectrum and empirical evidence suggests that designing a front-end to operate in a similar non-linear manner improves recognition performance [24].

The mel-scale introduces a frequency warping effect in an attempt to conform with certain psychoacoustic observations which have indicated that the human auditory system can distinguish neighboring frequencies more easily in the low-frequency region. Frequency f in Hertz is converted to m mel by using Equation 2.4.

$$m(f) = 2595 \log\left(1 + \frac{f}{700}\right) \tag{2.4}$$

where mel is the unit of pitch. This function is graphed in Figure 2.8.

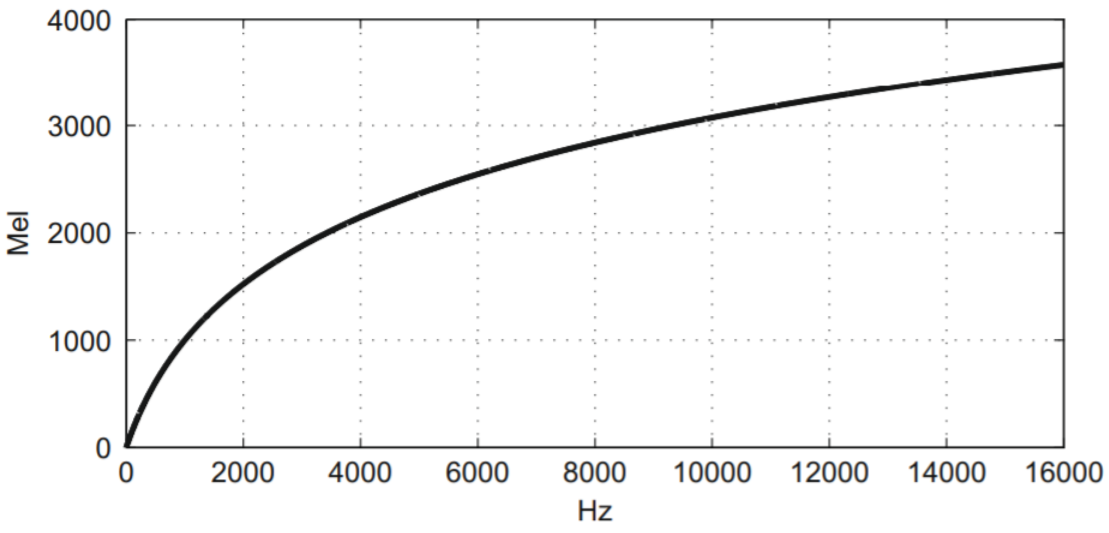


Figure 2.8 - Frequency warping function for the computation of the MFCCs [25]

The inverse transform can be readily derived as

$$f = 700\left(10^{\frac{m}{2595}} - 1\right) \tag{2.5}$$

2.2.5 Discrete Cosine Transform

Discrete Cosine Transformations (DCTs) are used to convert data into the summation of a series of cosine waves oscillating at different frequencies. They are widely used in image and audio compression and are very similar to Fourier Transforms. The difference is that DCT involves the use of just *cosine* functions and real coefficients, whereas Fourier Transformations make use of both the *sine* and *cosine* functions and require the use of complex numbers. For this reason, DCTs are simpler to calculate [26].

Both Fourier Transforms and DCTs convert data from a *spatial-domain* into a *frequency-domain* and their respective inverse functions convert the *frequency-domain* back the other way. When compressing analog signals, we often discard information to enable efficient compaction. We must be careful about what information in a signal we should discard when removing bits to compress a signal. DCT helps with this process by carefully removing the higher-frequency elements of an analog signal in way as to not distort the signal so much that it is unrecognizable [26].

2.2.6 Filter Bank

In practice, a common implementation for obtaining MFCCs uses a triangular filter bank where each filter is spaced according to the Mel-Frequency scale (Equation 2.4; see also Figure 2.9). The energy coefficients $\tilde{X}_m(t)$ in the band *m* are obtained as a weighted sum of the spectral amplitude components $|X[t, f]|$ (where the weights are given according to the amplitude value of the corresponding triangular filter). The number of filters typically varies between 12 and 30 for a bandwidth of 16 kHz [16].

Each filter in the filter bank is triangular, having a response of 1 at the center frequency and decreasing linearly towards 0 until it reaches the center frequencies of the two adjacent filters, where the response is 0, as shown in Figure 2.9, which shows a filter bank with 40 filters, set according to the mel scale.

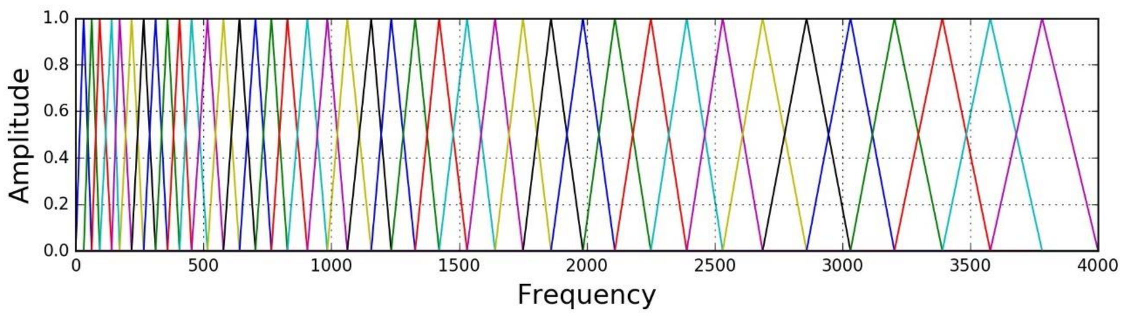


Figure 2.9 - Filter bank with 40 filters on a Mel-Scale [27]

This filter bank can be modeled by the system of Equations 2.6 [28]:

$$H_m(k) = \begin{cases} 0, & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)}, & f(m-1) \le k < f(m) \\ 1, & k = f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)}, & f(m) < k \le f(m+1) \\ 0, & k > f(m+1) \end{cases} \tag{2.6}$$

Where:

m is the number of filters we want, and
$f(\cdot)$ is the list of $m + 2$ mel-spaced frequencies

To implement this filter bank, the window of audio data is transformed using a Fourier transform and the magnitude is taken. The magnitude coefficients are then *binned* by correlating them with each triangular filter. Here, *binning* means that each FFT magnitude coefficient is multiplied by the corresponding filter gain and the results accumulated. Thus, each *bin* holds a weighted sum representing the spectral magnitude in that filter bank channel [24].

2.2.7 Mel-Frequency Cepstral Coefficients

Passing a spectrum through the Mel filter bank, followed by taking the log magnitude and a discrete cosine transform (DCT) produces the Mel-Cepstrum. DCT extracts the signal's main information and peaks. The peaks are the gist of the audio information. The DCT can be described as the process of transforming the log filter bank amplitudes into cepstral coefficients [29]. Typically, the first 13 coefficients extracted from the Mel cepstrum are called the MFCCs. These hold very useful information about audio and are often used to train machine learning models [21].

The Mel-Frequency Cepstrum (MFC) is a representation of the short-term power spectrum of a sound based on a linear cosine transform of a Log power spectrum on a nonlinear mel scale of frequency. MFCCs are derived from a type of cepstral representation of the audio clip and are commonly used as features in Automated Speech Recognition (ASR) systems. Recently, they have been used in music information retrieval (MIR) methods, such as genre classification.

A summary of the feature extraction process can be found below [30]:

- Audio signal is analyzed over short analysis windows (10 to 30 ms)
- For each short analysis window, a power spectrum is obtained using FFT
- Spectrum (or periodogram power spectral: the square of absolute value of the complex Fourier Transform) is passed through Mel-Filters (Mel-spaced filter-

bank), using 40 triangular overlapping windows, to obtain Mel-Spectrum (energies)

- Take the logarithm of the powers at each of the mel-frequencies
- Take the Discrete Cosine Transform (DCT) of the list of mel log powers (log filter bank energies), as if it were a signal, to get uncorrelated MFCCs, which are the amplitudes of the resulting spectrum. We obtain 40 cepstral coefficients. We can choose N of the coefficients to keep, as long as N≤40.

The first 13–20 cepstral coefficients of a signal's short time spectrum succinctly capture the smooth spectral envelope information [31].

As such, the audio signal is represented as a sequence of cepstral vectors. It is these vectors that are given to pattern classifiers for sound event classification.

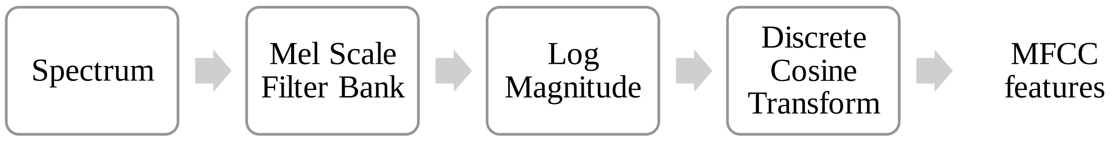


Figure 2.10 - Process of obtaining the MFCC (Adapted from [21])

2.3 Machine Learning

As stated in [6], the problem of sound event classification can be somewhat automated with the use of pattern recognition methodologies, such as is the case with machine learning.

A machine learning algorithm, also called model, is a mathematical expression that represents data in the context of a problem. The aim is to go from data to insight. There are two general ways to do this: *supervised* learning and *unsupervised* learning. We apply supervised machine learning techniques when we have a piece of data that we want to predict or explain. We do so by using previous data of inputs and outputs to predict an output based on a new input. In contrast, unsupervised machine learning techniques look at ways to relate and group data points without the use of a target variable to predict. In other words, it evaluates data in terms of traits and uses the traits to form clusters of items that are similar to one another [32].

Inside supervised learning, we can have two different types of problems: classification and regression. Classification is done when we want to assign data to a class, while regression is done when we want to predict a value for an unseen instance. On the other hand, unsupervised learning is done by clustering, where the data is not labelled, but can be divided into groups based on similarity and other measures of natural structure in the data. This hierarchy is displayed in Figure 2.11.

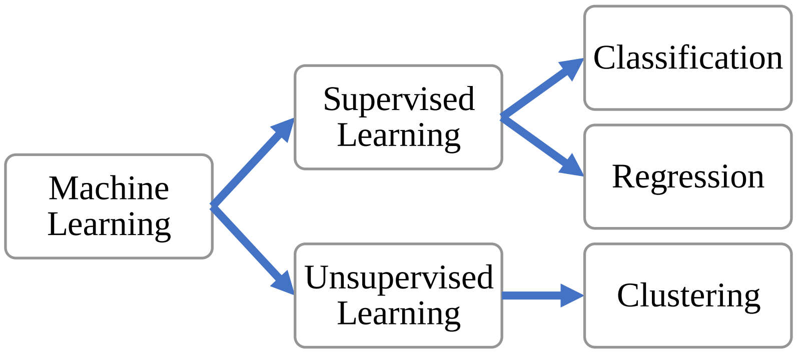


Figure 2.11 - Different types of machine learning (Adapted from [33])

2.4 Classification Models

Since we will be working with a labeled dataset, we will rely on supervised machine learning algorithms to do our audio surveillance project. As we will see in the dataset overview lator on this thesis, there is only one class of sound per audio file. That makes this a multiclass classification problem, as opposed to a multilabel classification problem, where the data to be classified can have more than one class in it. In order to classify a sound in a given audio signal, after the feature extraction process, we feed such features to the classifiers.

The best multiclass classification models are discriminative ones which try to directly construct the best hyper-surface of separation in the feature space, dividing it into subspaces segregating the most similar training samples for each class. This class of models presents high performances but lack model interpretability. Artificial Neural Networks (ANN) and Support Vector Machines (SVM) are the most common discriminative models employed in the audio classification task [6].

Nevertheless, we will be taking a look at some prominent classification models, some directly used in multiclass classification and some that are inherently binary classifiers.

2.4.1 Logistic Regression

Logistic Regression (LR) is one of the most basic classification algorithms. LR is not a regression algorithm though, but a probabilistic classification model. It extracts real-valued features from the input, multiplies each by a weight, sums them, and passes the sum through a sigmoid function, Figure 2.12, to generate a probability. A threshold is used to make a decision [34].

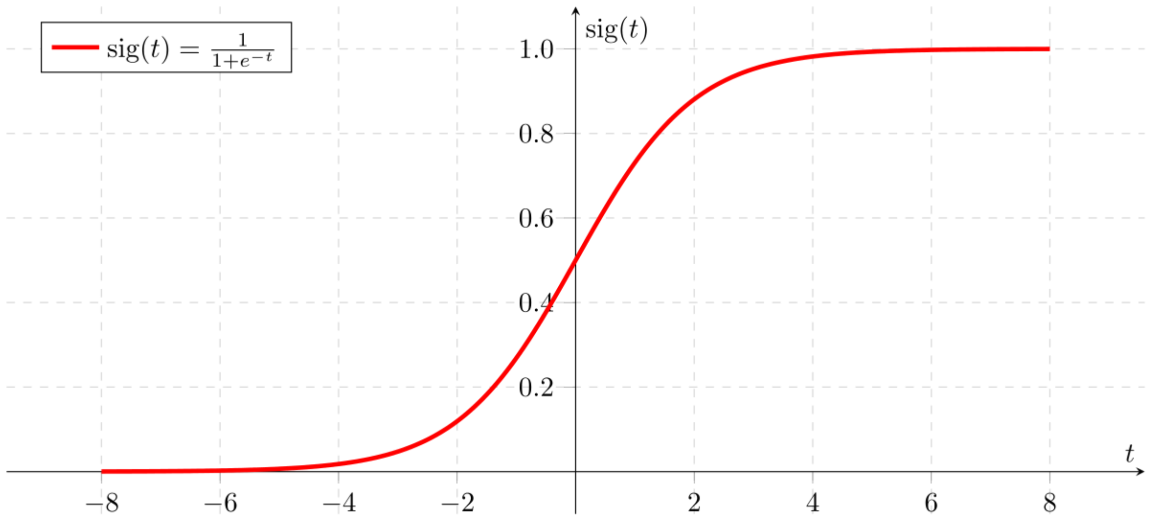


Figure 2.12 - Sigmoid function [35]

It is by nature a binary classification method, so instead of the output **y** = {0, 1}, we will need to expand the definition so that the output is **y** = {0, 1, …, N}, where N is the number of classes to be classified..

Since **y** = {0, 1, …, N}, we divide our problem into N+1 binary classification problems where, in each one, we predict the probability that **y** is a member of one of our classes.

We are basically choosing one class and then lumping all the others into a single second class, the so-called one-vs-rest method. We do this repeatedly, applying binary logistic regression to each case (Figure 2.13), and then use the hypothesis that returned the highest value as our prediction. Because logistic regression is the simplest classification model, it’s a good place to start for classification [36].

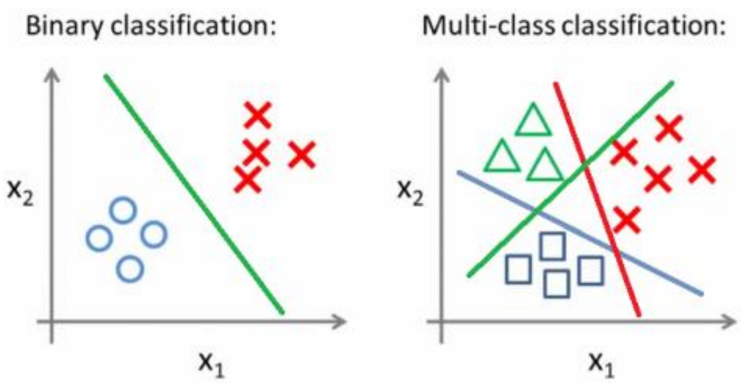


Figure 2.13 - Binary classification and multi-class classification [36]

Multiclass logistic regression uses the softmax function (Equation 2.7) to compute probabilities.

$$\text{softmax}(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{2.7}$$

In other words, we apply the standard exponential function to each element y_i of the input vector $\boldsymbol{y}$ and normalize these values by dividing by the sum of all these exponentials. With this normalization, we ensure that the sum of the components of the output vector is 1.

The weights are learned from a labeled training set via a loss function that must be minimized. Minimizing this loss function is a convex optimization problem, and iterative algorithms like gradient descent are used to find the optimal weights. Logistic regression is also one of the most useful analytic tools, because of its ability to transparently study the importance of individual features [34].

The logistic regression cost function can be found in Equation 2.8:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right)$$

$$= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right] \tag{2.8}$$

Where:

$J(\theta)$ is the cost function's value
m is the number of features
h_θ is the hypothesis function
$y^{(i)}$ is the actual value of the output, and
$x^{(i)}$ is the feature with index i

One-vs-Rest

The One-vs-Rest approach, illustrated in Figure 2.4, involves training a single classifier per class, with each sample of that class seen as a positive sample and all other samples as negatives. To do that, one must break the N classes problem into N binary problems and solve each one separately. This strategy requires the base classifiers to produce a real-valued confidence score for its decision, rather than just a class label. An advantage of this method is its interpretability. Since each class is represented by one and only one classifier, it is possible to gain knowledge about the class by inspecting its corresponding classifier.

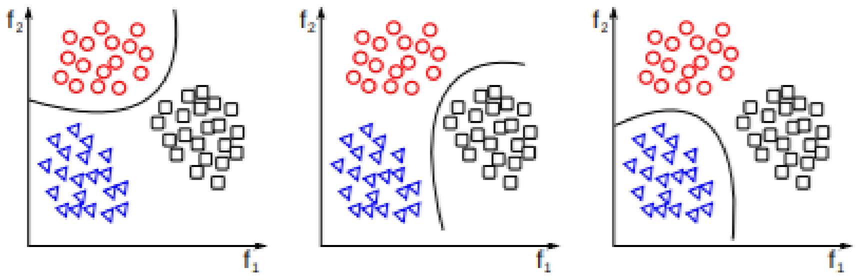


Figure 2.14 - One-vs-Rest approach in multiclass classification [37]

Making decisions means applying all classifiers to an unseen sample x and predicting the label k for which the corresponding classifier reports the highest confidence score with Equation 2.9:

$$\hat{y} = \underset{k \in \{1 \ldots K\}}{\operatorname{argmax}} f_k(x) \tag{2.9}$$

2.4.2 Support Vector Machines

As SVM classification was originally developed for binary discrimination, the extension to multi-class classification has been achieved by a set of one-vs-one or one-vs-all strategies: in the former case, $N \cdot (N-1) = 2$ SVMs, with N being the number of classes, are trained with data related to each couple of classes, while in the latter case N SVMs are trained taking into account all the data available [6].

SVM classifiers operate based on a simple principle: given an input of training data consisting of two classes, the system will generate a dividing hyperplane with maximum distance to the training samples, as seen in Figure 2.15. Twice this distance is considered as the margin. Margin maximization reduces susceptibility to noise while employing the SVM to classify new data sets [38].

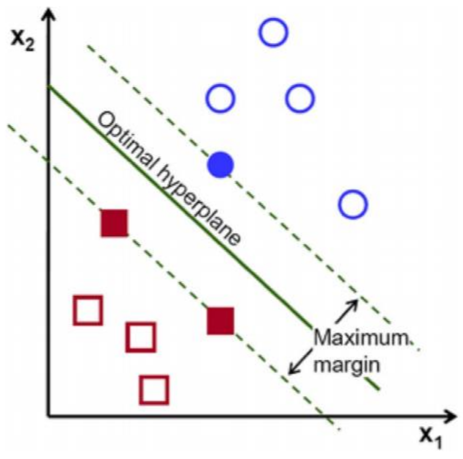


Figure 2.15 - Demonstration of optimal SVM hyperplane. The samples in full are the support vectors [38]

2.4.3 Random Forests

The Random Forest algorithm is an ensemble method that combines many Decision Trees trained with different samples of the data sets. As a result, the quality of the predictions of a Random Forest is higher than the quality of the predictions estimated with a single Decision Tree [32].

Decision trees are pretty intuitive and easy to understand. They basically decide what feature allows a split in the observations at hand in a way that the resulting groups are as different from each other as possible (and the members of each resulting subgroup are as similar to each other as possible). The basic structure of a decision tree is shown in Figure 2.16.

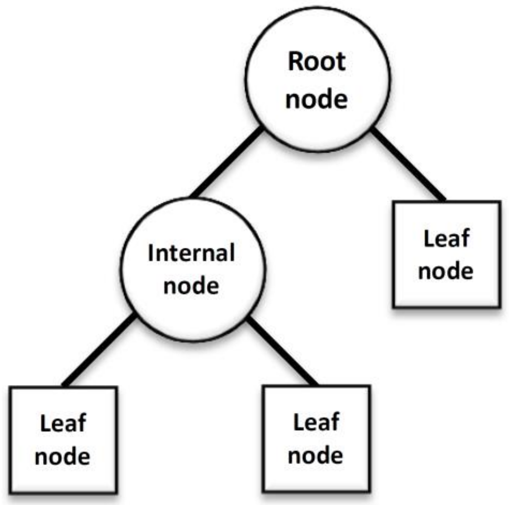


Figure 2.16 - Basic structure of a Decision Tree [39]

The combination of learning models increases the classification accuracy. That is the main idea behind a term called *bagging*, a technique that is used to average noisy and unbiased models in order to create another model with a lower variance in terms of classification. The Random Forest algorithm works as a large collection of decorrelated Decision Trees, hence the name "Forest", and uses this group of Decision Trees to perform a classification.

Let's suppose we have a matrix of training samples S (Equation 2.10) that we will submit to the algorithm to create a classification model, where the *jth* element of the *ith* row represents the *jth* feature of the *ith* sample. The last column indicates the class of the element.

$$S = \begin{bmatrix} f_{A1} & f_{B1} & f_{C1} & C_1 \\ \vdots & \vdots & \vdots & \vdots \\ f_{AN} & f_{BN} & f_{CN} & C_N \end{bmatrix} \tag{2.10}$$

If we create M subsets with random values from the sample set (we call this *bootstrapping*) and then create a Decision Tree for each one of them considering only a subset of the variables at each step, we will end up with a large variety of trees. The variety is what makes Random Forests more effective than individual decision trees.

After that, we do a ranking of classifiers in the following way: we use each of the Decision Trees we created and feed the sample we would like to classify to them. After obtaining the results from the Decision Trees, the class that was predicted the most (we can think of it as having the majority of votes) is the overall predicted class. *Bootstrapping* (random sampling with replacement) the data and *aggregating* the results to make a decision gives us the definition of *bagging* [40].

2.4.4 K-Nearest Neighbors

The k-Nearest Neighbors (kNN) algorithm is based on feature similarity. That means that the model will output a prediction that is related to how closely a given instance's features resemble the training set, as shown in Figure 2.17. The test sample (inside circle) should be classified either to the first class of blue squares or to the second class of red triangles. If k = 3 (outside circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If, for example, k = 5, it is assigned to the first class (3 squares vs. 2

triangles outside the outer circle). As such, the classification will depend on the *k* number of neighbors chosen to classify the new instance.

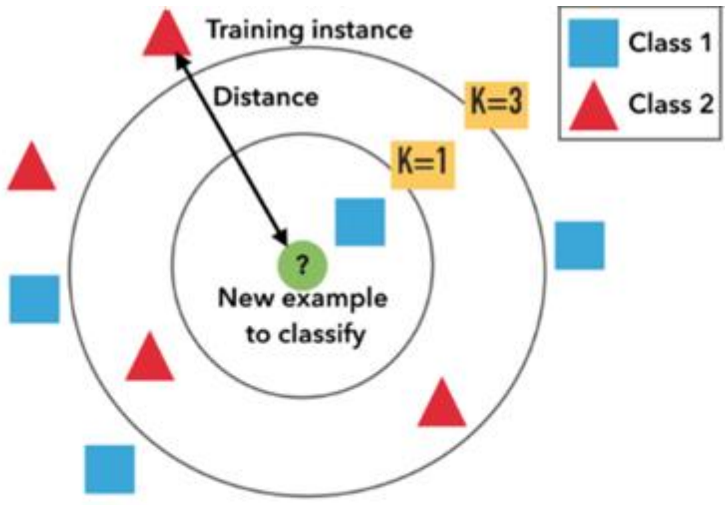


Figure 2.17 - Example of kNN classification [41]

The output of a kNN model used for multiclass classification will be the number correspondent to the class to which the object was attributed, by a majority vote of its neighbors.

This simple (to explain and to understand/interpret) algorithm has relatively high accuracy when compared to other simple algorithms but is computationally expensive because it stores all of the training data, which might make the prediction stage slow, especially with a large number of samples. This algorithm produces good results, generally speaking.

We have to choose an odd number for *k* for a 2 class problem and *k* must not be a multiple of the number of classes to prevent ties. We also have to remember that the main drawback of kNN is the complexity in searching for the nearest neighbors for each sample. In the case of a big dataset, we have lots of elements, so this can be a problem [41].

2.4.5 Artificial Neural Networks

An artificial neuronal network is a prediction model formed by several layers of “neurons” that send signals to each other, starting in the first layer (input layer) and ending in the last layer (output layer), in a process that tries to mimic the human brain.

In contrast to linear models, the objective of neural networks is to capture non-linear patterns in data by adding layers of parameters to the model [32].

The neuron is the basic unit of the artificial neuronal network. It receives a set of signals. from the neurons of the previous layer and a constant (usually named bias), makes the sum, and using an activation function, outputs the signal to the neurons in next layer. Exceptions to this mode of operation are the input and output layers. The input layer will be the size of the number of parameters in the sample and receives one signal per parameter. The output layer conveys the model prediction. As such, these layers have a number of neurons equal to the number of parameters and to the number of possible classes of classification, respectively.

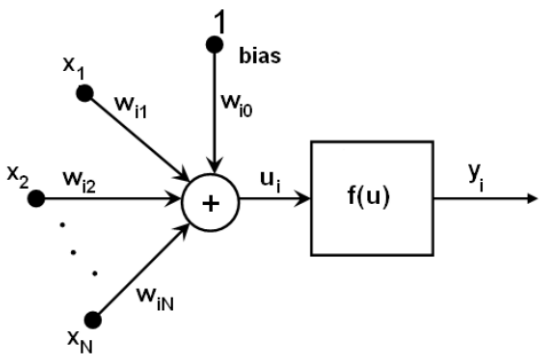


Figure 2.18 - Shape and function of a neuron [42]

The behavior of a neuron in a neural network can then be described by Equations 2.11 and 2.12:

$$u_i = w_{i0} \sum_{j=1}^{N} \left(x_j \times w_{ij} \right) \qquad (2.11)$$

Where:

u_i is the signal neuron *i* receives
w_{i0} is the bias of neuron *i*
x_j is the signal neuron *j* emits, and
w_{ij} is the weight of the signal of neuron *j* in neuron *i*

$$y_i = f(u_i) \qquad (2.12)$$

Where:

u_i is the signal neuron *i* receives, calculated by Equation 2.11
f is the neurons activation function, and
y_i is the signal neuron *i* emits

The activation functions normally share these characteristics:

- Pass through the origin (point (0,0));
- Have horizontal asymptotes;
- Tend quickly to horizontal asymptotes;
- Have positive y values when x is positive and negative y values when x is negative, i.e., they are drawn in the first and third quadrants.

Some examples of these functions are the hyperbolic tangent function, arctangent function, identity function, sigmoid function and ReLU (Rectified Linear Unit). The best results are provided by the hyperbolic tangent, sigmoid and more recently, the ReLU functions, described in Equations 2.13, 2.14 and 2.15, respectively:

$$f(u) = \frac{(e^{u} - e^{-u})}{(e^{u} + e^{-u})} \qquad (2.13)$$

$$f(u) = \frac{1}{1 + e^{-u}} \qquad (2.14)$$

$$f(u) = \begin{cases} 0, & u < 0 \\ u, & u \geq 0 \end{cases} \qquad (2.15)$$

In Figure 2.19 we can observe an example of an ANN architecture.

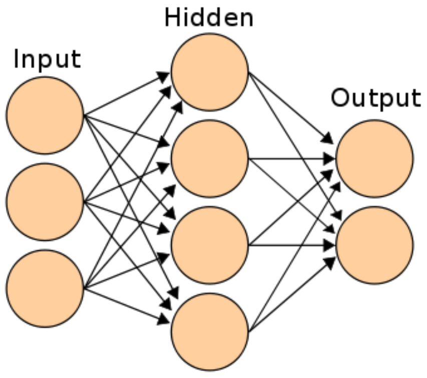


Figure 2.19 - Artificial Neural Network architecture with 3 input neurons, 4 hidden neurons and 2 output neurons [43]

Backpropagation/Gradient Descent

For the learning part of the model, backpropagation or gradient descent is used, which consists in minimizing the prediction error given a certain sample and the correct classification. For this, it is necessary to partially derive the function of the error in relation to the output neurons and continue the derivation from the previous layers.

After derivation it is necessary to adjust the weight and bias parameters of each neuron to minimize the error, i.e. nullify the partial derivative. For this method, it is convenient that the activation function of the neuron is easily derivable, hence its importance. The result will be a vector with the lowest slope direction in the function. error, and with successive iterations, it is possible to find a local minimum of the loss function.

2.5 Model performance metrics

Since the models we will be using will classify labelled data, we can evaluate them based on the predictions that they output. Some standard supervised learning metrics are as follows.

2.5.1 Classification related metrics

In classification problems where the classes are unbalanced, there is a tendency for the model to adapt only to dominant classes, correctly classifying most samples that belong to the dominant classes but incorrectly classifying the samples that are from the minority class. This can result in models that, despite their high accuracy, are not useful [44].

The confusion matrix (Table 1) is one of the most intuitive methods used for finding the performance of the model. It is used for classification problems where the output can be of two or more types of classes [45].

Table 1 - Binary classification's confusion matrix (Adapted from [44])

		Actual Class	
		Positive (1)	Negative (0)
Predicted Class	Positive (1)	True Positive (TP)	False Positive (FP)
	Negative (0)	False Negative (FN)	True Negative (TN)

With the values present within this matrix, we can calculate several performance metrics. One such metric is *accuracy*, which can be calculated as

$$accuracy = \frac{TP + TN}{N} \tag{2.16}$$

where N is the total number of instances. Accuracy is then the proportion of unseen instances that was correctly classified.

One of the most commonly used metrics to know the performance of the classifier is the *F-score*, which relates precision and recall, and it aims to give importance to classification of classes who are in the minority.

$$F = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.17}$$

Where:

$$precision = \frac{TP}{TP + FP} \tag{2.18}$$

$$recall = \frac{TP}{TP + FN} \tag{2.19}$$

As we can see from Equation 2.18, precision is the proportion of predicted positives that is truly positive. From Equation 2.19, we can see that recall can be defined as the proportion of actual positives that is correctly classified as positive.

The macro-averaged F1-score, precision and recall can be calculated by doing an arithmetic mean of the per-class metrics. The weighted-averaged metrics can also be calculated by multiplying each of the classes' metrics by the respective number of samples and then dividing by the total number of samples.

2.6 Model performance optimization

A machine learning model's performance can be optimized by doing, for instance, cross validation and data augmentation.

2.6.1 Cross-validation

A solution to the problem of ensuring each instance is used for training and testing an equal number of times while reducing the variance of an accuracy score is to use cross validation. Specifically, k-fold cross validation, where k is the number of splits to make in the dataset. A very common value for k is 10. This will split the dataset into 10 parts (10 folds) of equal number of samples and the algorithm will be run 10 times. Each time the algorithm is run, it will be trained on 90% of the data and tested on 10%, and each run of the algorithm will change which 10% of the data the algorithm is tested on. Each data instance would be used as a training instance exactly 9 times and as a test instance 1 time. The accuracy will not be a mean and a standard deviation, but instead will be an exact accuracy score of how many correct predictions were made [46].

2.6.2 Data augmentation

Accessing labeled samples is often difficult because of the cost of acquiring them. This can limit the performance of supervised machine learning. To expand the effective training set, we can perform something called data augmentation. That is, we can synthetically generate new labeled samples from existing ones. A simple form of data augmentation is slightly modifying the sample data. Some common data augmentation techniques for audio include *time-shift*, *pitch-shift*, and *time-stretch*, all demonstrated in Figure 2.20. Data augmentation can be applied either to the raw audio waveform or to pre-processed spectrograms [18].

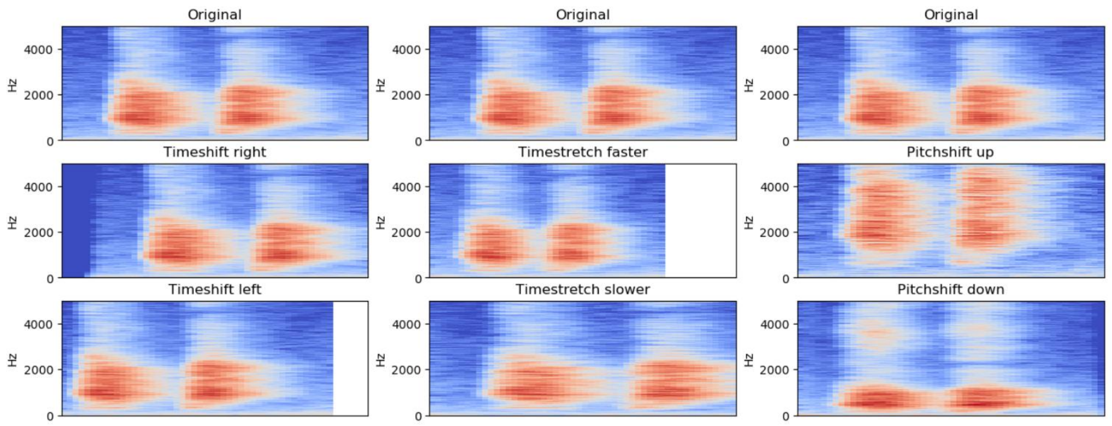


Figure 2.20 - Data augmentation methods for audio demonstrated on a dog bark. Figure shows linear-scaled spectrograms before and after applying the augmentation. The parameters are exaggerated to show the effects more clearly [18]

3 Methodology and Implementation

This chapter is dedicated to explaining the methodology used to pre-process the data and the implementation of that same methodology. If we do a good job pre-processing our audio, it makes it easier for machine learning models to tell the difference between our different classes. So, the first thing to address is the aspect of our data. For this, we have some programming tools at our disposal.

3.1 Software

When researching the topic of this thesis, it was found that most of the implementations of machine learning methods applied to sound event detection and classification were done in the programming language *Python*™ due to its ease of interpretation and ability to handle large data operations, so it was chosen early on that all code would be written in this language. It is a high-level language with a vast quantity of up-to-date libraries, making it the go-to language for machine learning.

Various *Python*™ packages were used in the programming of this project. They are detailed in the following paragraphs.

The *NumPy* package [47], due to its performance and simple implementation, is useful for data storing, math operations and matrix modifications.

The *libROSA* package [48], as it provides some useful functionalities for processing audio, turning the sound clips into *NumPy* arrays. It is also used to extract features from the clips, for creating spectrograms and extracting the MFCCs.

The *Pandas* package [49], to read the CSV file that links samples to the respective classes.

The *matplotlib* library [50], for plotting the various graphs and figures needed to study the dataset.

The *Sklearn* package [51], for a wide variety of tasks like dataset splitting into test and train, training various classification algorithms, and creating the confusion matrix.

The *TensorFlow* package with *Keras* interface [52], which is considered the most feature-rich machine learning tool in *Python*™.

The *joblib* library [53], to store the trained models so we don't have to run the models every time we want to use them.

The *PyAudio* package [54], used in audio signal analysis in real-time, importing the stream of audio.

The *datetime* library [55], to record the date and time of occurrence of the desired sound class in the log.

3.2 Dataset overview

The first thing we need to address is the data that will be used in our approach. The dataset to be used is the *UrbanSound8k* dataset [8] which consists of 8732 labeled sound excerpts (with 4 or less seconds of duration) of urban sounds from 10 classes:

- Air conditioner
- Car horn
- Children playing
- Dog bark
- Drilling
- Engine idling
- Gunshot
- Jackhammer
- Siren
- Street music

As we can see, these are classes of sounds one can find in an urban environment. Of the list above, only one class is present in each audio file, being that all files are in the .wav format.

The dataset is already shuffled and separated into 10 folds for a 10 fold cross-validation and the authors recommend not reshuffling the data during training due to the distribution of the sound classes within each fold. A single fold may contain multiple clips from the same source file, but the same source file is not used in multiple folds to prevent data leakage (sharing data between the train and the test sets).

The target sound is rarely the only sound event happening in the audio clip and might be in the background, partially obscured by sounds outside the available classes. This makes *UrbanSound8k* a relatively challenging dataset.

All classes have the same number of audio files, except for the siren, car horn and gunshot classes, which can be observed in Figure 3.1. The classes are then unbalanced and that is a problem that may need to be addressed.

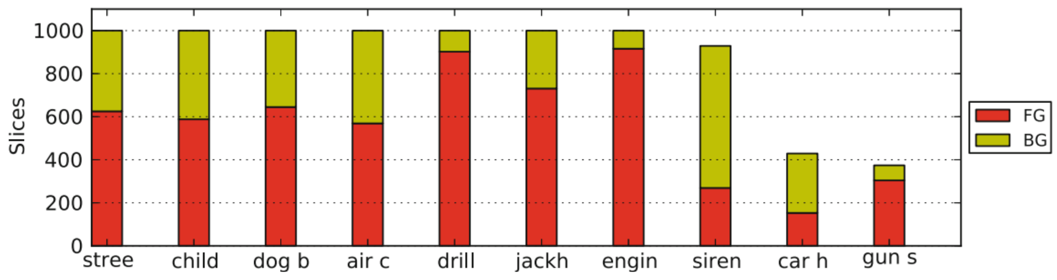


Figure 3.1 - Number of sound clips per class in the UrbanSound8K dataset with a breakdown by foreground (FG) and background (BG) [5]

Audio channels – as can be observed in Table 2, about 92% of the audio files are stereo, while the other 8% are mono.

Table 2 - Number of files sorted by number of audio channels

Number of channels	Number of files
1	739
2	7993

Sample rates – looking at the sample rates distribution in Table 3, we can see that there is a wide range of sample rates that have been used across all files, ranging from 8 kHz to 96 kHz, with most of the files being sampled at either 44.1 kHz or 48 kHz.

Table 3 - Number of files sorted by value of the sample rate

Sample rate	Number of files
44100	5370
48000	2502
96000	610
24000	82
16000	45
22050	44
11025	39
192000	17
8000	12
11024	7
32000	4

Bit-depth – looking at Table 4, we can see that there is also a large range of bit-depths (ranging from 4bit to 32bit).

Table 4 – Percentage of files sorted by bit-depth

Bit-depth	% of files
16	65,94%
24	31,53%
32	1,94%
8	0,49%
4	0,10%

We can observe that the dataset has a range of varying audio properties that will need standardizing.

The audio events in these sound excerpts were subjectively labeled as happening in the foreground or in the background. Being careful as to represent the audio events in the best way possible in the audio dataset exploration process, only sound files with events occurring in the foreground were used as examples. As such, one random file from each class was chosen and its waveform plotted. The results can be seen in the 10 graphs shown in Figure 3.2, with the y axis representing the amplitude of the sound wave and the x axis representing time.

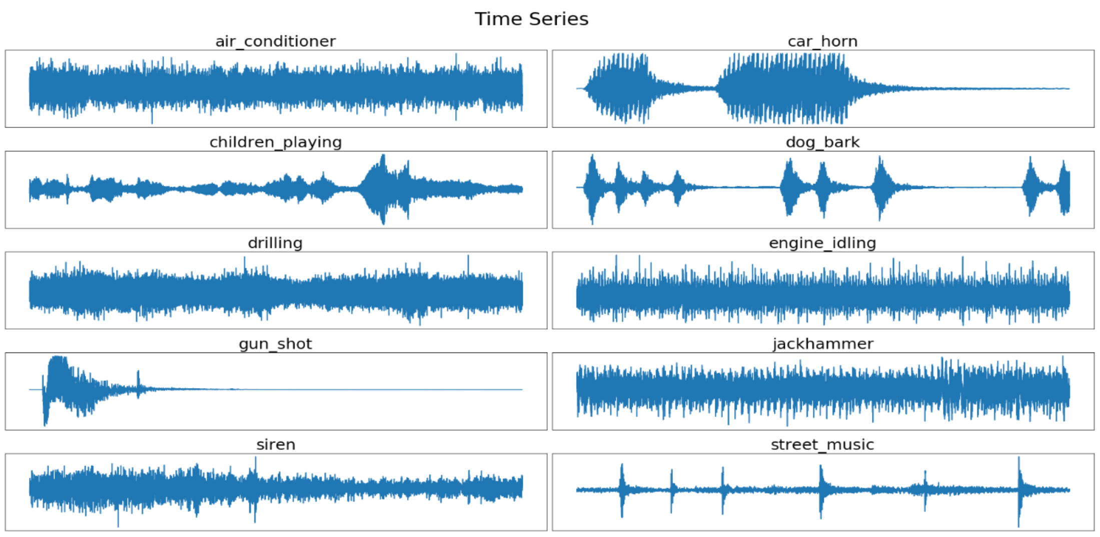


Figure 3.2 - Time series graphs for a random audio snippet of each class

If we look at the waveform representations, we can see that it is difficult to differentiate the classes, especially for repetitive sounds such as air conditioner, drilling, engine idling and jackhammer.

However, if we do the Short Time Fourier Transform (STFT) of the signals using the Fast Fourier Transform (FFT) algorithm, we get the periodograms in Figure 3.3, with the y axis representing the power and the x axis representing frequency. These periodograms are composed of the real values of the Fourier transform and go up to Nyquist frequency (half of its sampling rate) of the sample used.

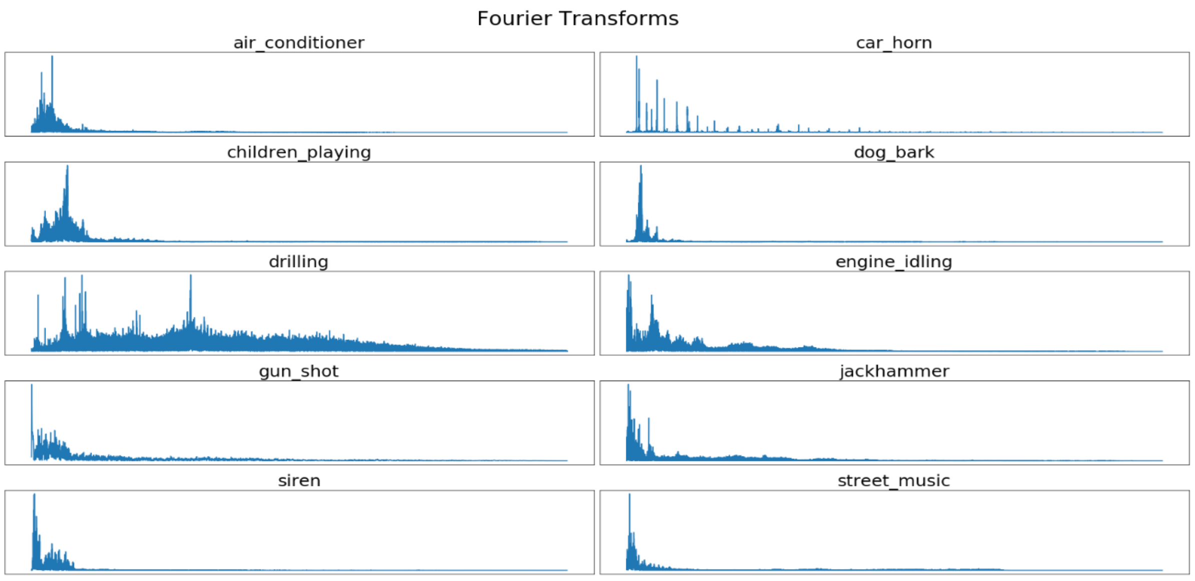


Figure 3.3 - Periodograms of each audio snippet represented in Figure 3.2

Analyzing the periodograms, we can see that most of the frequency content in the samples is present in the first half of the graph.

If we stack the periodograms adjacent to each other over time and color-code them to represent the amplitude, we obtain the spectrograms, represented in Figure 3.4. Every tenth of a second 4 periodograms are created and thus, we obtain an image that represents the frequency contents over time. This way, we can see how the signal changes over time. The sampling rate used is 16 kHz and the window length is of 25 ms, which gives us 400 samples. Since the step size is of 10 ms, that is equivalent to 160 samples. The size of FFT is of 512 samples, se we are going to zero-pad our 400 samples with 112 zeros. We use a Hanning window when we compute the FFT to reduce spectral leakage.

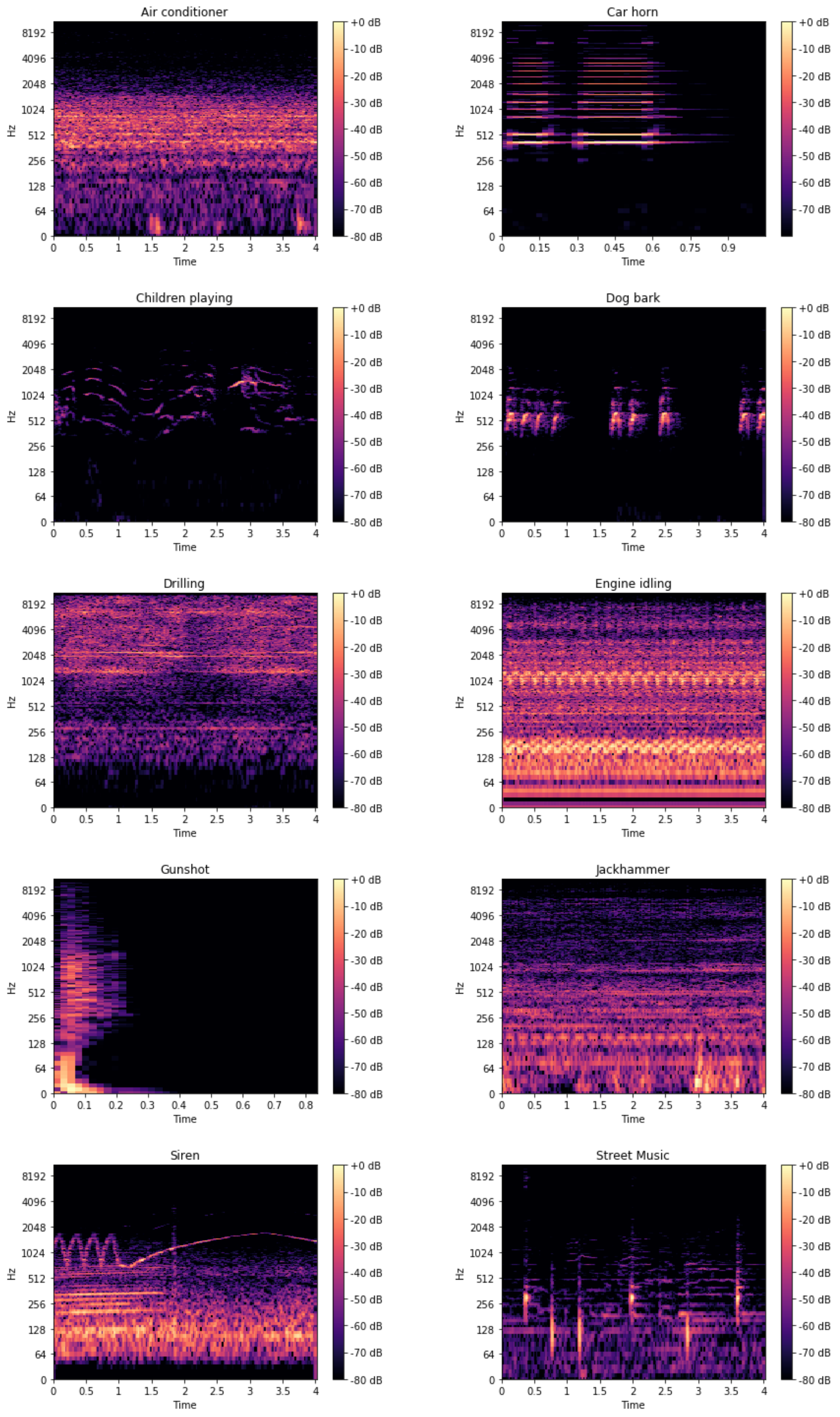


Figure 3.4 - Spectrograms of the selected samples

The spectrograms give us some idea about the frequencies present in the audio signal. However, these frequencies are too close and intertwined. We will re-bin our frequency content to make it accurately represent what a human might consider important in an audio signal. With MFCC we decorrelate these frequencies and modify this on a log scale, which is more relevant to how our ear perceives it. We also modify it on a Mel scale. The end result of this process are the MFCCs of the audio signal. We create a filter bank, with a standard 26 filters, with the spacing based on the mel scale. We get 26 triangular filters to bin our energies into. To obtain the MFCCs, we compute the Filter Bank Coefficients, plotted in Figure 3.5, with the y axis representing the frequency in mel scale and the x axis representing time.. We then compute the MFCCs.

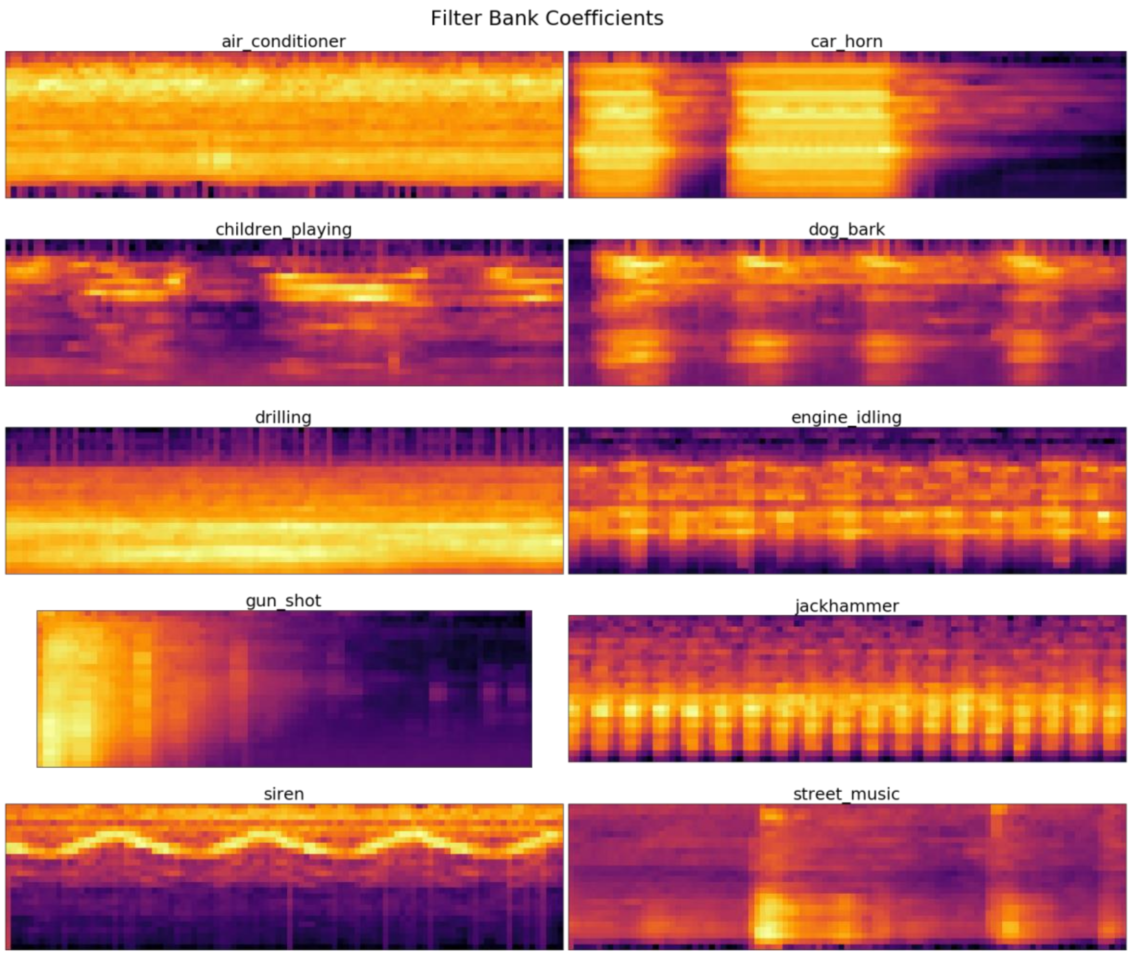


Figure 3.5 - Filter Bank Coefficients for one second of the 10 analyzed classes

Because we're using triangular filters that overlap and doing the STFT which has signals that inherently overlap during our calculation, what we're left with, with the filter bank energies, is with lots of values that are highly correlated. The dimensions of these filter banks, if we use 26 filters, for 1 second of data (rounds to 99), it should be a 26x100 matrix. We do one last step to decorrelate these energies: The Discrete Cosine Transform (DCT).

A visual depiction of the extracted MFCC Coefficients for one window of data is shown in Figure 3.6 , with the y axis representing the value of the coefficient and the x axis representing the index of the coefficient.

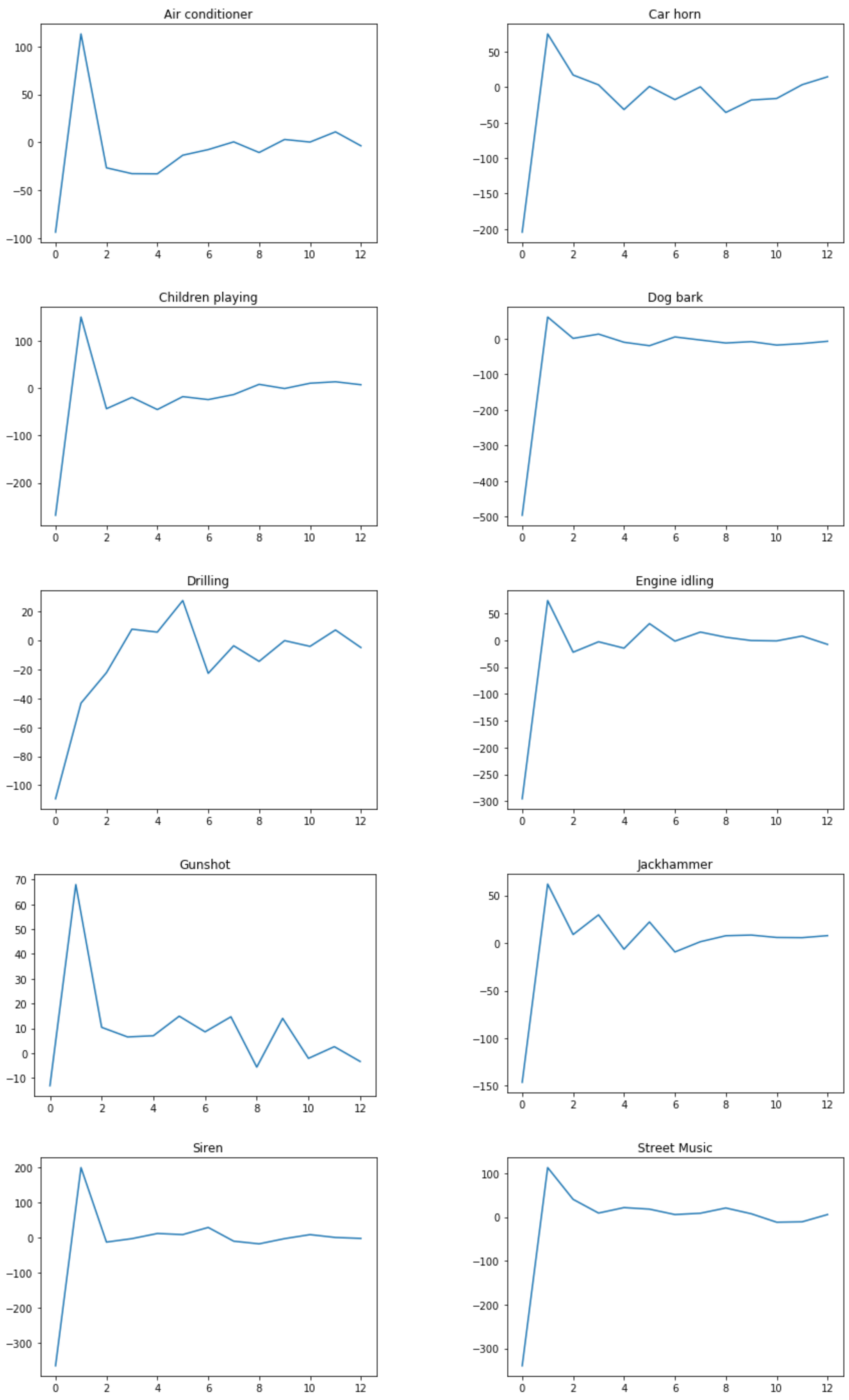


Figure 3.6 - Mel-Frequency Cepstral Coefficients for the first bin of every class

If we stack the windows as columns of an image over time, like we did when constructing the spectrograms, we obtain the graphs in Figure 3.7 , with the y axis representing the index of the coefficient and the x axis representing time.. These are for only a second of data.

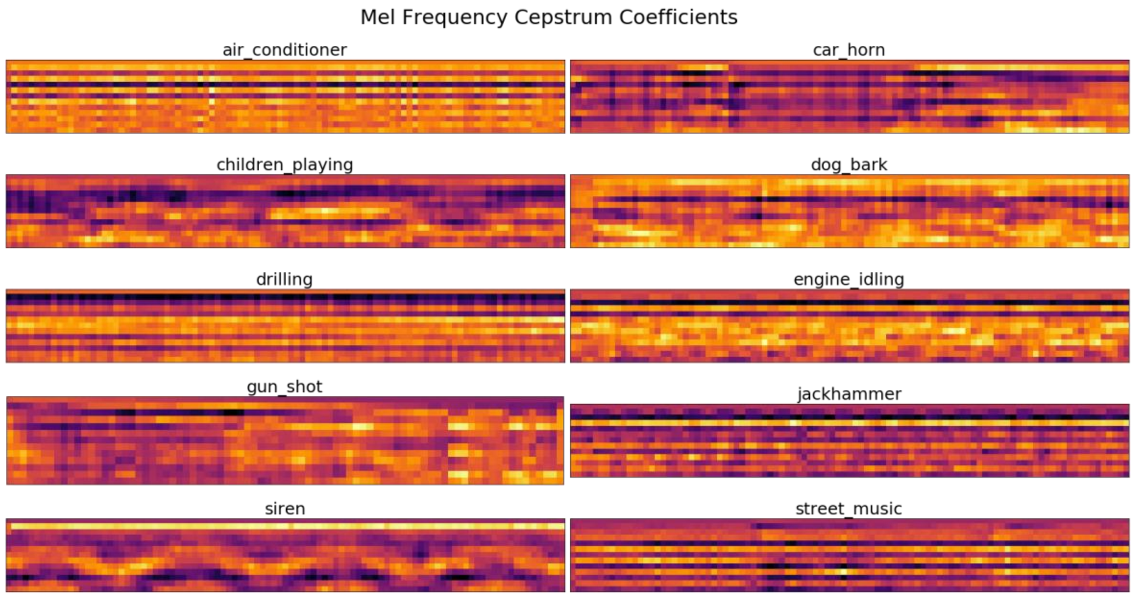


Figure 3.7 - Mel-Frequency Cepstrum Coefficients for one second of the 10 analysed classes

3.3 Train-test data splitting

We are going to randomly use 80% of the data for training and the remaining 20% for testing, since the number of data is sufficiently high. If that wasn't the case, the percentage of testing data should be higher than 20%.

3.4 Models used

All machine learning models talked in chapter 2.4 will be used to process the features extracted from the dataset.

For the Logistic Regression model, the One-vs-Rest approach was used on the preprocessed dataset.

For the SVM model, a linear kernel was implemented, and the multiclass classification was handled according to the One-vs-Rest scheme.

In the case of the Random Forest classifier, the number of decision trees used was 20 with a random state equal to 42. Bootstrapping was also used.

In the case of the k-Nearest Neighbors model, only one neighbor was used, i.e., k=1, since it revealed to be the best choice after some experimentation that exposed that any uneven number above 1 would wrongfully predict the class "Air conditioner" most of the times.

For the Artificial Neural Network, one hidden layer was used with a dropout of 0.5 and the sigmoid function as activation for the neurons. On the output layer, the *softmax* function was used as activation. A total of 1000 epochs were performed, giving us the best results of all the models.

Since after doing the research for Chapter 2 there seemed to be no actual consensus on what the number of MFCCs to use is, the author of this thesis chose to experiment with all the values that have been used in similar projects. Those values are 10, 13, 20 and 40 MFCCs. With this range of values, we can have a broader look at their influence when fed to a machine learning classification model.

The results can be consulted in chapter 4.

4 Results and Discussion

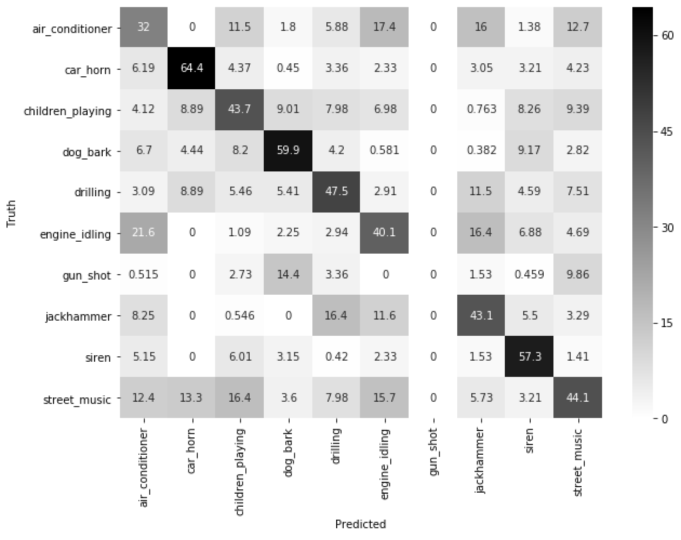
4.1 Classification results

Since we have a somewhat imbalanced dataset, it's easy to get a high value of accuracy without actually making useful predictions because it only makes sense to use accuracy as an evaluation metric if the class labels are uniformly distributed. Hence, a confusion matrix is the best technique to assess the performance of the model.

Every confusion matrix will be normalized in such a way as to give the precision for every class in its diagonal, because the focus is to minimize the false positives, which is what precision can show. The correct predictions are along the diagonal, with the misclassifications on the off-diagonal.

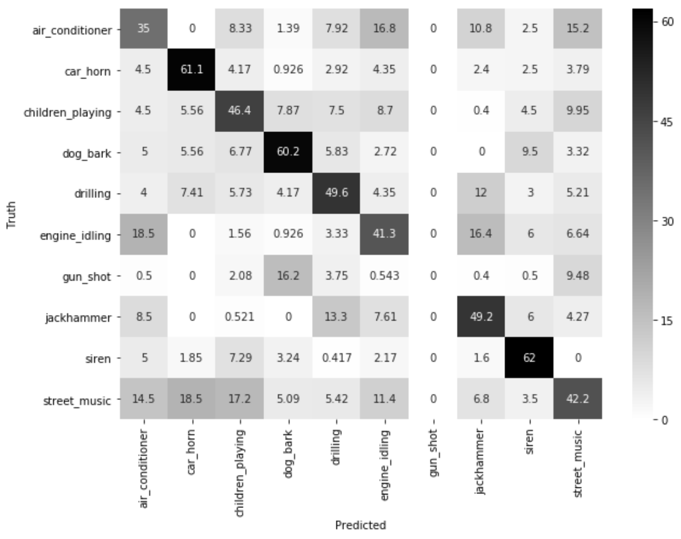
4.1.1 Logistic Regression

The confusion matrices for the LR model using 10, 13, 20 and 40 MFCCs are displayed in Figure 4.1 a), b), c) and d), respectively.



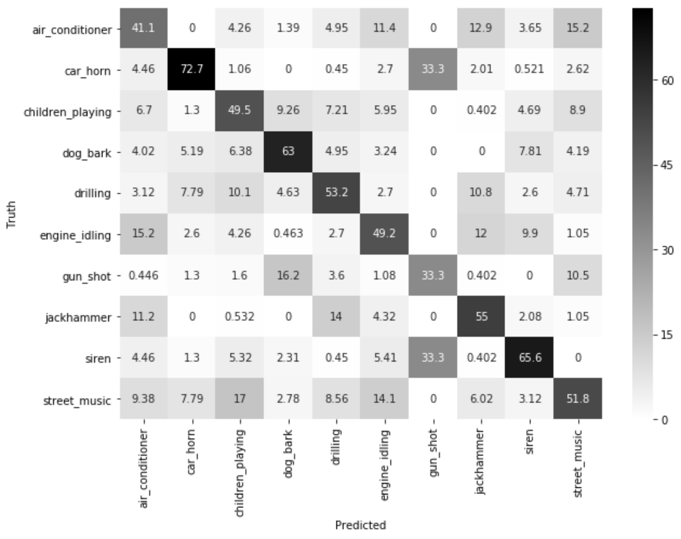
Truth \ Predicted	air_conditioner	car_horn	children_playing	dog_bark	drilling	engine_idling	gun_shot	jackhammer	siren	street_music
air_conditioner	32	0	11.5	1.8	5.88	17.4	0	16	1.38	12.7
car_horn	6.19	64.4	4.37	0.45	3.36	2.33	0	3.05	3.21	4.23
children_playing	4.12	8.89	43.7	9.01	7.98	6.98	0	0.763	8.26	9.39
dog_bark	6.7	4.44	8.2	59.9	4.2	0.581	0	0.382	9.17	2.82
drilling	3.09	8.89	5.46	5.41	47.5	2.91	0	11.5	4.59	7.51
engine_idling	21.6	0	1.09	2.25	2.94	40.1	0	16.4	6.88	4.69
gun_shot	0.515	0	2.73	14.4	3.36	0	0	1.53	0.459	9.86
jackhammer	8.25	0	0.546	0	16.4	11.6	0	43.1	5.5	3.29
siren	5.15	0	6.01	3.15	0.42	2.33	0	1.53	57.3	1.41
street_music	12.4	13.3	16.4	3.6	7.98	15.7	0	5.73	3.21	44.1

a)



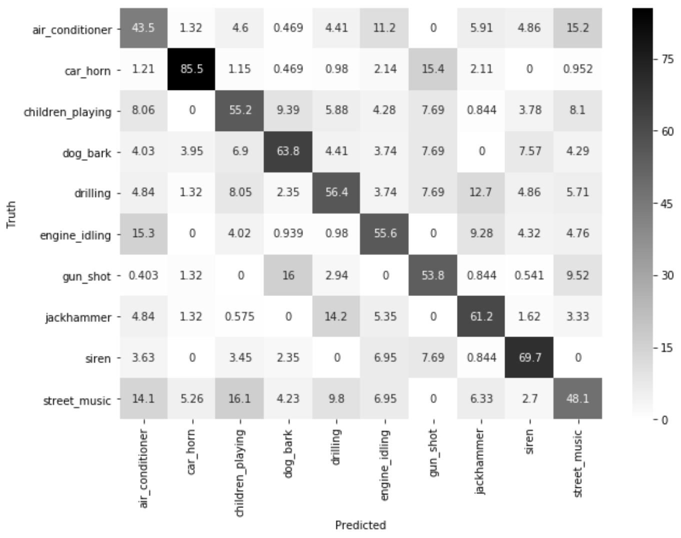
	air_conditioner	car_horn	children_playing	dog_bark	drilling	engine_idling	gun_shot	jackhammer	siren	street_music
air_conditioner	35	0	8.33	1.39	7.92	16.8	0	10.8	2.5	15.2
car_horn	4.5	61.1	4.17	0.926	2.92	4.35	0	2.4	2.5	3.79
children_playing	4.5	5.56	46.4	7.87	7.5	8.7	0	0.4	4.5	9.95
dog_bark	5	5.56	6.77	60.2	5.83	2.72	0	0	9.5	3.32
drilling	4	7.41	5.73	4.17	49.6	4.35	0	12	3	5.21
engine_idling	18.5	0	1.56	0.926	3.33	41.3	0	16.4	6	6.64
gun_shot	0.5	0	2.08	16.2	3.75	0.543	0	0.4	0.5	9.48
jackhammer	8.5	0	0.521	0	13.3	7.61	0	49.2	6	4.27
siren	5	1.85	7.29	3.24	0.417	2.17	0	1.6	62	0
street_music	14.5	18.5	17.2	5.09	5.42	11.4	0	6.8	3.5	42.2

b)



	air_conditioner	car_horn	children_playing	dog_bark	drilling	engine_idling	gun_shot	jackhammer	siren	street_music
air_conditioner	41.1	0	4.26	1.39	4.95	11.4	0	12.9	3.65	15.2
car_horn	4.46	72.7	1.06	0	0.45	2.7	33.3	2.01	0.521	2.62
children_playing	6.7	1.3	49.5	9.26	7.21	5.95	0	0.402	4.69	8.9
dog_bark	4.02	5.19	6.38	63	4.95	3.24	0	0	7.81	4.19
drilling	3.12	7.79	10.1	4.63	53.2	2.7	0	10.8	2.6	4.71
engine_idling	15.2	2.6	4.26	0.463	2.7	49.2	0	12	9.9	1.05
gun_shot	0.446	1.3	1.6	16.2	3.6	1.08	33.3	0.402	0	10.5
jackhammer	11.2	0	0.532	0	14	4.32	0	55	2.08	1.05
siren	4.46	1.3	5.32	2.31	0.45	5.41	33.3	0.402	65.6	0
street_music	9.38	7.79	17	2.78	8.56	14.1	0	6.02	3.12	51.8

c)



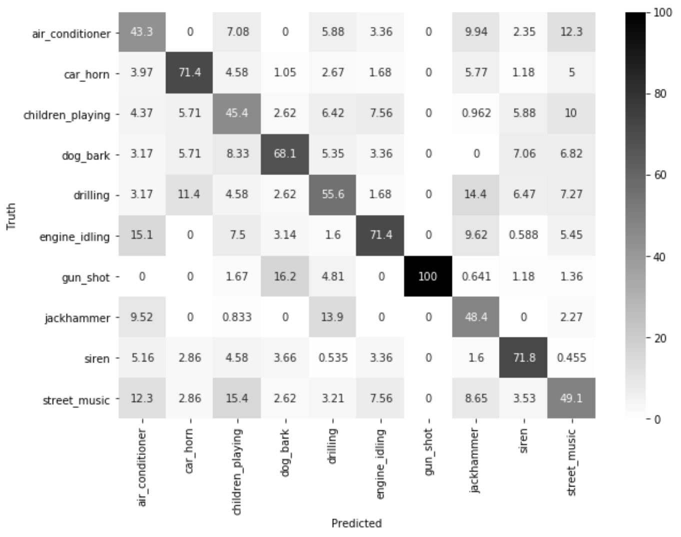
	air_conditioner	car_horn	children_playing	dog_bark	drilling	engine_idling	gun_shot	jackhammer	siren	street_music
air_conditioner	43.5	1.32	4.6	0.469	4.41	11.2	0	5.91	4.86	15.2
car_horn	1.21	85.5	1.15	0.469	0.98	2.14	15.4	2.11	0	0.952
children_playing	8.06	0	55.2	9.39	5.88	4.28	7.69	0.844	3.78	8.1
dog_bark	4.03	3.95	6.9	63.8	4.41	3.74	7.69	0	7.57	4.29
drilling	4.84	1.32	8.05	2.35	56.4	3.74	7.69	12.7	4.86	5.71
engine_idling	15.3	0	4.02	0.939	0.98	55.6	0	9.28	4.32	4.76
gun_shot	0.403	1.32	0	16	2.94	0	53.8	0.844	0.541	9.52
jackhammer	4.84	1.32	0.575	0	14.2	5.35	0	61.2	1.62	3.33
siren	3.63	0	3.45	2.35	0	6.95	7.69	0.844	69.7	0
street_music	14.1	5.26	16.1	4.23	9.8	6.95	0	6.33	2.7	48.1

d)

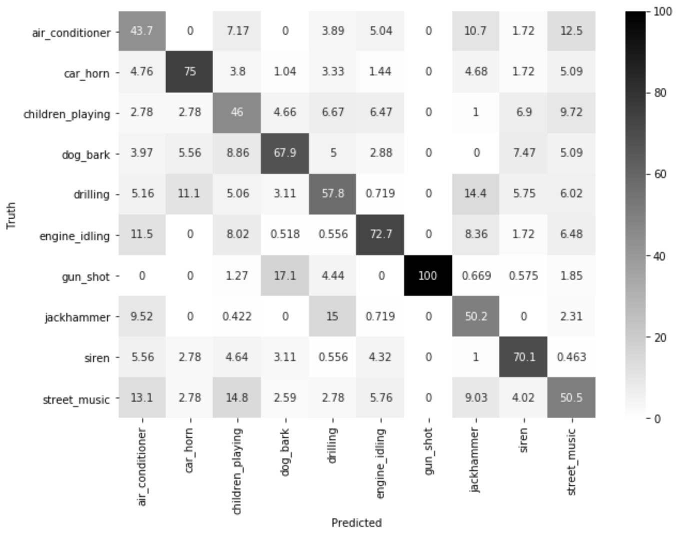
Figure 4.1 - Confusion matrices for the LR model using **a)** 10, **b)** 13, **c)** 20 and **d)** 40 MFCCs

4.1.2 Support Vector Machines

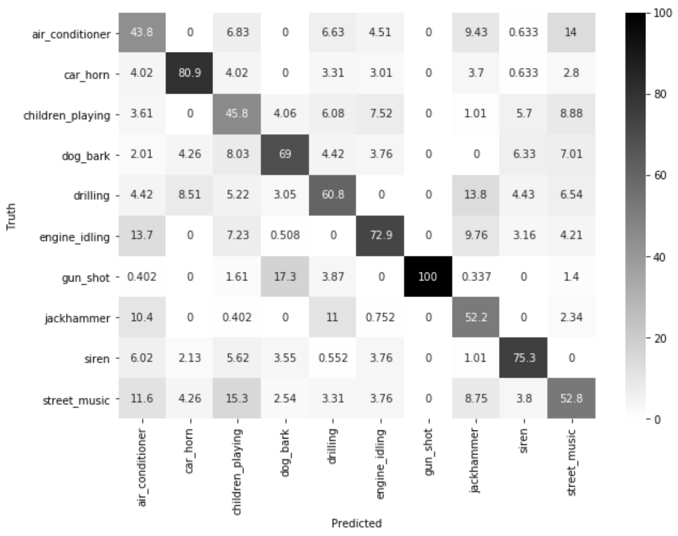
The confusion matrices for the SVM model using 10, 13, 20 and 40 MFCCs are displayed in Figure 4.2 a), b), c) and d), respectively.



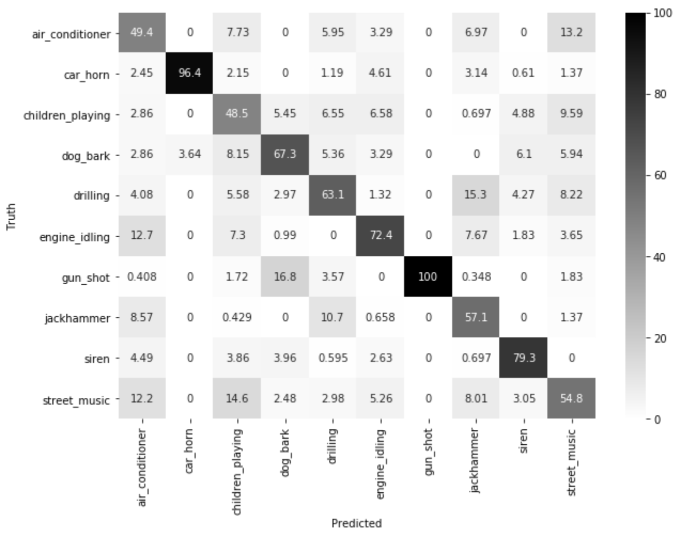
a)



b)



c)

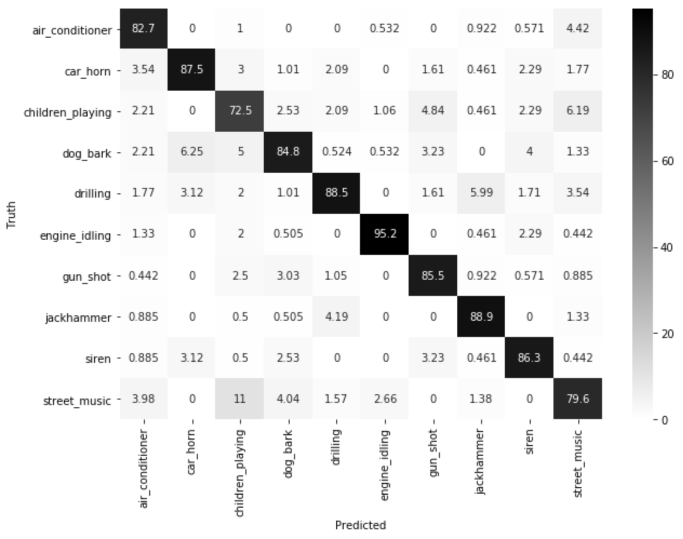


d)

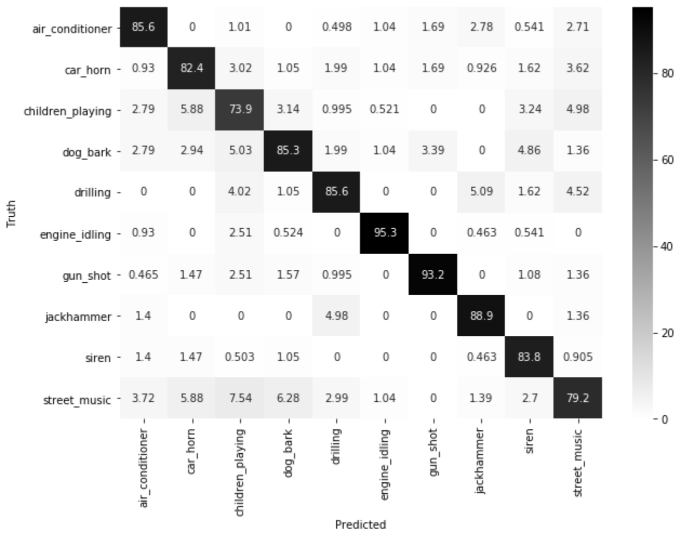
Figure 4.2 - Confusion matrices for the SVM model using **a)** 10, **b)** 13, **c)** 20 and **d)** 40 MFCCs

4.1.3 Random Forests

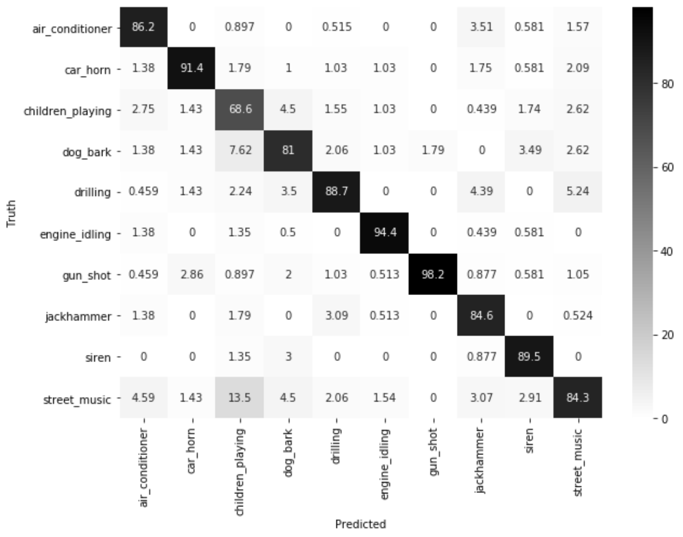
The confusion matrices for the RF model using 10, 13, 20 and 40 MFCCs are displayed in Figure 4.3 a), b), c) and d), respectively.


Truth
Predicted
air_conditioner
car_horn
children_playing
dog_bark
drilling
engine_idling
gun_shot
jackhammer
siren
street_music

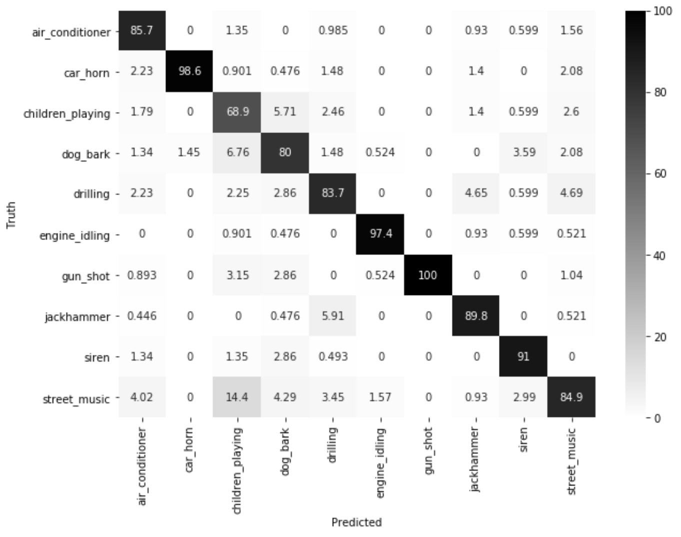
a)


Truth
Predicted
air_conditioner
car_horn
children_playing
dog_bark
drilling
engine_idling
gun_shot
jackhammer
siren
street_music

b)


Truth
Predicted
air_conditioner
car_horn
children_playing
dog_bark
drilling
engine_idling
gun_shot
jackhammer
siren
street_music

c)

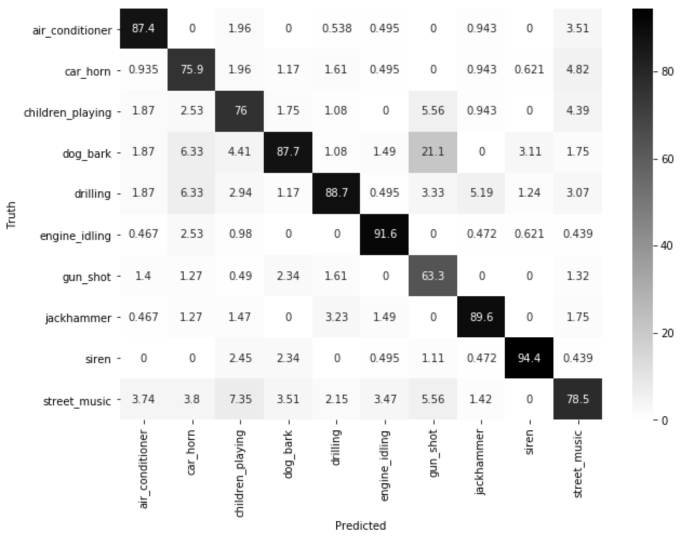


d)

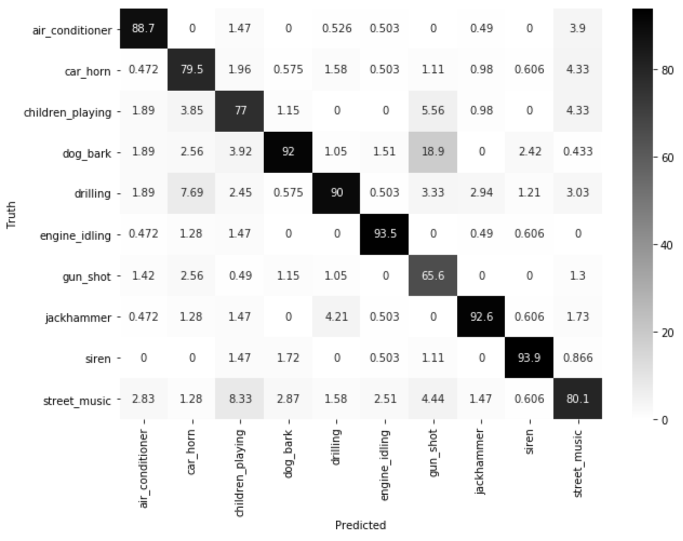
Figure 4.3 - Confusion matrices for the RF model using **a)** 10, **b)** 13, **c)** 20 and **d)** 40 MFCCs

4.1.4 Nearest Neighbor

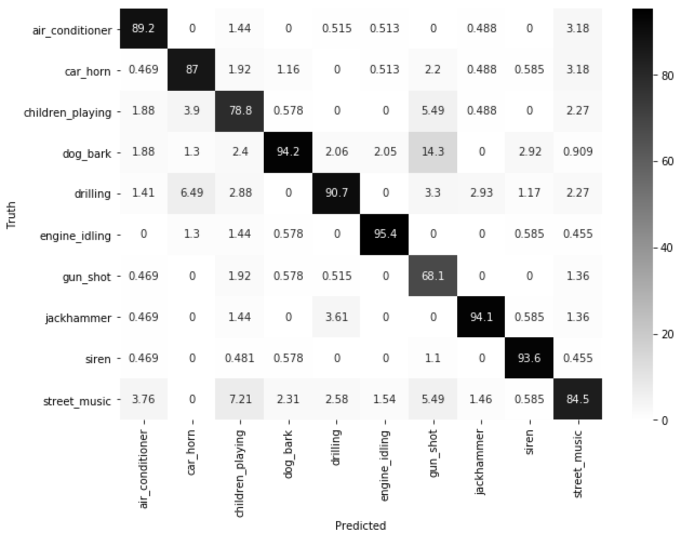
The confusion matrices for the NN model using 10, 13, 20 and 40 MFCCs are displayed in Figure 4.4 a), b), c) and d), respectively.



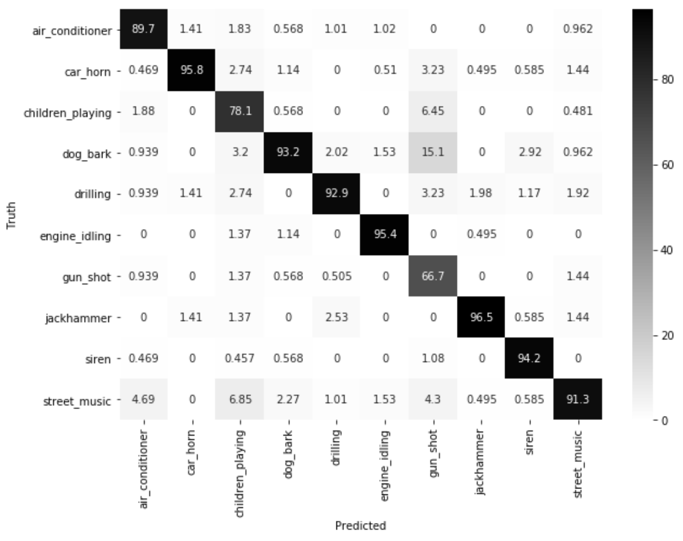
a)



b)



c)

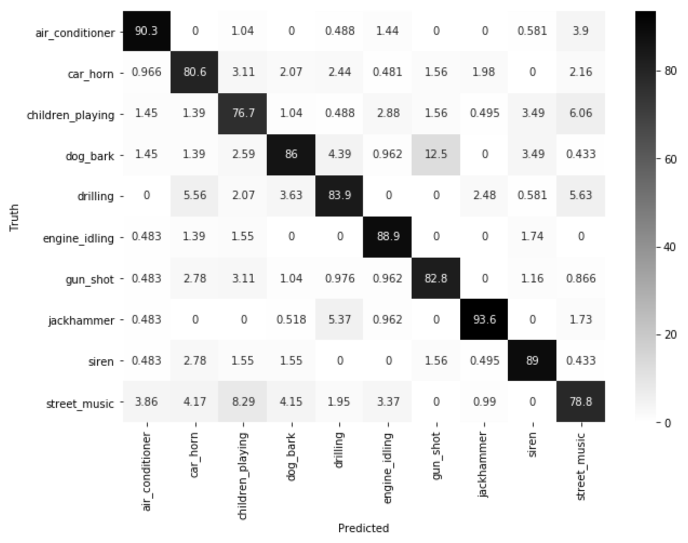


d)

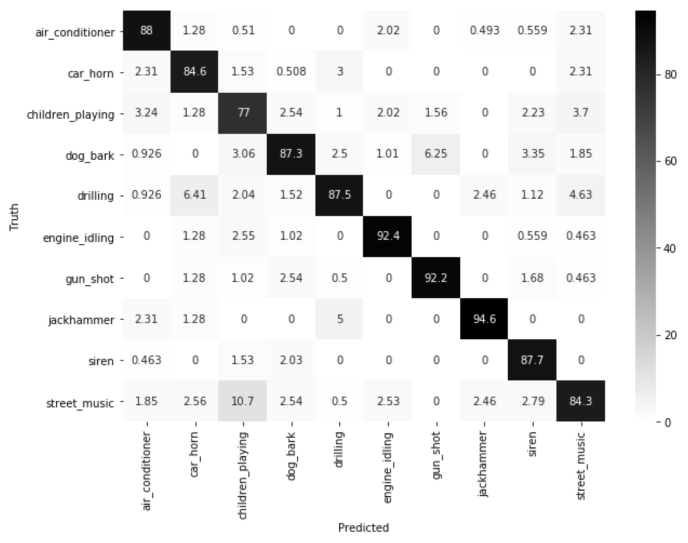
Figure 4.4 - Confusion matrices for the NN model using **a)** 10, **b)** 13, **c)** 20 and **d)** 40 MFCCs

4.1.5 Artificial Neural Network

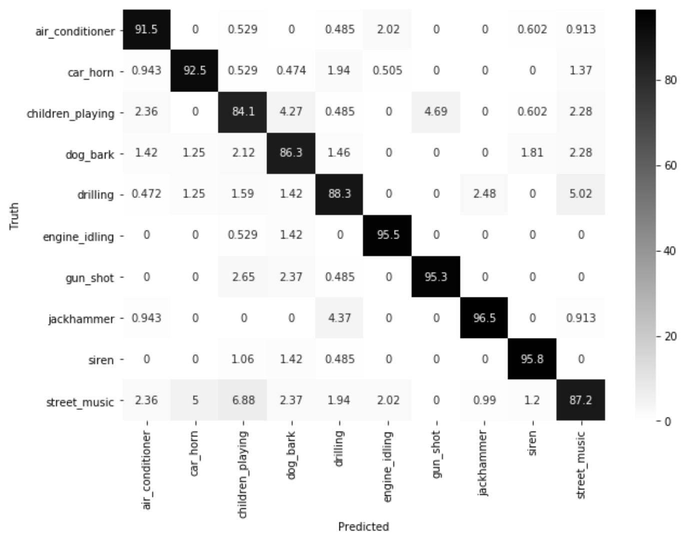
The confusion matrices for the ANN model using 10, 13, 20 and 40 MFCCs are displayed in Figure 4.5 a), b), c) and d), respectively.



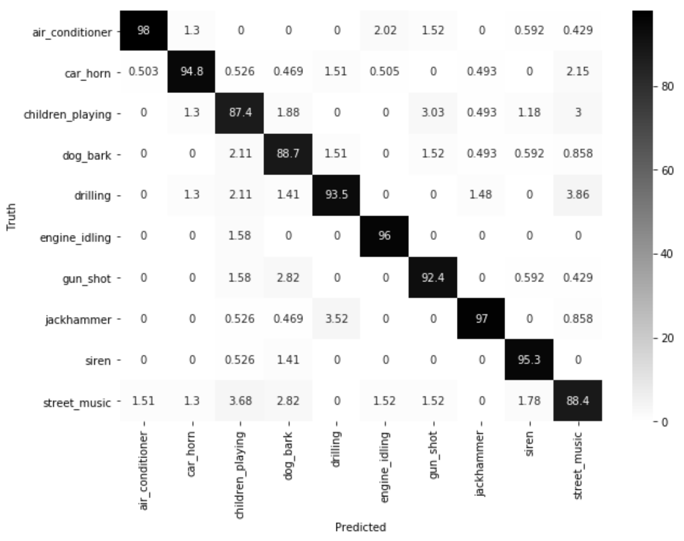
a)



b)



c)



	air_conditioner	car_horn	children_playing	dog_bark	drilling	engine_idling	gun_shot	jackhammer	siren	street_music
air_conditioner	98	1.3	0	0	0	2.02	1.52	0	0.592	0.429
car_horn	0.503	94.8	0.526	0.469	1.51	0.505	0	0.493	0	2.15
children_playing	0	1.3	87.4	1.88	0	0	3.03	0.493	1.18	3
dog_bark	0	0	2.11	88.7	1.51	0	1.52	0.493	0.592	0.858
drilling	0	1.3	2.11	1.41	93.5	0	0	1.48	0	3.86
engine_idling	0	0	1.58	0	0	96	0	0	0	0
gun_shot	0	0	1.58	2.82	0	0	92.4	0	0.592	0.429
jackhammer	0	0	0.526	0.469	3.52	0	0	97	0	0.858
siren	0	0	0.526	1.41	0	0	0	0	95.3	0
street_music	1.51	1.3	3.68	2.82	0	1.52	1.52	0	1.78	88.4

d)

Figure 4.5 - Confusion matrices for the ANN model using **a)** 10, **b)** 13, **c)** 20 and **d)** 40 MFCCs

4.1.6 Model performance metrics

The results of the model performance metrics for each model can be consulted in the following tables.

Table 5 - Accuracy and F-scores for each class using Logistic Regression

	10 MFCCs **Accuracy** 0.468	13 MFCCs **Accuracy** 0.488	20 MFCCs **Accuracy** 0.543	40 MFCCs **Accuracy** 0.576
Air conditioner	0.312	0.347	0.431	0.479
Car horn	0.443	0.471	0.687	0.802
Children playing	0.437	0.475	0.501	0.539
Dog bark	0.629	0.624	0.652	0.657
Drilling	0.509	0.534	0.551	0.561
Engine idling	0.378	0.403	0.481	0.547
Gunshot	0.000	0.000	0.027	0.165
Jackhammer	0.481	0.537	0.600	0.652
Siren	0.653	0.679	0.706	0.737
Street music	0.424	0.404	0.470	0.459

Table 6 - Accuracy and F-scores for each class using Support Vector Machines

	10 MFCCs **Accuracy** 0.552	13 MFCCs **Accuracy** 0.563	20 MFCCs **Accuracy** 0.580	40 MFCCs **Accuracy** 0.615
Air conditioner	0.479	0.484	0.482	0.540
Car horn	0.413	0.443	0.571	0.752
Children playing	0.515	0.519	0.528	0.543
Dog bark	0.663	0.665	0.683	0.675
Drilling	0.529	0.539	0.568	0.567
Engine idling	0.545	0.608	0.595	0.638
Gunshot	0.452	0.452	0.468	0.468
Jackhammer	0.581	0.592	0.614	0.663
Siren	0.728	0.720	0.737	0.790
Street music	0.480	0.489	0.509	0.535

Table 7 - Accuracy and F-scores for each class using Random Forests

	10 MFCCs **Accuracy** 0.848	13 MFCCs **Accuracy** 0.848	20 MFCCs **Accuracy** 0.851	40 MFCCs **Accuracy** 0.858
Air conditioner	0.872	0.880	0.893	0.899
Car horn	0.747	0.727	0.821	0.877
Children playing	0.757	0.770	0.754	0.756
Dog bark	0.842	0.832	0.808	0.818
Drilling	0.851	0.845	0.860	0.831
Engine idling	0.940	0.951	0.948	0.969
Gunshot	0.791	0.840	0.859	0.857
Jackhammer	0.908	0.906	0.885	0.913
Siren	0.888	0.886	0.914	0.916
Street music	0.789	0.776	0.765	0.773

Table 8 - Accuracy and F-scores for each class using Nearest Neighbor

	10 MFCCs **Accuracy** 0.847	13 MFCCs **Accuracy** 0.865	20 MFCCs **Accuracy** 0.886	40 MFCCs **Accuracy** 0.900
Air conditioner	0.897	0.906	0.913	0.918
Car horn	0.727	0.756	0.822	0.866
Children playing	0.801	0.811	0.839	0.851
Dog bark	0.806	0.853	0.872	0.870
Drilling	0.842	0.864	0.880	0.911
Engine idling	0.937	0.949	0.959	0.961
Gunshot	0.704	0.728	0.761	0.752
Jackhammer	0.905	0.917	0.935	0.951
Siren	0.933	0.939	0.952	0.958
Street music	0.782	0.803	0.827	0.868

Table 9 - Accuracy and F-scores for each class using Artificial Neural Network

	10 MFCCs **Accuracy** 0.855	13 MFCCs **Accuracy** 0.874	20 MFCCs **Accuracy** 0.908	40 MFCCs **Accuracy** 0.930
Air conditioner	0.912	0.907	0.935	0.970
Car horn	0.734	0.805	0.892	0.896
Children playing	0.787	0.797	0.855	0.890
Dog bark	0.843	0.864	0.883	0.913
Drilling	0.837	0.862	0.883	0.919
Engine idling	0.923	0.936	0.967	0.972
Gunshot	0.779	0.868	0.897	0.884
Jackhammer	0.922	0.934	0.951	0.959
Siren	0.908	0.913	0.961	0.964
Street music	0.790	0.816	0.851	0.890

4.2 Real-time audio classification

In order to perform classification in real time, an audio stream has to be evaluated in chunks by the model, which will output a prediction between 0 and 1 of how certain it is of the classification it is doing. For this, we will be using a context window of 4 seconds, obtain 40 MFCCs for each chunk of the window of 4 seconds, make a medium of each coefficient and feed them to the classifier. The classifier chosen was the best performing one, the Artificial Neural Network. The result is a system that can predict in real time the sound it is being fed through the microphone of the laptop (even though any microphone would work the same) [1]. The final aspect of this system is in Figure 4.6, with the system in between predictions.

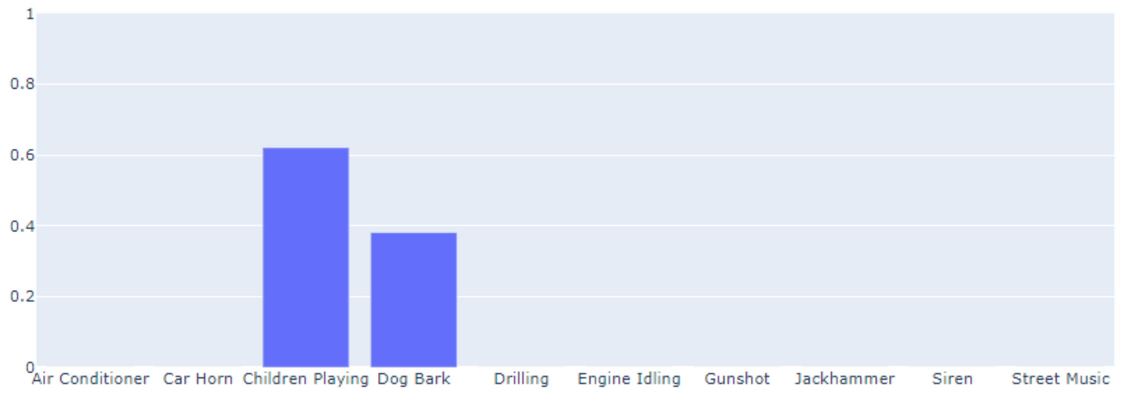
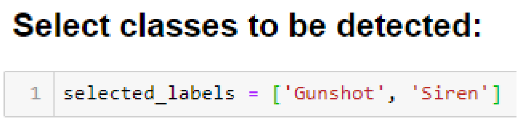


Figure 4.6 - Final aspect of window of predictions, with two labels being predicted as likely occurring

The classes of sounds to be identified can be changed in the notebook, depending on the desired use, as seen in Figure 4.7. In Figure 4.8 we can see the aspect of the system when only the classes “Gunshot” and “Siren” are selected.

[1] The *Jupyter Notebooks* used to build the prototype of this project can be consulted in the following GitHub repository: https://github.com/up201305898/thesis



Select classes to be detected:

```
selected_labels = ['Gunshot', 'Siren']
```

Figure 4.7 - Selecting the desired classes to be detected

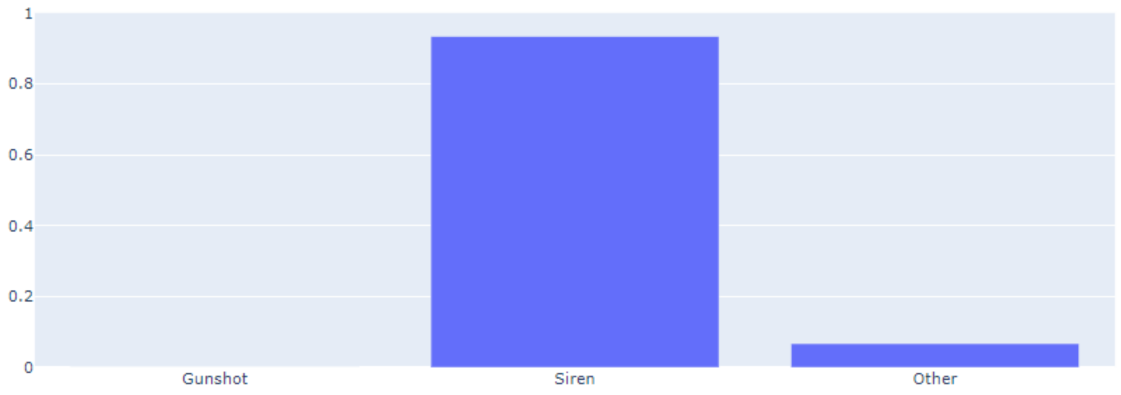
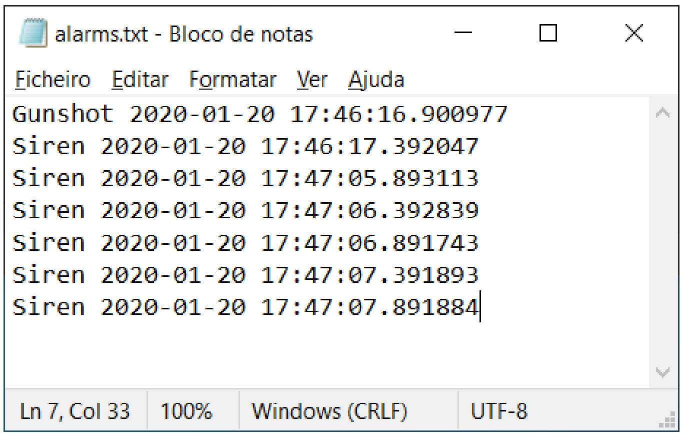


Figure 4.8 - Result of selecting the desired classes to be detected

In addition to this feature, a log is created (file “alarms.txt”) that registers the occurrence of any of the desired classes. This log checks every half a second if the prediction for the given classes is over 0.5 and registers that occurrence in the .txt file, as shown in Figure 4.9.


alarms.txt - Bloco de notas
Ficheiro Editar Formatar Ver Ajuda
Ln 7, Col 33
100%
Windows (CRLF)
UTF-8

```
Gunshot 2020-01-20 17:46:16.900977
Siren 2020-01-20 17:46:17.392047
Siren 2020-01-20 17:47:05.893113
Siren 2020-01-20 17:47:06.392839
Siren 2020-01-20 17:47:06.891743
Siren 2020-01-20 17:47:07.391893
Siren 2020-01-20 17:47:07.891884
```

Figure 4.9 - Log of the chosen classes' detection by the system

4.3 Discussion

Looking at the classification matrices of all models using the chosen number of MFCCs, one can quickly tell that the worst performers are the Logistic Regression and the Support Vector Machines models. One example of this bad performance is the misclassification of the “Gunshot” sound for every one of the test samples when feeding 10 or 13 MFCCs to the LR model. On the other hand, the SVM model predicted correctly this class 100% of the times. Despite this, both models’ classification matrices present a lot of cases of misclassification on all classes, regardless of the number of MFCCs used.

On the other hand, the remaining three classification models performed well on the test samples, as evidenced by the high contrast between the diagonal of their confusion matrices and the off-diagonal of them. As such, all three models were considered for the real-time implementation.

Despite all this, it can easily be seen through the classification matrices and F-scores from the various models that the model that overall performs better is the Artificial Neural

Network when using 40 MFCCs as input. This was somewhat to be expected, as we are giving the largest number of parameters to the most complex model. As such, it will be this model that will be used to create the real time audio classification system.

Despite being able to work out a functioning prototype of what an audio-surveillance system would be like, the results are far from ideal, in part due to the absence of more sophisticated microphones that wouldn’t induce so much noise (and thus, increase the SNR) in the stream of audio fed to the classifier.

With that in mind, it could be observed that some of the classes did not appear to be recognizable by the prototype, such as “Jackhammer” and “Air Conditioner”. As such, these results could be due to the fact that both of these classes are perceived by the model as an extremely specific kind of noise that can only be reproduceable with the same machines used to construct the dataset.

5 Conclusions and Future Work

5.1 Conclusions

The main objective of this project was to find a way to classify urban environment sound events using machine learning tools for an automated surveillance system. To do so, five machine learning algorithms were tested: Logistic Regression, Support Vector Machines, Random Forests, Nearest Neighbor and Artificial Neural Networks. The features fed to these machine learning algorithms were Mel-Frequency Cepstral Coefficients (MFCCs). A total of four sets of extracted features were given to each model: one with 10 MFCCs, one with 13, one with 20 and one with 40.

All models, with the exception of Logistic Regression and Support Vector Machines, revealed a satisfactory performance when trained with 80% of the data and tested with the remaining 20%. This can be corroborated by the confusion matrices for the following three models: Random Forests, Nearest Neighbor and Artificial Neural Network. These models generalized well and presented accuracies on the validation sets of over 85%.

The best performing model was the Artificial Neural Networks one when used in consolidation with 40 MFCCs. As such, the implementation used exactly these two parameters to develop a system capable of recognizing a sound that was fed into the laptop's microphone.

The resulting multiclass audio classification system revealed to be slightly flawed due to the training set being composed of samples that included a lot of background (and even foreground) noise. Because of that, the predictions for some of the classes are often wrong, with a potential for misclassifying a certain unknown sound as desired class. All other features of the classification system were satisfying: the latency between the occurrence of the sound event and the registration in the log was pretty much nonexistent, resulting in a somewhat robust audio event classifier that could be used in an audio-surveillance system.

Another cause of concern is the noise introduced by the microphones used to test the system with. Every single one of them compressed the audio in such a way that the sound captured was always somewhat masked behind a constant "hum". This low signal-to-noise ratio has caused several sound classes to not be recognized in the testing phase.

The one sound class that revealed itself to be problematic was the "Dog Barking" one. Upon further inspection into the audio samples used to train the machine learning models, one can have the intuition that the problem stems from the fact that most of the "Dog Barking" samples are comprised of silence or background noise in between the barks. That way, whenever the audio classifier is exposed to silence, it classifies it as a dog bark, even though no sound is present. This happens because our dataset is a multiclass dataset, where each of the samples has only one label.

Since the proof of concept in this project was successful and the system could be seen working with some robustness, the author of this thesis believes that the approach made in this project can be expanded into a system that detects any sound that is required by the user by adding those same classes of sounds to the dataset or even using another dataset entirely with similar dimensions.

5.2 Future Work

To improve the results obtained when running the *notebook*, some optimization of the machine learning models could be done through k-fold cross validation, data augmentation and hyperparameter tuning. The use of more sophisticated microphones, with better sound capturing performance, instead of the microphones used could also see the improvement of the classification results.

Deep learning methods are also a promising path to follow in order to attain better results with the selected dataset. One could even use image recognition algorithms on the MFCC graphs obtained through the audio stream to classify the sound present. Feature learning, which is a set of techniques that allows a system to automatically discover the representations needed for feature detection or classification from raw data, can also be employed to try and improve the sound detection system.

If a multiclass audio classification system is not enough for the desired use, one can also turn to multilabel audio classification, in which a given audio stream is inspected for the simultaneous occurrence of a group of classes of sounds.

Finally, a user-friendly graphical user interface (GUI) could be developed to be utilized by someone without experience in coding in *Python*™ language in order to easily change the variables of the system and adapt it to the person's desired use.

References

[1] L. Edwards, *Privacy, Security and Data Protection in Smart Cities: a Critical EU Law Perspective*. 2015.

[2] The World Bank Group. "Urban population (% of total population)." https://data.worldbank.org/indicator/SP.URB.TOTL.in.zs (accessed October, 2019).

[3] Population Division of the United Nations Department of Economic and Social Affairs (UN DESA). "68% of the world population projected to live in urban areas by 2050, says UN." https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html (accessed October, 2019).

[4] European Comission - Knowledge for policy. "Worldwide urban population growth." https://ec.europa.eu/knowledge4policy/foresight/topic/continuing-urbanisation/worldwide-urban-population-growth_en (accessed October, 2019).

[5] J. P. Bello, C. Mydlarz, and J. Salamon, "Sound Analysis in Smart Cities," in *Computational Analysis of Sound Scenes and Events*, T. Virtanen, M. D. Plumbley, and D. Ellis Eds. Cham: Springer International Publishing, 2018, pp. 373-397.

[6] M. Crocco, M. Cristani, A. Trucco, and V. Murino, "Audio surveillance: A systematic review," *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, p. 52, 2016.

[7] C. Mydlarz, M. Sharma, Y. Lockerman, B. Steers, C. Silva, and J. P. Bello, "The Life of a New York City Noise Sensor Network," *Sensors (Switzerland)*, vol. 19, 03/02 2019, doi: 10.3390/s19061415.

[8] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014: ACM, pp. 1041-1044.

[9] F. Font, G. Roma, and X. Serra, "Freesound technical demo," in *Proceedings of the 21st ACM international conference on Multimedia*, 2013, pp. 411-412.

[10] M. Valera and S. A. Velastin, "Intelligent distributed surveillance systems: a review," *IEE Proceedings-Vision, Image and Signal Processing*, vol. 152, no. 2, pp. 192-204, 2005.

[11] J. Chen, A. H. Kam, J. Zhang, N. Liu, and L. Shue, "Bathroom activity monitoring based on sound," in *International Conference on Pervasive Computing*, 2005: Springer, pp. 47-61.

[12] J. P. Bello *et al.*, "SONYC: A system for the monitoring, analysis and mitigation of urban noise pollution," *arXiv preprint arXiv:1805.00889*, 2018.

[13] V. Lostanlen, J. Salamon, A. Farnsworth, S. Kelling, and J. P. Bello, "Birdvox-full-night: A dataset and benchmark for avian flight call detection," in *2018 IEEE*

International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018: IEEE, pp. 266-270.

[14] J. F. Gemmeke *et al.*, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017: IEEE, pp. 776-780.

[15] A. Ståhlbröst, A. Sällström, and D. Hollosi, "Audio monitoring in Smart Cities: an information privacy perspective," 2014.

[16] R. Serizel, V. Bisot, S. Essid, and G. Richard, "Acoustic features for environmental sound analysis," in *Computational Analysis of Sound Scenes and Events*: Springer, 2018, pp. 71-101.

[17] D. M. Nogueira, C. A. Ferreira, E. F. Gomes, and A. M. Jorge, "Classifying heart sounds using images of motifs, MFCC and temporal features," *Journal of medical systems*, vol. 43, no. 6, p. 168, 2019. [Online]. Available: https://link.springer.com/content/pdf/10.1007/s10916-019-1286-5.pdf.

[18] J. O. Nordby, "Environmental sound classification on microcontrollers using Convolutional Neural Networks," Norwegian University of Life Sciences, Ås, 2019.

[19] M. Smales. "Sound Classification using Deep Learning,." https://medium.com/@mikesmales/sound-classification-using-deep-learning-8bc2aa1990b7 (accessed November, 2019).

[20] S. Doshi. "Music Feature Extraction in Python." https://towardsdatascience.com/extract-features-of-music-75a3f9bc265d (accessed November, 2019).

[21] J. Singh. "An introduction to audio processing and machine learning using Python." https://opensource.com/article/19/9/audio-processing-machine-learning-python (accessed December, 2019).

[22] M. Cohen, "Short-time Fourier transform," 1:39 ed. YouTube, 2017, p. https://www.youtube.com/watch?v=8nZrgJjl3wc.

[23] R. X. Gao and R. Yan, "Non-stationary signal processing for bearing health monitoring," *International journal of manufacturing research*, vol. 1, no. 1, pp. 18-40, 2006.

[24] D. Ellis. "Filterbank Analysis." https://labrosa.ee.columbia.edu/doc/HTKBook21/node54.html (accessed December, 2019).

[25] T. Giannakopoulos and A. Pikrakis, *Introduction to audio analysis: a MATLAB® approach*. Academic Press, 2014.

[26] N. Berry. "Discrete Cosine Transformations." http://datagenetics.com/blog/november32012/index.html (accessed November, 2019).

[27] H. Fayek. "Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between." https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html] (accessed November, 2019).

[28] J. Lyons. "Mel Frequency Cepstral Coefficient (MFCC) tutorial." http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/ (accessed November, 2019).

[29] ETSI. "Speech Processing, Transmission and Quality Aspects (STQ)." https://www.etsi.org/deliver/etsi_es/201100_201199/201108/01.01.03_60/es_201108v010103p.pdf (accessed Decemebr, 2019).

[30] E. F. Gomes, F. Batista, and A. M. Jorge, "Using smartphones to classify urban sounds," in *Proceedings of the Ninth International C* Conference on Computer Science & Software Engineering*, 2016, pp. 67-72.

[31] P. Borkar and L. G. Malik, "Acoustic signal based traffic density state estimation using adaptive neuro-fuzzy classifier," in *2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2013: IEEE, pp. 1-8.

[32] J. Castañón. "10 Machine Learning Methods that Every Data Scientist Should Know." https://towardsdatascience.com/10-machine-learning-methods-that-every-data-scientist-should-know-3cc96e0eeee9 (accessed December, 2019).

[33] N. GHalder. "Multi-class Classification using Decision Tree, Random Forest and Extra Trees Algorithm in Python: An End-To-End Data Science Recipe — 016." https://medium.com/@nilimeshhalder/multi-class-classification-using-decision-tree-random-forest-and-extra-trees-algorithm-in-python-ec2428d576ec (accessed December, 2019).

[34] C. Parsing, "Speech and language processing," 2009.

[35] S. Swaminathan. "Logistic Regression — Detailed Overview." https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc (accessed December, 2019).

[36] P. Chandrayan. "Logistic Regression For Dummies: A Detailed Explanation." https://towardsdatascience.com/logistic-regression-for-dummies-a-detailed-explanation-9597f76edf46 (accessed December, 2019).

[37] J. G. Colonna, J. Gama, and E. F. Nakamura, "How to correctly evaluate an automatic bioacoustics classification method," in *Conference of the Spanish Association for Artificial Intelligence*, 2016: Springer, pp. 37-47.

[38] C.-F. Cheng, A. Rashidi, M. A. Davenport, and D. V. Anderson, "Evaluation of Software and Hardware Settings for Audio-Based Analysis of Construction Operations," *International Journal of Civil Engineering*, vol. 17, no. 9, pp. 1469-1480, 2019.

[39] M.-H. Chiu, Y.-R. Yu, H. L. Liaw, and L. Chun-Hao, "The use of facial micro-expression state and Tree-Forest Model for predicting conceptual-conflict based conceptual change," *Chapter Title & Authors Page*, vol. 184, 2016.

[40] T. Yiu. "Understanding Random Forest." https://towardsdatascience.com/understanding-random-forest-58381e0602d2 (accessed December, 2019).

[41] A. Bronshtein. "A Quick Introduction to K-Nearest Neighbors Algorithm." https://blog.usejournal.com/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7 (accessed December, 2019).

[42] "Artificial Neural Networks - Multi-Layer Perceptrons." https://kyamagu.github.io/mexopencv/matlab/ANN_MLP.html (accessed December, 2019).

[43] J. Z. a. N. P. Fan Liu. "Neural Networks." https://sites.google.com/site/autosignlan/algorithms-used/neural-networks (accessed December, 2019).

[44] A. Mesaros, T. Heittola, and D. Ellis, "Datasets and evaluation," in *Computational Analysis of Sound Scenes and Events*: Springer, 2018, pp. 147-179.

[45] M. Sunasra. "Performance Metrics for Classification problems in Machine Learning." https://medium.com/thalus-ai/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b (accessed December, 2019).

[46] J. Brownlee. "How To Choose The Right Test Options When Evaluating Machine Learning Algorithms." https://machinelearningmastery.com/how-to-choose-the-right-test-options-when-evaluating-machine-learning-algorithms/ (accessed December, 2019).

[47] T. E. Oliphant, *A guide to NumPy*. Trelgol Publishing USA, 2006.

[48] B. McFee *et al.*, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, vol. 8.

[49] W. McKinney, "Pandas, Python Data Analysis Library," *see http://pandas. pydata. org*, 2015.

[50] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90-95, 2007, doi: 10.1109/MCSE.2007.55.

[51] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825-2830, 2011.

[52] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[53] G. Varoquaux. "Joblib: running Python functions as pipeline jobs." https://joblib.readthedocs.io/en/latest/ (accessed December, 2019).

[54] T. Giannakopoulos, "pyaudioanalysis: An open-source python library for audio signal analysis," *PloS one*, vol. 10, no. 12, 2015.

[55] P. S. Foundation. "datetime — Basic date and time types." https://docs.python.org/3.3/library/datetime.html# (accessed December, 2019).