CENTRO UNIVERSITÁRIO FEI

ANDRÉ LUIZ FLORENTINO

ENVIRONMENTAL SOUND RECOGNITION IN EMBEDDED SYSTEMS FOR AUTONOMOUS VEHICLES

São Bernardo do Campo

2024

ANDRÉ LUIZ FLORENTINO

ENVIRONMENTAL SOUND RECOGNITION IN EMBEDDED SYSTEMS FOR AUTONOMOUS VEHICLES

Master's thesis, presented to the University Center of FEI for the purpose of obtaining the degree of Master of Scince (MS.c) in Electrical Engineering. Main advisor Prof. Dr. Plinio Thomaz Aquino Junior.

São Bernardo do Campo

2024

BLOCKING AND RESTRICTION NOTICE

Publications on the content of this study assignment or bachelor/master/diploma academic qualification work ("**Thesis**") are only permitted with the prior written consent of EDAG Engineering GmbH, Kreuzberger Ring 40, 65205 Wiesbaden, Germany ("**EDAG**"). The results, opinions and conclusions of this Thesis are not necessarily those of EDAG but reflect the personal opinion of the author.

This Thesis contains trade and/or business secrets of EDAG and is therefore to be classified as

c o n f i d e n t i a l .

For a period of five (5) years from the date of printing, this Thesis may only be made accessible

- within the EDAG Group (i.e. the employees of EDAG as well as the employees with EDAG pursuant to sections 15 et seq. of the German Stock Corporation Act [*Aktiengesetz*]);
- to academic advisors or supervisors at the college or university where this Thesis is written ("**Institution**");
- to the Institution's reviewers and correctors involved in the assessment of this Thesis;

and/or

- to members of the Institution's examination board or committee responsible for evaluating this Thesis or supervising the Institution.

In any case, this Thesis is to be treated as strictly confidential for a period of five (5) years from the date of printing. During this period, this Thesis – in particular its contents and results – may not be used or made available to third parties, either directly or indirectly, not even in part.

Furthermore, this Thesis must not be transcribed, copied or otherwise reproduced in either digital or any other form.

Ficha catalográfica

Folha de aprovação

I dedicate this study to Almighty God, to my beloved wife Priscila, and in loving memory of my parents who guided me along the path of integrity, choosing what is right over what is easy.

ACKNOWLEDGMENTS

First and foremost, I would like to express my profound gratitude to God for the gift of life and the resilience provided to navigate challenges.

My heartfelt thanks go to my advisor, Prof. Dr. Plinio Thomaz Aquino Junior, for unwavering support throughout all stages of this study. I am also grateful to my former advisor, Prof. Dr. Guilherme Alberto Wachs Lopes, whose guidance introduced me to essential tools for the completion of this dissertation.

I extend my appreciation to the entire staff at FEI and my work colleagues at EDAG GmbH. Special thanks to Mr. Martin Vollmer and Mr. Alexandre Sberveglieri for their invaluable contributions to my professional development, and to Mr. Johannes Barckmann, Mr. Michael Jahn, and Mr. Maximilan Happel for their support in this project.

Finally, I am deeply indebted to my beloved wife, whose unconditional support served as a constant source of strength during these three years. In moments of discouragement, her presence was my sanctuary, providing me hope and inspiration to continue moving forward.

"Our greatest weakness lies in giving up. The most certain way to succeed is always to try just one more time."

Thomas A. Edison

"We don't have better algorithms, we just have more data."

Peter Norvig

ABSTRACT

The autonomous vehicle market is experiencing significant growth, with indications of transitioning from the "trough of disillusionment" to the "slope of enlightenment" on the Gartner hype cycle chart. This trajectory presents a promising opportunity to generate substantial revenue, potentially amounting to billions of dollars within the upcoming decade. The fundamental technologies encompassing extensive data analytics, computational capabilities, and sensor fusion techniques have already been established, and all stakeholders in this industry are persistently exploring novel approaches to enhance the overall perception of end users in terms of safety and trustworthiness. The main objective of this study is to investigate the feasibility of enhancing the perception abilities of autonomous vehicles by enabling them to recognize and categorize environmental sounds and merge its output in the sensor fusion network of the vehicle's architecture, resulting in specific responses either to the vehicle itself or the occupants. The methodology involved selecting relevant sound classes associated with the context of this study, evaluate the accuracy of several machine learning classifiers and neural network models, deploy the best-performing classifier to an embedded device, test the overall proposal using the realistic conditions of a regular passenger car and assess the potential enhancement of the vehicle's safety, decision-making, and overall autonomy.

Keywords: Environmental sound recognition, Autonomous vehicle, Embedded system, Feature extraction.

RESUMO

O mercado de veículos autônomos vem apresentando um crescimento significativo, com indicações de transição do "vale da desilusão" para a "inclinação da iluminação" no gráfico *Gartner hype cycle*. Este movimento representa uma perspectiva promissora para a geração de receitas significativas, possivelmente na ordem de bilhões de dólares nos próximos dez anos. As tecnologias essenciais que abrangem análise extensa de dados, capacidades computacionais avançadas e técnicas de fusão de sensores já foram estabelecidas e todas as partes interessadas nesta indústria estão constantemente explorando novas abordagens para aprimorar a percepção global dos usuários finais em termos de segurança e confiabilidade. O objetivo central desta pesquisa é avaliar a viabilidade de aprimorar as capacidades perceptivas de veículos autônomos, ao permitir o reconhecimento e a categorização de sons ambientais, bem como a integração dessas informações na rede de fusão de sensores da arquitetura do veículo, resultando em respostas específicas tanto para o controle do veículo em si como para seus ocupantes. A metodologia consistiu na seleção de classes de sons pertinentes, relacionadas ao contexto específico desta pesquisa, na avaliação da acurácia de diversos classificadores de aprendizado de máquina e modelos de redes neurais, na implementação do classificador de melhor desempenho em um sistema embarcado, na realização de testes da abordagem proposta em um ambiente de condução realista e na avaliação do potencial aprimoramento da segurança, tomada de decisões e autonomia geral do veículo.

Palavras-chave: Reconhecimento de som ambiental, Veículo autônomo, Sistema embarcado, Extração de características.

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ADC	Analog-to-Digital Conversion
ADSR	Attack-Decay-Sustain-Release
ANC	Active Noise Control
ANN	Artificial Neural Network
API	Application Programming Interface
CAN	Controller Area Network
CASA	Computational Auditory Scene Analysis
CNN	Convolutional Neural Network
CRF	Crest Factor
DCASE	Detection and Classification of Acoustic Scenes and Events
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DoA	Direction-of-Arrival
ECM	Electret Condenser
ECU	Electronic Control Unit
EE	Electrics & Electronics
ESR	Environmental Sound Recognition
ETi	Standard Temporal Integration
FFT	Fast Fourier Transform
FLA	Flatness
FPGA	Field-Programmable Gate Arrays
FR	Frequency Response
GAN	Generative Adversarial Network
GLM	Generalized Linear Model
GMM	Gaussian Mixture Model
GNB	Gaussian Naïve Bayes
GPU	Graphics Processing Unit
HCI	Human-Computer Interaction
k-NN	k-Nearest Neighbors
LiDAR	Light Detection and Ranging

LPCC	Linear Predictive Cepstral Coefficients
LR	Logistic Regression
LSTM	Long Short-Term Memory
MCR	Mean Crossing Rate
MEMS	Microelectromechanical Systems
MFCC	Mel-Frequency Cepstral Coefficients
MLP	Multilayer Perceptron
MSD	Mean Sequential Difference
PCA	Principal Component Analysis
PRISMA	Preferred Reporting Items for Systematic Reviews
RADAR	Radio Detection And Ranging
RAM	Random Access Memory
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RMS	Root Mean Square
SAE	Society of Automotive Engineers
SB	Spectral Bandwidth
SC	Spectral Centroid
SCT	Spectral Contrast
SFb	Standard Frame Based
SGD	Stochastic Gradient Descent
SI	Speech Intelligibility
SoC	System on Chip
SPL	Sound Pressure Level
SRP	Spectral Roll-off Point
std	standard deviation
STFT	Short Time Fourier Transform
STi	Standard Temporal Integration
SVM	Support Vector Machine
TPU	Tensor Processing Units
US8K	UrbanSound8K
YAMNet	Yet another Audio Mobilenet
ZCR	Zero-Crossing Rate

SYMBOLS

B	Byte
dB	Decibel
G	Giga, 10^9
Hz	Hertz
k	Kilo, 10^3
M	Mega, 10^6
m	Mili, 10^{-3}
s	Second

CONTENTS

1 INTRODUCTION

The autonomous vehicle market is experiencing rapid expansion, with numerous established brands and startups dedicating considerable efforts, resources, and investments to secure a position in this lucrative sector that may create 300$ billion to 400$ billion in revenues by the year 2035 (DEICHMANN, JOHANNES et al., 2023). These companies strive to offer sustainable, technologically advanced solutions that enhance user experience and provide innovative advancements. In recent years, there has been a proliferation of essential technologies that have greatly facilitated the advancement and reliable operation of autonomous vehicles, including the availability of large-scale data analytics (commonly known as big data), increased computational capabilities, and a wide range of sensor technologies such as ultrasonic sensors, Radio Detection And Ranging (RADAR), Light Detection and Ranging (LiDAR), cameras, and the integration of all these sensors through sensor fusion techniques (HUSSAIN et al., 2018).

According to the Society of Automotive Engineers (SAE), autonomous vehicles can be classified into six levels based on their ability to operate without human intervention. Level 0 represents vehicles that rely entirely on human drivers for all aspects of driving. Level 1 vehicles have certain functions automated, such as braking or steering, but still require full driver engagement. At Level 2, the automation system can control both steering and acceleration/deceleration, while the driver must remain fully attentive and ready to take control when needed. Level 3 vehicles can operate autonomously under certain conditions, allowing the driver to engage in other tasks, but prompt transition is required if the system requests intervention. Level 4 vehicles are highly autonomous and can operate without human input in specific geographic areas or predetermined scenarios, but not in all conditions. Finally, Level 5 vehicles are fully autonomous and can operate under all conditions, without the need for human intervention, enabling passengers to engage in non-driving related activities (SAE, 2021).

Much like a human driver relies on their sight, touch, and hearing to navigate and make informed decisions on the road, autonomous vehicles have primarily focused on vision and touch, neglecting the crucial sense of hearing or audition. This omission is akin to a human driver being deprived of their ability to hear, which would severely limit their situational awareness and pose significant challenges in detecting important auditory cues, such as emergency vehicle sirens, approaching vehicles or any other road users.

In the realm of human perception, the ability to effortlessly segregate and identify individual sound sources within a complex acoustic mixture is a common phenomenon, for

instance, an individual can effortlessly pick out a specific voice amidst a bustling background that includes the voices of other individuals and music. The field of sound analysis, commonly referred to in the literature as Computational Auditory Scene Analysis (CASA) (HERMES, 2023), delves into the investigation of methods to disentangle and recognize sound sources present in an auditory scene with the primary objective to equip computers with the ability to perceive and comprehend audio content, akin to the human auditory system.

Given the vast scope of CASA applications, this field of research is typically subdivided into three core areas according to Wang and Brown (2006):

a) **Context awareness**, also known as audio context recognition, pertains to identifying the location or ongoing activity within a specific environment based on audio information or events. It provides a context location of the sound, such as determining whether it's coming from a supermarket or a highway;

b) **Sound event detection and recognition** involves categorizing individual sound events present in an auditory scene. It addresses the question of "who and what" by recognizing the specific sound sources within the scene, for example a children crying;

c) **General audio classification** involves classifying and recognizing the contents of audio signals, commonly utilized for audio content retrieval, indexing, and audio-based searching.

Understanding the acoustic environment by **detecting and recognizing sound events** can be crucial for autonomous vehicles to perceive and react to their surroundings, providing a more comprehensive sensory system for safer and more efficient autonomous driving. By deploying audio classification algorithms on embedded systems within these vehicles, real-time Environmental Sound Recognition (ESR) can be achieved, allowing for advanced functionalities such as emergency vehicle detection, sound-based traffic monitoring, and early warning systems. Moreover, the effective utilization of embedded systems can alleviate the burden on external computing infrastructure and facilitate seamless integration of ESR into the existing vehicle architecture.

There is a substantial body of literature covering ESR algorithms that are implemented on embedded devices, operating at both low-level platforms such as microcontrollers (ABREHA, 2014) and (NORDBY, 2019), as well as high-level platforms like Field-Programmable Gate Arrays (FPGA) and Tensor Processing Units (TPU) (SILVA et al., 2019), (VANDENDRIESSCHE

et al., 2021) and (LHOEST et al., 2021). Despite the lack of research focusing on the application of these algorithms in autonomous vehicles or regular passenger vehicles, there is significant potential for research and development in this area. By addressing this research gap, the overall performance and safety of autonomous and regular passenger vehicles can be improved, while also making valuable contributions to the emerging field of acoustic perception in intelligent transportation systems. In addition to the identified research gap, it is worth noting that the field of acoustic perception in intelligent transportation systems has already attracted attention in the industry. Google Inc., for instance, has filed a patent related to the utilization of audio data for controlling autonomous vehicles (FERGUSON; ZHU, 2014).

1.1 CONTEXT

Urban mobility is a complex interaction of energy usage, movement patterns, and spatial allocation, and particularly in densely populated cities, space is a scarce and highly valuable resource. The extent and design of available space determine individuals' abilities to accomplish their professional and personal objectives, reconcile their desires and necessities, and enhance the quality of their living and working conditions. An alternative to achieving these goals relies on urban centers being planned with reduced vehicle presence, decreased parking infrastructure, and minimized environmental pollution. Furthermore, the integration of previously distinct and individual mobility systems is necessary to create a cohesive and efficient urban mobility network.

The concept of "SmartCity" has emerged as a response to the global phenomenon of urbanization and the projected growth of cities. This overarching term, embraced by politics, business, administration, and urban planning, encapsulates comprehensive development strategies aimed at enhancing efficiency, technological advancement, sustainability, and inclusivity within cities, primarily concentrate on technical, economic, and social innovations[1].

To fulfill the objectives of SmartCity endeavors, certain technical and urban planning prerequisites need to be addressed, namely a few of them:

a) Establishing designated areas for new mobility concepts where only bicycles are allowed, prohibiting individual or private vehicles. Pedestrians should always have priority over other vehicles, and designated pedestrian crossings may not be necessary;

[1] Citations were omitted due to the disclaimed blocking and restriction notice.

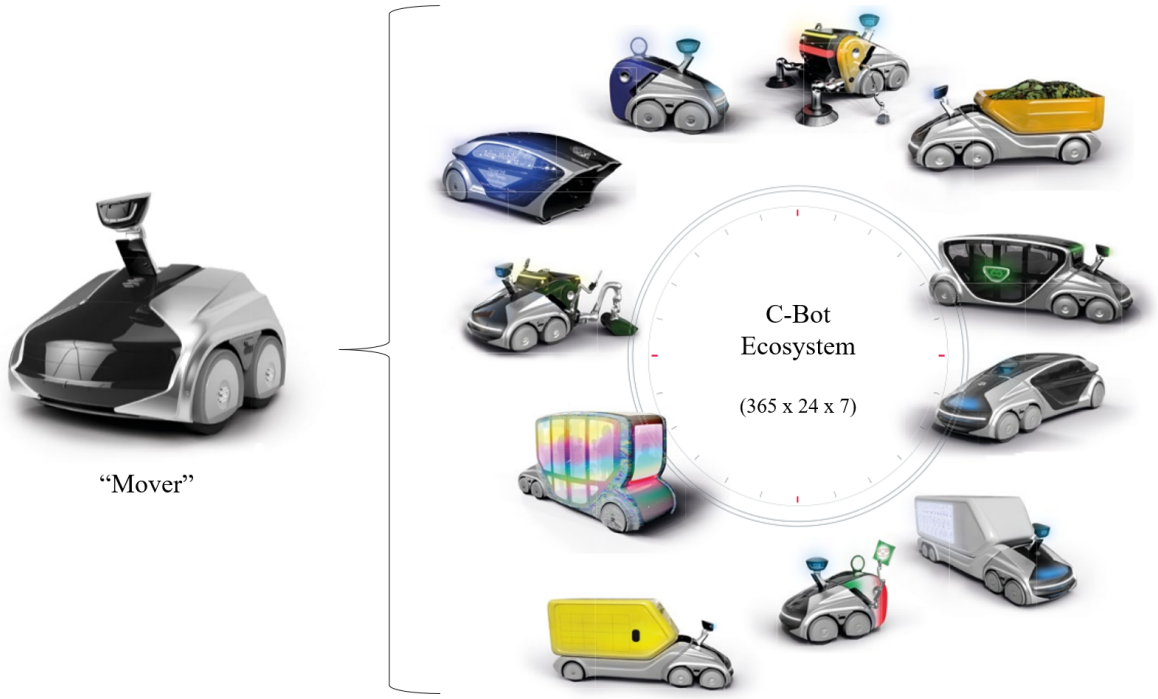
b) Integrating public transportation into the urban mobility system;
c) Adhering to open data and open standards, as well as promoting interoperability and barrier-free principles;
d) Incorporating Park+Ride zones on the outskirts of the city into the mobility system;
e) Expanding the network of cycling paths in newly available areas;
f) Ensuring that **autonomous transport systems** are constantly active, equipped with self-learning capabilities, intelligent decision-making algorithms, and future-oriented designs;
g) Implementing appropriate software solutions for traffic control and coordination, potentially eliminating the need for traditional traffic lights;
h) Collaborating with municipalities, industry stakeholders, and civil society to develop and expand these concepts.

One critical component of SmartCities will be the implementation of the autonomous transport systems, named hereinafter as C-Bots, swarm-intelligent, multi-functional, and fully autonomous robotic vehicles. These vehicles rely on emissions-free energy derived from fuel cells and possess modular and adaptable features, enabling constant usage throughout the day. Moreover, C-Bots can be equipped with rudimentary tools integrated into their "mover" component and can be fitted with additional modules to serve as passenger carriers, cargo transporters, or city cleaning devices, depending on specific requirements. Figure 1 illustrates the proposed ecosystem of the C-Bots under development by the company "X" since 2019.

In the background, and as part of the C-Bot project, the control system optimizes route planning and the utilization of the C-Bots in continuous 24-hour operation. Users can conveniently input their requests into the system, whether it be the user requesting transportation to the next stop via app, the store owner planning the delivery of new products for the following day, or the municipal vehicle organizing overnight waste removal. This can massively reduce the number of vehicles in city centers, make parking spots available for other uses, and alleviate traffic jams through consistent coordination. All of this makes a significant contribution to the concept of "shared space" traffic, where all **road users coexist on an equal footing**, and to a smart, sustainable, and livable city.

The project is currently in its third phase, as shown in Figure 2, where development partners will explore challenges and possible solutions for core elements of the planned ecosystem: **automated driving functions**, networking and data exchange, Human-Computer Interaction

Figure 1 – Illustration of the C-Bot ecosystem concept.



Source: Author, "adapted from" the original confidential source

(HCI), **acceptance and trust**, integrated order management, identification and realization of potential economic and technical optimization in operation, and many more secondary elements[2].

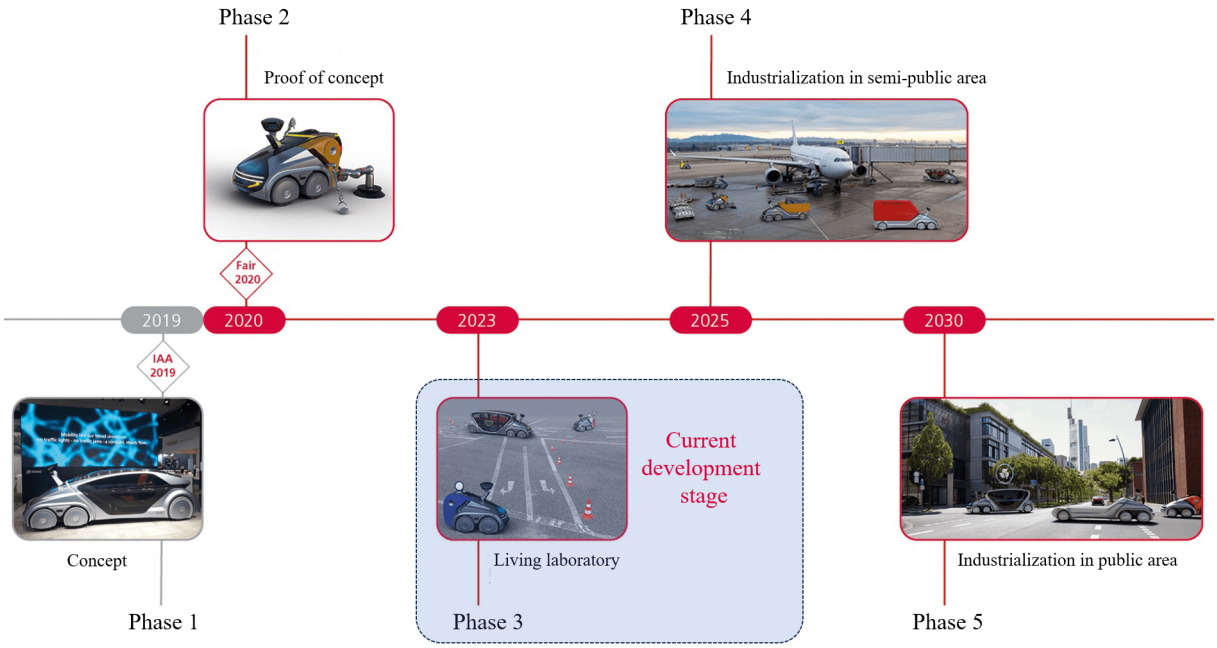
In this phase, the C-Bots are equipped with speech recognition capabilities enabling HCI with the end-users, however, as in all other autonomous vehicles identified until this publication, ESR is not yet implemented. Throughout several meetings with the project product owner and the technical staff, the feature of the ESR was ranked as "important" in the product backlog and it shall be integrated in the next milestone (Phase 4), primarily to confirm its implementation feasibility in the vehicle Electrics & Electronics (EE) and its relevance as a feature itself.

1.2 OBJECTIVE

The objective of this study is to develop and implement an ESR algorithm in embedded systems for autonomous vehicles ranging from level 2 to level 5. This can be achieved by investigating existing sound recognition algorithms, analyzing their performance, and selecting the most suitable ones for real-time embedded implementation.

[2]Citations were omitted due to the disclaimed blocking and restriction notice.

Figure 2 – Roadmap of the C-Bot development, since its original concept presented in 2019 until its fully implementation in public areas in 2030.



Source: Author, "adapted from" the original confidential source

1.2.1 General objective

As a general objective, this study intends to investigate whether it's feasible to enhance the perception capabilities of autonomous vehicles by enabling them to recognize and classify environmental sounds, merging the output into the sensor fusion network to produce specific responses from the vehicle EE architecture in due time. If the feasibility is confirmed, this study may contribute to improving their safety, decision-making, and overall autonomy.

Considering the context presented in section 1.1, and the fact that the C-Bot hardware construction is under development in a foreign country, an alternative approach to validate this study is proposed using as context, a regular passenger vehicle, with the ESR algorithms embedded in a high-end general-purpose platform, a separated microphone array to emulate the audio processing capabilities of the vehicle control unit and the same microphone utilized in the vehicle for speech recognition tasks.

1.2.2 Specific objectives

To achieve the general objective, this section outlines the specific objectives of this study:

a) Dataset benchmarking to confirm the classifiers' accuracy:
   - Establish the benchmark datasets consisting of diverse environmental sounds that will be used to evaluate the accuracy and performance of the implemented classifiers;
   - Implement and train different classifiers utilizing the benchmark datasets;
   - Assess the accuracy of the classifiers by comparing their results against known ground truth labels (validation sets) and the literature results.

b) Construction of a new dataset tailored for environmental sounds in the context of autonomous vehicles:
   - Define the relevant classes within the benchmark datasets that are relevant for this study, considering also the alternative approach proposed in the general objectives;

c) Training and evaluating the classifiers on a notebook and in an embedded device:
   - Implement and train the classifiers on a high-performance notebook computer using both benchmark and tailored datasets;
   - Evaluate the classifiers' performance in terms of accuracy, computational efficiency, and memory requirements;
   - Deploy the best-performing classifier to an embedded device, specifically a Raspberry Pi, and assess its performance in terms of accuracy and real-time processing capabilities.

d) Evaluating the best classifier in the context of a regular passenger vehicle:
   - Prepare the Raspberry Pi model to receive a microphone array and a microphonic sensor;
   - Evaluate the classifier's performance in the realistic context of a passenger vehicle, taking into consideration factors such as background noise, varying environmental conditions, and real-time response requirements;
   - Assess the classifier's ability to contribute to the enhancement of the vehicle's safety, decision-making, and overall autonomy. As ramification of this study, assess the contribution of the classifier in the context of a regular passenger vehicle.

1.3 ORGANIZATION

The organization of this study is structured into four chapters. In chapter 2 , the theoretical framework of the main concepts is outlined. Initially, the concepts and definitions related to audio processing are presented in section 2.1. Microphones and their automotive applications are explained in section 2.2, while section 2.3 highlights the machine learning and ensemble methods implemented in the classifiers, followed by section 2.4 that presents the same idea, but for neural networks. The chapter ends on section 2.5 where the hardware platforms in the automotive industry are presented, including the proposal utilized in the experiments of this study.

Chapter 3 addresses the state of the art in the recognition of environmental sound, where various concepts and works related to the proposed research are presented and discussed.

Chapter 4 focuses on the methods and materials employed in this study, encompassing section 4.1 where the hardware and software aspects are discussed in detail. This section provides an overview of the specific systems and tools utilized for the implementation and execution of the ESR algorithms. Section 4.2 pertains to the datasets used for training, evaluation and benchmarking purposes, outlining their content, the classifiers and features utilized for sound classification and its metrics. Normalization techniques applied to preprocess the audio data are described in section 4.3. It elaborates on the normalization methods used to ensure consistency and optimal performance during the subsequent stages of this study. In section 4.4, the augmentation techniques employed to enhance the dataset are explained, providing insights into the methods used to artificially increase the size and diversity of the dataset. Section 4.5 delves into the feature extraction process, detailing the techniques employed to extract relevant acoustic features from the audio data, taking into consideration the specific requirements for ESR. The training procedures followed for the classifiers are discussed in section 4.6. This section outlines the steps taken to train the machine learning, ensemble and neural network models, including the optimization of hyperparameters. Finally, section 4.7 addresses the evaluation procedures utilized to assess the performance of the implemented classifiers.

Chapter 5 presents the results of the conducted experiments encompassing section 5.1 that focuses on the evaluation metrics used, including accuracy, processing memory, allocation memory and response time. While section 5.2 outlines the results in the training and classification flow and the results of the evaluation flow into a regular passenger vehicle.

At last, chapter 6 brings the conclusion and future work within the context of ESR for autonomous vehicles and regular passenger vehicles.

1.4 HYPOTHESES

The purpose of this chapter is to enumerate all conceivable hypotheses that may arise from the present study, whether they pertain to practical applications or scientific advancements. The specific articulation of how these hypotheses will be incorporated into the dissertation remains to be determined.

Table 1 – List of alternative hypotheses H_a, potential application or contribution, and current status of its implementation on this study.

Id	**Description**	**Application or contribution**
H_1	ESR is performed in milliseconds, perhaps even seconds, before other types of object detection (cameras, radars, and sensors)	Other sensors can be set to higher levels of accuracy based on ESR outputs or be complemented by sensor fusion techniques.
Status H_1	**Alternative** H_a. Veeraraghavan and Ranga Charan (2020) confirmed by Simulink simulations the feasible of an acoustic controlled driving system level 3. Initial findings of Yin et al. (2023) also confirmed a suitable response time (Section 3)	
H_2	Driving a vehicle requires: sight or vision, touch or somatosensation, and hearing or audition. To this publication, autonomous vehicles have been neglecting the last one, it's plausible to state that ESR integrated in the vehicle EE architecture will improve its drivability experience.	Safer conditions to the vehicle occupants and overall better user experience.
Status H_2	**Alternative** H_a. To this point, this study concludes that the integration of ESR in autonomous vehicles seems logical and feasible, however, there is no empirical evidence of improvement in the drivability experience.	

Source: Author

Table 2 – List of null hypotheses H_0, potential application or contribution, and current status of its implementation on this study.

Id	Description	Application or contribution
H_3	A regular passenger car microphone has auditory sensitivity higher or equivalent than human ears.	If confirmed, there will be no need of special acoustic devices to identify specific sounds.
Status H_3	**Null** (H_0). However, it is imperative to note that automotive microphones do exhibit sufficient attributes capable of detecting a substantial range of acoustic events. This inference imparts significant technological insights regarding the use of automotive microphones in environmental sound recognition (Section 2.2.4).	
H_4	ESR will not improve decision-making algorithms of autonomous vehicles by merging its output into the sensor fusion network of the vehicle.	If null, faster and more accurate information can be transmitted to the driver.
Status H_4	**Null** H_0. One government funding project (I-SPOT, 2020) is exploring this topic, and in the private industry, one prototype already implemented audio sensor fusion (WAYMO, 2023), (Section 3)	
H_5	It's not possible to identify an object trajectory through ESR in a time frame (sound vector).	If null, different ESR outputs to the driver depending whether the identified sound is moving away or approaching the vehicle.
Status H_5	**Null** (H_0). Using external microphone array (MARCHEGIANI; NEWMAN, 2022) and (SUN et al., 2021), and with internal microphone array (SHABTAI; TZIRKEL, 2019), (Section 3)	

Source: Author

2 THEORETICAL FRAMEWORK

The theoretical framework presented in this chapter provides a comprehensive exploration of key components and concepts essential for understanding the field of environmental sound recognition in embedded systems for autonomous vehicles. Starting with the fundamentals of audio, an in-depth analysis of microphones is presented, highlighting their role as capturing devices. This is followed by an exploration of machine learning and the utilization of neural networks, laying the foundation for the subsequent discussions on classification metrics. Subsequently, the focus shifts towards environmental sound recognition and the available framework under licensing. Finally, the chapter concludes by delving into the electronic control unit, elucidating its significance as a crucial element in the overall system architecture.

2.1 AUDIO FUNDAMENTALS

This section provides a brief overview, definitions and concepts of the fundamental aspects of audio laid down by Mueller (2016) and Pelgrom (2018), setting the stage for a deeper exploration of environmental sound recognition.

Frequency, measured in hertz Hz, refers the number of vibrations per second in a sound wave and plays a crucial role in determining the pitch of the sound. Higher frequencies are perceived as higher pitch, and vice versa. **Pitch**, in turn, encompasses the perception of the frequency of a sound and is logarithmic in nature, meaning the difference between two pitches of an octave is perceived as the same, regardless of their frequency values.

Amplitude, measures the maximum displacement of a sound wave from its rest position. It is directly correlated with the loudness of a sound, as greater amplitude results in a louder sound. Additionally, an **octave** represents an interval between two frequencies where the higher frequency is double the lower frequency. This frequency relationship contributes to a logarithmic perception of pitch.

Magnitude, is often associated with amplitude, signifying the strength or intensity of the sound wave. While, **phase** refers to the position of a sound wave at a specific point in time, measured in degrees. It represents the progression of the wave cycle and is an essential aspect of understanding the characteristics of sound waves.

Sound power is the total acoustic energy produced by a sound source, measured in Watts. **Sound intensity** (W/m^2) represents the power per unit area of a sound wave, quantifying the sound energy passing through a given area, and is measured in Watts per square meter. The

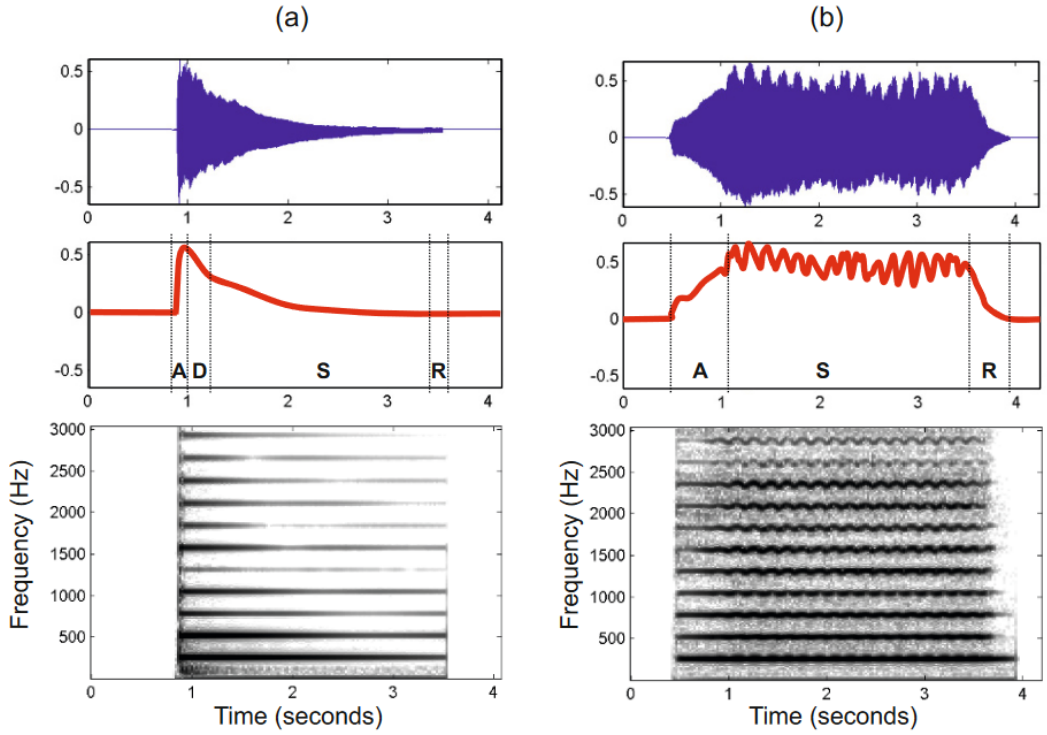
concept of **intensity level**, expressed in decibels dB, serves as a logarithmic scale to measure sound intensity, allowing for the representation of a broad range of sound intensities.

Loudness, on the other hand, is the subjective perception of the intensity of a sound and is influenced by various factors such as sound pressure, frequency, and individual hearing characteristics. This subjective aspect ties closely to the objective measurement of sound intensity and intensity level.

Additionally, **timbre** plays a role in the overall perception of sound quality. Timbre refers to the perceived color or quality of sound that distinguishes one source from another. It is influenced by factors such as harmonic content, attack, and decay of the sound. Thus, the objective measurements of sound power and intensity connect to the subjective aspects of loudness and the distinctive quality of timbre in our perception of sound

The **sound envelope** represents the temporal evolution of a sound over time and is typically described using the Attack-Decay-Sustain-Release (ADSR) model. It characterizes the initial attack, subsequent decay, sustained portion, and final release of a sound as depicts by the letters "A", "D", "S" and "R" in Figure 3.

Figure 3 – Waveform, amplitude envelope, and spectrogram representation for different instruments playing the same note C4 (261,6 Hz). (a) Piano. (b) Violin.



Source: Mueller (2016), page 28

The **threshold of audibility**, typically around 0 dB Sound Pressure Level (SPL)L, denotes the minimum sound level detectable by the human ear. In contrast, the threshold of pain, situated at approximately 120 dB SPL, represents the sound level at which sound becomes physically painful to the human ear.

Analog-to-Digital Conversion (ADC) involves two main processes:

a) **Sampling** refers to capturing the value of an analog signal at discrete time intervals while **sampling rate** determines the number of samples taken per second and is measured in Hz. In CD audio, the sampling rate is 44.100 Hz, meaning 44.100 samples are taken per second;

b) **Quantization** focuses on the amplitude of the signal, essentially dividing the range of possible amplitudes into discrete levels. The number of quantization levels determines the resolution of the digital signal.

A **digital signal** is represented by a sequence of discrete values, typically binary digits (bits). It differs from the analog electrical signals created by a microphone, which have a continuous range of values.

The **resolution of a digital signal** is often specified in terms of the number of bits used for quantization, also known as the bit depth. Common bit depths include 16, 24, and 32 bits, with 16 bits being the standard resolution for audio CDs. For 1 minute (60 seconds) of audio at a sampling rate of 44.100 Hz and a bit depth of 16 bits, the required memory is 5,17 MB, calculated using the formula:

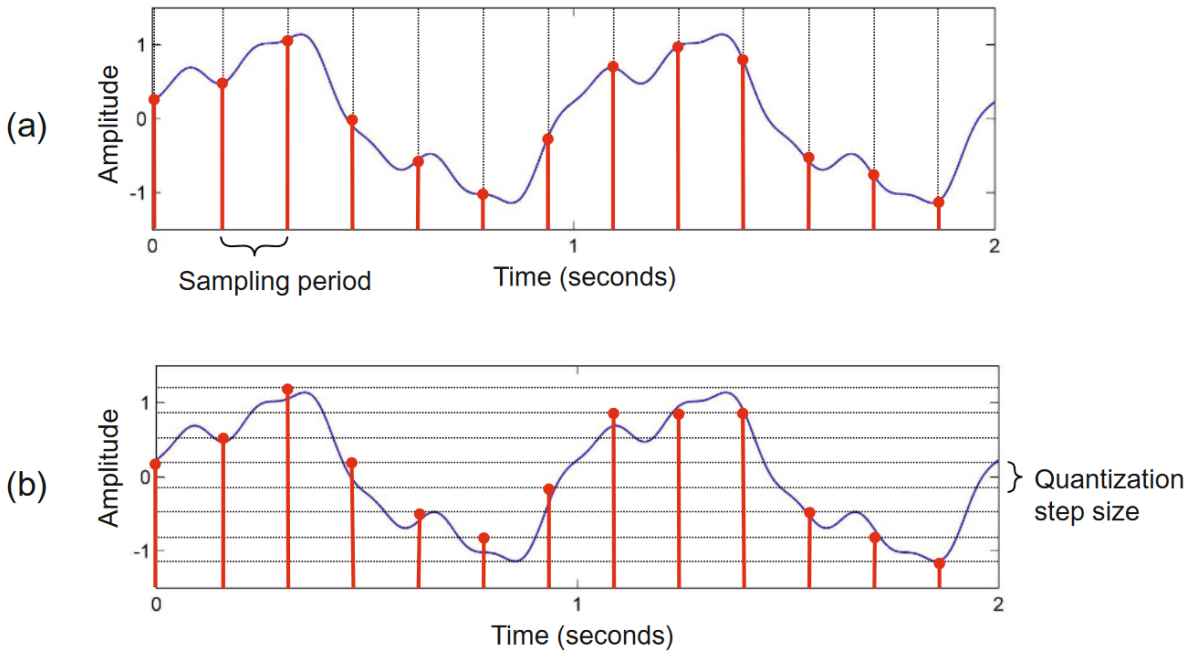
$$\text{Memory (MB)} = \frac{\text{Sampling rate} \times \text{Bit depth} \times \text{Duration}}{8 \times 1024} \tag{1}$$

The higher the bit depth (or resolution) of a digital audio signal, the greater the dynamic range it can capture and reproduce. A higher bit depth allows for a more accurate representation of subtle changes in amplitude.

The digitization process in the ADC involves the quantization of the signal in time at a specific sampling rate and the quantization of its amplitude at a particular bit-depth. With the typical audio CD values above, these parameters efficiently capture the majority of perceptible information in the acoustic sound. The resultant digital representation of sound takes the form of a one-dimensional sequence of numbers, also known as a waveform, as illustrated in the Figure 4.

Artifact refers to an unintended and undesirable distortion or alteration of the audio signal that occurs during recording, transmission, processing, or playback. Artifacts can manifest as unwanted sounds, anomalies, or imperfections that degrade the quality of the audio signal.

Figure 4 – Two steps of a digitization process to transform an analog signal (solid curve) into a digital signal (stem plot). (a) Sampling. (b) Quantization.



Source: Mueller (2016), page 61

According to the Nyquist-Shannon theorem, the **Nyquist frequency** is half of the sampling rate. It represents the highest frequency that can be accurately represented in a digital signal without introducing distortion.

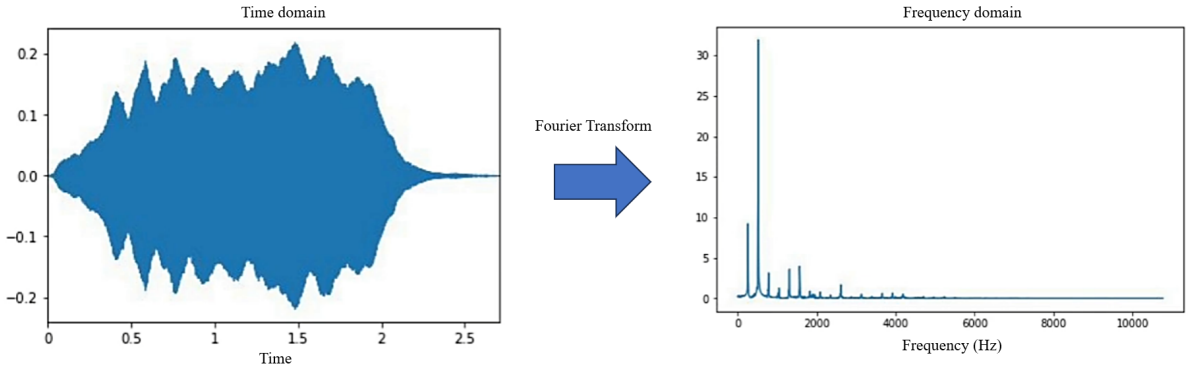
A **spectrogram** is a visual representation of the frequency content of a signal over time. It displays how the intensity of different frequencies in the signal changes over time, typically with frequency on the vertical axis, time on the horizontal axis, and color or brightness representing intensity

2.1.1 Digital audio analysis

The conventional method for analyzing signals in the spectral domain relies on classical Fourier analysis, which is applied to the entire signal shifting the information from the time domain to the frequency domain as shown in Figure 5. However, the definition of the Fourier Transform faces challenges in audio signal analysis because real-world audio signals are time-varying. These signals are non-stationary and their significance is inherently linked to this temporal variability. Therefore, there is a crucial need to develop techniques for analyzing sound

that can capture the important time-varying aspects of audio signals which are essential for tasks such as feature extraction and other audio analysis tasks.

Figure 5 – Example of Fourier Transform applied to an audio signal illustrating the switch from the time domain to the frequency domain.



Source: Author

In order to overcome these challenges, the audio signal is initially divided into short segments called frames where each frame is specifically chosen to be brief enough to be considered approximately stationary. This segmentation process is commonly referred to as **Framing**. The duration of each frame typically ranges from 10 to 50 milliseconds, assuming that within such a short timeframe, the audio signal undergoes minimal changes. Audio processing tasks, such as Fourier Transform and feature extraction, are performed on a frame-by-frame basis. To ensure a smoother transition and minimize abrupt changes and discontinuities which causes spectral leakage in the signal, frames are often multiplied by smoothing functions, such as the Hanning window function, prior to applying Fourier Transform operations which renders this process the name **Windowing** (ABREHA, 2014).

The technique of analyzing audio signals frame by frame is referred to as short-time signal analysis. In the existing literature (DEBNATH; SHAH, 2014), there are various methods for short-time analysis, namely the Short Time Fourier Transform (STFT), Discrete Wavelet Transform (DWT), and Wigner-Ville Distribution (WVD). Among these, STFT stands out as the most commonly employed short-time analysis technique due to its computational simplicity.

2.1.1.1 Short-Time Fourier Transform

Joseph Fourier, born in 1768, made significant contributions to the field of mathematical physics, and he is particularly known for his work on heat transfer and the mathematical techniques

he developed, including what is now known as the Fourier series and the Fourier Transform. His work on the decomposition of functions into sums of sinusoidal functions laid the foundation for the understanding of periodic functions and played a crucial role in various fields, including signal processing, communication theory, and quantum mechanics (DEBNATH; SHAH, 2014).

The formula for the continuous complex Fourier Transform of a signal $f(t)$ is given by:

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(t) \cdot e^{-i\omega t} dt \tag{2}$$

Where $\hat{f}(\omega)$ is the complex Fourier Transform of the signal at frequency ω, the input signal is given by $f(t)$, the the angular frequency by ω, and the imaginary unit is i.

This formula involves integrating the product of the input signal and a complex exponential term over all time values t. The result is a complex-valued function of frequency ω that describes the frequency content of the input signal.

Given an analog signal $f : \mathbb{R} \to \mathbb{R}$ and a positive real number $T > 0$, the function $x : \mathbb{Z} \to \mathbb{R}$ is defined by $x(n) := f(n.T)$. Considering x is defined only on a discrete set of time points, the value $x(n)$ is named **sample** taken at time $t = n.T$ of the original analog signal f at the sampling period T, which is the inverse of the sampling rate.

In the context of discrete-time signals, the Discrete Fourier Transform (DFT) is often used assuming, firstly, that most of the relevant information of f is limited to a certain duration in time, resulting in a finite number of samples $x(0), x(1), \ldots, x(N-1)$ for some suitable number $N \in \mathbb{N}$, and secondly, one calculates the Fourier Transform for a limited set of frequencies. Analogous to discretizing the time axis, the frequency axis is commonly sampled with frequencies $\omega = k/M$ for some appropriate $M \in \mathbb{N}$ and $k \in [0 : M - 1]$. In practical applications, it is common to align the number N of samples with the parameter M determining the frequency resolution by setting $N = M$. It is essential to note that N and M refers to distinct aspects. Nevertheless, this alignment proves advantageous, ensuring not only the invertibility of the resulting transform but also yielding a computationally efficient algorithm (MUELLER, 2021).

The formula for the DFT with integers $k \in [0 : M - 1] = [0 : N - 1]$ is given by:

$$X(k) = \hat{x}(k/N) = \sum_{n=0}^{N-1} x(n) \cdot e^{-i2\pi n \frac{k}{N}} \tag{3}$$

Where $\hat{x}(k/N)$ is the DFT of the signal at frequency index k, the input signal is $x(n)$, and N is the length of the signal.

The DFT represents the frequency content of the signal at equally spaced frequency bins. In practice, the Fast Fourier Transform (FFT) algorithm is often used to efficiently compute the DFT by taking advantage of redundancies among sinusoids of various frequencies, allowing for the simultaneous computation of all Fourier coefficients through a recursive process. This recursion is especially effective when N is a power of two, consequently, the FFT significantly decreases the total number of operations, reducing it from $O(N^2)$ to $O(N \log_2 N)$.

Instead of considering the entire domain of the signal which hides *when* the frequencies occur, Dennis Gabor introduced in 1946, the Short-Time Fourier Transform (STFT) by focusing on specific sections of the signal or **frames**. This is achieved by multiplying the original signal with a **window function**, which defines the section being considered. By shifting the window function across time and performing a Fourier Transform on each windowed signal, frequency information at different time instances can be obtained. In other words, the STFT breaks the signal into frames, which overlap with each other to reduce artifacts at the boundary and to create more instances for analysis.

Let a discrete signal x be a function that maps integers to real numbers $\mathbb{Z} \to \mathbb{R}$. This signal is obtained by sampling at a regular rate F_s, given in Hz. Now let $w : [0 : N-1] \to \mathbb{R}$ be a sampled window function of length $N \in \mathbb{N}$. For example, considering a rectangular window, $w(n) = 1$ for $n \in [0 : N-1]$, outside this window, at time parameters not within $[0 : N-1]$, $w(n)$ will be assumed to be 0. The length parameter N determines the duration of the sections (frames) considered in terms of N/F_s seconds. Additionally, the parameter H is introduced, a positive integer referred to as the **hop length** which determines the number of samples by which the window is shifted across the signal (MUELLER, 2021).

The formula for the discrete STFT $\mathcal{X}$ of the signal x with $m \in \mathbb{Z}$ and $k \in [0 : K]$ where $K = N/2$ (assuming that N is even) is the frequency index associated with the Nyquist frequency is given by:

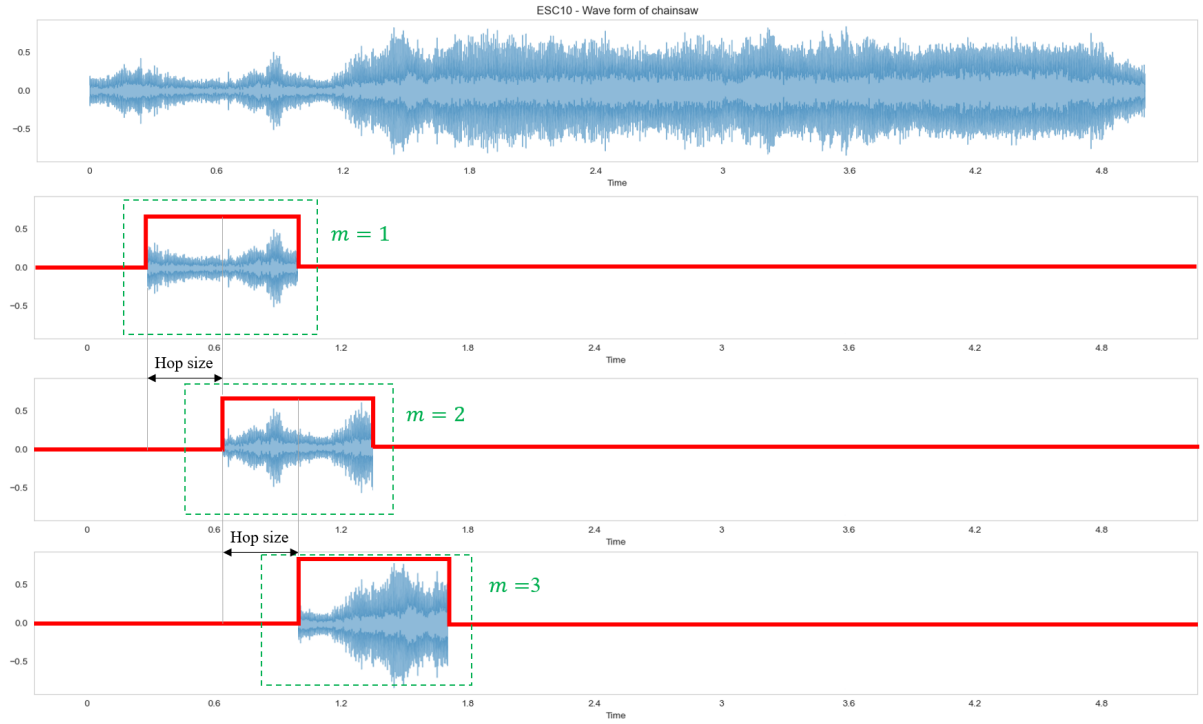
$$\mathcal{X}(m,k) := \sum_{n=0}^{N-1} x(n + mH)w(n) \cdot e^{-i2\pi n \frac{k}{N}} \tag{4}$$

Where $\mathcal{X}(m,k)$ represents the k^{th} Fourier coefficient for the m^{th} time frame. For each time frame m, one gets a spectral vector of size $K+1$ given by its associated Fourier coefficients. The computational costs of each spectral vector amounts to a DFT of size N, and as previously mentioned, it's more efficiently calculated using FFT. Figure 6 depicts this windowing process using a rectangular window function and hop length of 50%.

Assuming the total length (in samples) of an audio signal as L, the number of frames m^{th} calculated using STFT is given by:

$$\text{Number of frames} = \left\lfloor \frac{L - N}{H} \right\rfloor + 1 \tag{5}$$

Figure 6 – Illustration of rectangular window function applied to an audio signal simulating the overlap of the frames.



Source: Author

2.1.1.2 Windows for the Short Time Fourier Transform

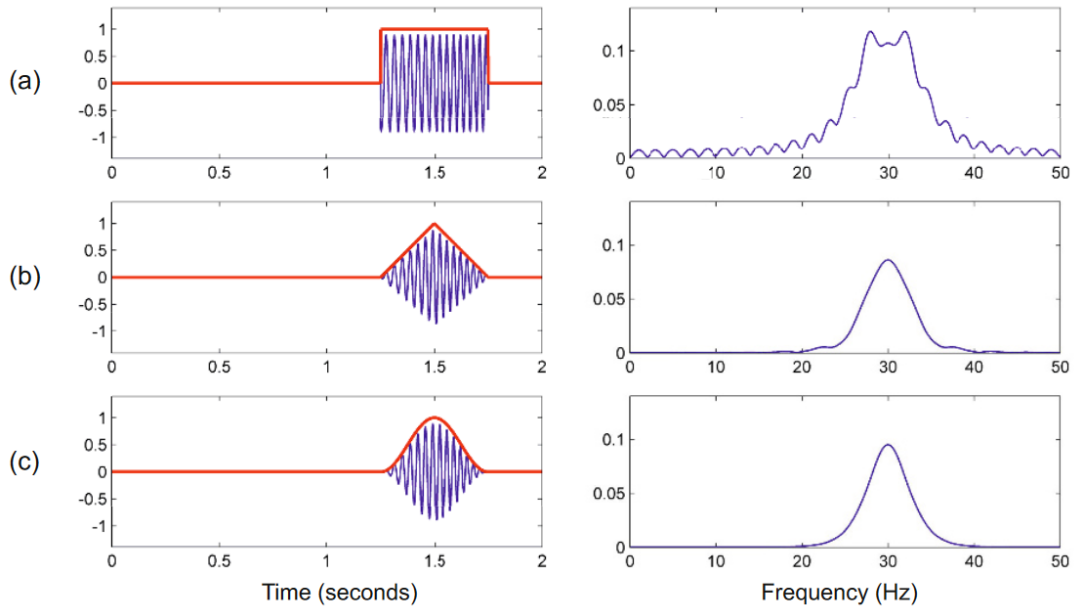
As already described, in audio analysis, techniques such as STFT are used to analyze sound by dividing the audio signal into smaller time segments called frames. The length of each frame is typically chosen to balance the trade-off between time and frequency resolution and overlapping frames are often used to improve the time resolution of the analysis. In the literature, a few types of framing have been utilized according the end application, for example, Preece et al. (2009) describe three types of framing to identify activities using body-mounted sensors: sliding windows, event-defined windows and activity-defined windows.

Various window sizes have been utilized in earlier research, with some studies incorporating variable overlap between neighboring windows. Notably, the sliding window technique

is straightforward and advantageous for real-time applications as it eliminates the need for pre-processing the audio signal. Its simplicity of implementation has made it the most commonly used approach for audio analysis and classification studies.

Considering the sliding window technique as the mainstream, the frames have to be synchronized or filtered using window functions. Several different windows are available named after their original developers in the 1950s, namely: Hamming window, Blackman window, Bartlett window (triangular), Hanning window (raised cosine window) and rectangular window (SMITH, 2013), the last three illustrated in Figure 7. In STFT, a window function plays a crucial role in mitigating spectral leakage and also managing trade-offs between time and frequency resolution by tapering the signal smoothly to zero at the edges thus minimizing abrupt transitions. The choice of window function influences the balance between main lobe width and side lobe levels, impacting the resolution of both time and frequency components in the resulting spectrogram. Named after Julius Von Hann, the **Hanning window** is one of the most used window in signal processing according to Mueller (2021).

Figure 7 – Windowed chirp signal and its magnitude Fourier Transform using different window functions. (a) Rectangular window. (b) Triangular window. (c) Hanning window.



Source: Mueller (2021), page 98

Usually, the frame size and the Hanning window size have equal number of samples, although in some situations, one may choose different values for each one of them. Considering $w(n)$ is the value of the window at sample index n, and N is the total number of samples in

the window with $0 \leq n \leq N-1$, the Hanning window is mathematically defined according to Smith (2013) as:

$$w(n) = 0.5 - 0.5\cos\left(\frac{2\pi n}{N-1}\right) \tag{6}$$

2.1.1.3 Window function parameters

When performing a STFT using methods from specialized Python libraries such as openSMILE 3.0 (EYBEN; WÖLLMER; SCHULLER, 2010) or Librosa (MCFEE et al., 2015), there are essentially three parameters that influence the result: window type, window size and overlapping size or hop length.

The choice of **window type** affects directly the resulting of the signal spectrogram. For instance, rectangular windows have a wide main lobe, providing high spectral resolution but suffer from significant spectral leakage due to high side lobes which causes a suddenly interruption of the signal at its edges, possibly leading to incorrect estimations of frequency components. On the other hand, window types like Hamming and Hanning have narrow main lobes and lower side lobes which reduces the signal's values towards zero at the edges of the window, preventing abrupt changes, resulting in a smoother spectrogram with reduced spectral leakage. The selection of an appropriate window type depends on the specific requirements of the application, such as the desired frequency resolution and the ability to distinguish between closely spaced spectral components (ZOELZER, 2008).

The **window size** parameter plays a vital role in determining the trade-off between time and frequency resolutions in the STFT analysis. Assumption within one frame, the signal is stationary, the window size directly influences the temporal and frequency granularity of the resulting spectrogram. A smaller window size provides better temporal resolution, enabling the identification of rapid changes in the audio signal, however, it comes at the cost of reduced frequency resolution, making it challenging to accurately distinguish between closely spaced frequency components. Conversely, a larger window size improves frequency resolution but compromises temporal resolution. The choice of window size is typically application-dependent, where tasks requiring fine spectral details may necessitate smaller window sizes, while tasks focusing on temporal dynamics may benefit from larger window sizes. Notably, the terms "window size" and "frame size" refer to slightly different aspects but are closely related: the window size specifically refers to the length or duration (in samples) of the window function that

is applied to segments of the audio signal during the STFT analysis while the frame size refers to the duration of the segments (in samples) into which the audio signal is divided before applying the window function (SMITH, 2013). The output vector from a N-sample DFT of the audio signal sampled at F_s results in $N/2 + 1$ frequency bins, with each bin capturing the energy from a small range of frequencies in the original signal (ABREHA, 2014), hence, the output vector of the STFT applied to N samples results in array of dimension: [frequency bins , number of frames] where the number of frames is given by equation 5. Most Python libraries implement their methods considering the window size and frame size to be identical, with the only option being to specify the frame size.

The process of overlapping frames guarantees that audio features present at a discontinuity are included in their entirety in the subsequent overlapped frame. The **overlapping size**, typically measured as a percentage, indicates the extent to which the previous frame is replicated in the following frame. Similar to selecting the window size, choosing the appropriate overlap size heavily depends on the specific objectives of the analysis or application. Generally, increasing the amount of overlap results in more analysis points and thus yields smoother results over time, potentially enhancing recognition accuracy in classification algorithms, however, this also leads to a proportional increase in computational cost. In Librosa (MCFEE et al., 2023), this parameter is named **hop length** and inputs an integer representing the number of samples overlapped.

2.1.2 Audio features

Essentially, there are two main types of audio features commonly used in the field of audio analysis: **physical features** and **perceptual features**. Physical features involve mathematical measurements derived directly from the sound wave, including the energy function, spectrum, cepstral coefficients, and fundamental frequency. On the other hand, perceptual features are subjective terms that relate to how sounds are perceived by humans, encompassing aspects such as loudness, brightness, pitch, timbre, and rhythm (ZHANG; C. JAY KUO, 2010) and (ALÍAS; SOCORÓ; SEVILLANO, 2016). Further on, physical features are further classified into **temporal features** and **spectral features**[1], which are based on the signal domain categorization. The selection of the following specific physical features for use in ESR applications was based on a comprehensive literature review presented in the related work chapter, but before that, a few definitions regarding feature categorization and requirements for feature selection are given.

[1]For simplicity, frequency, image and cepstral domains were all compiled in this classification

2.1.2.1 Requirements for selecting the audio features

Within the existing literature, numerous methods for extracting audio features are documented. The effectiveness of ESR heavily relies on the choice of audio feature extraction techniques utilized, and consequently, the careful selection of audio features becomes crucial in achieving optimal performance and accurate recognition. Even when constraining the application to ESR, the the choice of audio features depends on the application's type and purpose, particularly in resource-constrained devices like those used in the automotive industry, specifically for autonomous vehicles. The requirements for such implementation include multi-kernel support for end-to-end hybrid algorithms, real-time low-latency operation to quickly respond to each target event, and an optimized energy efficiency to reduce the overall vehicle power budget (YIN et al., 2023), nevertheless, these requirements can affect the recognition accuracy. Therefore, the selection of audio features should aim to develop a technique with low computational complexity, fast response time in the context of automotive applications, low energy consumption, and memory requirement, while still providing acceptable recognition accuracy. Given these requirements, the selection of audio features should include a small feature size to minimize computational cost, low computational complexity, high inter-class variability for improved discrimination between audio patterns, high intra-class similarity to decrease discrimination among similar classes, and low sensitivity to minor changes in the signal for robustness against noise and other sources of interference.

2.1.2.2 Temporal features

Temporal features refer to characteristics or attributes of sound signals that evolve over time, making the temporal domain the native representation domain for audio signals. Extracted directly from the raw audio signal, temporal features do not require any preceding transformation, keeping the computational complexity low in comparison to spectral features. These features provide valuable information about the temporal dynamics of the audio signal. The temporal features included in this study are: **Zero-Crossing Rate (ZCR)** and **Root Mean Square (RMS)**.

ZCR is a low level feature that captures the rate of transitions between positive and negative values within a signal, in other words, the smoothness of a signal by counting the number of times the signal crosses the zero axis within a segment of that signal. It finds widespread application in fields such as speech recognition, music information retrieval, and percussive sound classification. Additionally, ZCR proves valuable in tasks like basic pitch detection for

monophonic tonal signals and voice activity detection to ascertain the presence of human speech within audio segments (PARK, 2008), it's sensitive to noise though.

RMS is another low level feature that calculates the average energy of a frame and is an indicator of loudness which corresponds more closely to our hearing system's sensitivity to the change in intensity of audio signals. It is less sensitive to outliers compared to amplitude envelops and finds applications in audio segmentation and music genre classification.

2.1.2.3 Spectral features

Spectral features refer to the characteristics or attributes extracted from the frequency domain representation of an audio signal aiming to capture specific spectral patterns or properties present in the signal. Peeters (2004) presents a large set of audio features and most of them dwell in the frequency domain, so it's imperative to select the spectral features carefully. Each frequency or frequency bin provides information about its magnitude and phase, given that changes in phase have little impact on the perceived sound, there is no need to focus on features that analyze phase. Instead, the focus shifted to features that capture basic properties of the sound's frequency distribution. Additionally the spectral features utilized by Lhoest et al. (2021), Silva et al. (2019), and Bountourakis, Vrysis, and Papanikolaou (2015), provided a reliable direction. The spectral features included in this study are: **Spectral Centroid (SC)**, **Spectral Bandwidth (SB)**, **Spectral Roll-off Point (SRP)**, **Spectral Contrast (SCT)**, **Tonnetz**, **Chroma feature**, **Mel spectrogram**, **MFCC**, and two feature manipulations namely: **Delta MFCC** and **Delta-Delta MFCC**.

Spectral Centroid (SC) can be interpreted as the "center of mass" or "center of gravity" of the spectrum of a audio signal. It is a measure of where the "center" of the spectral content of an audio signal lies in terms of frequency and it is often used to describe the perceived "brightness" or "timbre" of a sound, and for that reason, sometimes is also addressed as a perceptual feature. It's a useful feature for distinguishing between different types of sounds based on their spectral characteristics. For example, sounds with higher spectral centroids tend to have a brighter or more treble-heavy quality, while those with lower centroids are generally more bass-heavy or darker in timbre (PARK, 2008).

Spectral Bandwidth (SB) is another important feature that characterizes the width or spread of the frequency content in an audio signal's spectrum. It provides information about how concentrated or dispersed the spectral energy is across different frequency components. It has

also a perceptual definition which is commonly associated with the perceived "sharpness" or "clarity" of a sound. A higher spectral bandwidth indicates that the spectral energy is spread out over a wider range of frequencies, which can result in a sound that is perceived as being more complex or noisy. In contrast, a lower spectral bandwidth suggests that the energy is concentrated around a narrow band of frequencies, leading to a cleaner or purer sound (PARK, 2008).

The Librosa (MCFEE et al., 2023) method to extract SB is based on Klapuri and Davy (2006), but instead of calculating the second central moment of the spectrum, also known as spectral spread, using $p = 2$, it enables the computation of the $p - th$ root of the sum, where p is left an input parameter (p determines the "order" of the bandwidth calculation).

Spectral Rolloff Point (SRP) characterizes the frequency below which a specified percentage (often a threshold like 85% or 95%) of the total spectral energy lies and it provides information about the lower boundary of the frequency range where most of the energy in an audio signal is concentrated. A higher spectral rolloff value indicates that the majority of the spectral energy is contained in lower frequencies, while a lower value suggests that energy extends into higher frequencies.

This feature is valuable for discerning voiced speech from unvoiced speech: unvoiced speech, including most of the environmental sounds, typically exhibits a significant concentration of energy in the higher frequency range of the spectrum, whereas voiced speech and music tend to have the majority of their energy concentrated in lower frequency bands. (GIANNAKOPOULOS; PIKRAKIS, 2014). This study adopted the threshold of 85% hence.

Spectral Contrast (SCT) represents a way to describe the tonal characteristics or timbre of an audio signal. It is a feature that quantifies the difference in spectral energy between peaks and valleys in the frequency spectrum of an audio signal. Higher spectral contrast indicates greater variation in energy levels across frequency bands, which can lead to a perception of greater clarity and distinction between different parts of the sound spectrum. This can be relevant in tasks such as speech recognition, music analysis, and sound classification, where distinguishing between different frequency components is important for understanding the content of the audio signal (JIANG et al., 2002).

Given a frequency domain divided into six Octave-scale sub-bands (0-200, 200-400, 400-800, 800-1.600, 1.600-3.200, and 3.200-8.000 Hz), SCT is computed by calculating the difference in energy between the maximum and minimum spectral values within each sub-band. The result is a multi-dimensional vector representing the contrast in energy between different frequency regions represented by the sub-bands.

Chroma Feature represents a way to describe the harmonic content or pitch class of an audio signal. It is a feature that characterizes the distribution of musical pitches (notes) in an audio signal, ignoring the octave or absolute frequency information. Chroma Features are typically computed using a chromagram , which is a representation of the 12 different pitch classes in Western music corresponding to the 12 semitones in an octave (e.g., C, C#, D, D#, E, F, F#, G, G#, A, A#, B). One way to calculate Chroma Feature is, first the audio signal is transformed into the frequency domain using techniques such the DFT, then, for each short time frame of audio, the energy or magnitude of each of the 12 pitch classes is measured, resulting in a 12-dimensional vector that represents the presence or strength of each pitch class in that time frame (BARTSCH; WAKEFIELD, 2005).

Although similar in concept, Chroma Feature and SCT capture different aspects of the audio signal. Chroma Feature summarize the distribution of energy in different pitch classes over time, SCT, on the other hand, focuses on capturing the difference in energy between peaks and valleys in the frequency spectrum. Chroma Features are computed by grouping frequency bins in the DFT spectrogram that correspond to the same pitch class and summing their magnitudes while SCT is computed by dividing the spectrum into sub-bands, computing the mean energy within each sub-band, and then calculating the difference between the maximum and minimum energy across adjacent sub-bands.

Tonnetz represents a way to describe the tonal relationships or harmonic structure of an audio signal that characterizes the tonal distance between musical pitches (notes) in an audio signal using the concept of tonnetz, which is a two-dimensional representation of pitch space. To calculate Tonnetz features, first the Chroma Features are extracted using the STFT, then the Chroma Features are converted to Tonnetz representation by using Euler's Tonnetz transformation. The result is a two-dimensional matrix or feature vector representing the tonal relationships between different pitch classes (HARTE; SANDLER; GASSER, 2006).

Librosa returns a vector [6, N] where each row of Tonnetz values corresponds to the tonal centroid features for each frame N. The vector dimensions represents 0: Fifth x-axis, 1: Fifth y-axis, 2: Minor x-axis, 3: Minor y-axis, 4: Major x-axis and 5: Major y-axis.

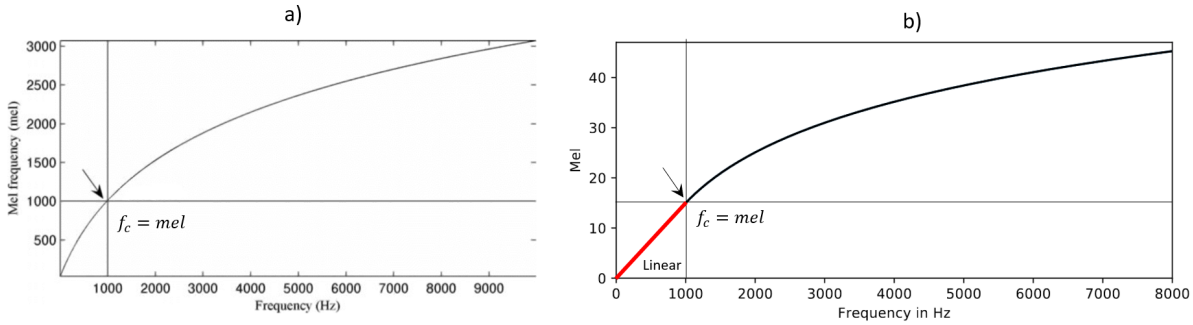
The **Mel** Scale represents a significant advancement in our understanding of how humans perceive frequencies. Research has revealed that our perception of frequencies is not linear; rather, we exhibit greater sensitivity to differences in lower frequencies compared to higher ones. To illustrate, distinguishing between 500 and 1.000 Hz is relatively straightforward, while

discerning a difference between 10.000 and 10.500 Hz proves considerably more challenging, despite the equal numerical interval between the two pairs.

In 1937, the concept of the Mel scale was introduced by Stevens, Volkmann, and Newmann as a unit of pitch, designed to ensure that equal pitch intervals sounded equally spaced to the human ear. The name "Mel" originates from the musical term "melody," highlighting its connection to the frequency aspect of musical compositions and the measurement of pitch-related perceptual distances within melodies. To achieve this, a mathematical transformation is applied to frequencies, converting them into the **Mel scale**, thereby aligning auditory perception more closely with human sensitivity to frequency differences, in other words, the Mel scale is constructed such that sounds of equal distance from each other on the Mel scale, also “sound” to humans as they are equal in distance from one another having less resolution at high frequencies and a finer resolution at low frequencies (MOORE, 2013).

Park (2008) presents the original Mel scale curve in Figure 8 a) when f_c is equal to m at $f_c = 1$ kHz — the reference point between Mel frequency and frequency in Hertz, where 1 kHz is equal to 1.000 mels (the reference dB is 40 dB above the threshold of hearing).

Figure 8 – Original Mel scale curve illustrating the intersection point where f_c = 1kHz = 1.000 mels a), and the common implementation utilized in Python and Matlab libraries with a linear response below f_c = 1 kHz b).



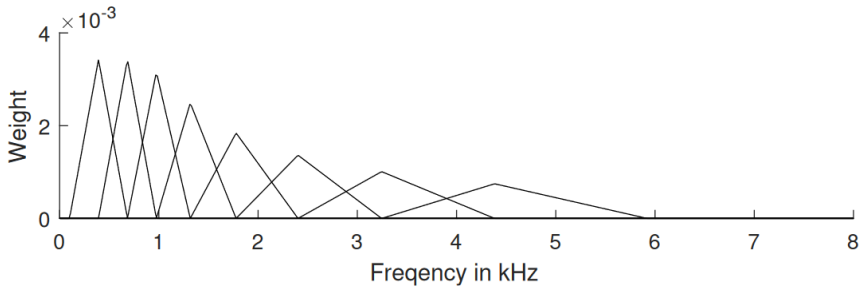
Source: Author (a) "adapted from" Park (2008), page 418, (b) "adapted from" Rothmund (2018), page 68

However, most current implementation in Matlab and Python, including Libosa, consider the auditory system’s frequency response as linear below $f_c = 1$ kHz, and upwards of f_c, the frequency resolution decreases incrementally as depicted in Figure 8 b).

Mel spectrograms are calculated by first computing STFT of the audio signal and then mapping the result to the corresponding Mel scale using a mapping matrix. The matrix is based on M triangular filters where each filter is scaled so that the area under each triangle is constant

as illustrated by Figure 9. Librosa (MCFEE et al., 2023) returns a vector $[M, N]$ where each row of Mel band m has the Mel-spectrogram magnitude of the frequency bins k.

Figure 9 – Mel-weighting filter bank consisting of 8 triangular filters combing linearly spaced frequency bins between 0 and 8 kHz into 8 Mel bands from 100 Hz to 6 kHz



Source: Rothmund (2018), page 69

The concept of **Mel-Frequency Cepstral Coefficients (MFCC)** was originally introduced in the 1980s for speech recognition tasks (DAVIS; MERMELSTEIN, 1980), aimed to address the limitations of simple spectral analysis techniques, which were found to be inadequate for representing the characteristics of human speech. Cepstrum and quefrency were coined by swapping the initial syllables of spectrum and frequency. According to Oppenheim and Schafer (2004), this consonant exchange illustrates the cross-disciplinary approach, where techniques typically used in the time domain are applied to frequency analysis, and vice versa. Although very popular in the field of speech recognition, it also found space in the field of audio classification being rated as the most utilized feature extraction technique in the systematic review of data augmentation and deep learning methods in sound classification (ABAYOMI-ALLI et al., 2022).

The computation of MFCC involves several steps according to Klapuri and Davy (2006):

a) A pre-emphasis filter is applied to flatten the spectrum;
b) The audio signal framed, windowed and transformed using the DFT;
c) A Mel-scale filter bank is applied in the frequency domain, as previously explained;
d) The power within each sub-band is computed by squaring and summing frequency bin magnitudes within bands;
e) The dynamic range of the spectrum is compressed by taking a logarithm of the band wise power values;
f) Finally, cepstral coefficients are computed by applying, to the log filter bank powers, the Discrete Cosine Transform (DCT)[2] which decorrelates the coefficients as:

[2]The Discrete Cosine Transform is similar to the DFT but only uses the cosine function to represent the signal.

The dimensionality of the representation can be reduced by retaining only approximately 13 lowest-order DCT, which usually carry the relevant timbral information. Succinctly the first DCT coefficient serves to denote the average power or energy in the spectrum, the second coefficient offers an approximation of the spectrum's broad shape and maintains a relation to the spectral centroid whereas for the higher-order coefficients, they capture finer spectral details such as pitch. Typically, in practice, a subset ranging from the first 8 to 13 MFCC coefficients adequately represents the spectrum's shape, with the higher-order coefficients being disregarded due to their redundancy in providing additional information (ABREHA, 2014). Librosa (MCFEE et al., 2023) returns a vector $[\sum i, N]$ where each row i has the MFCC coefficients of the frequency bins k.

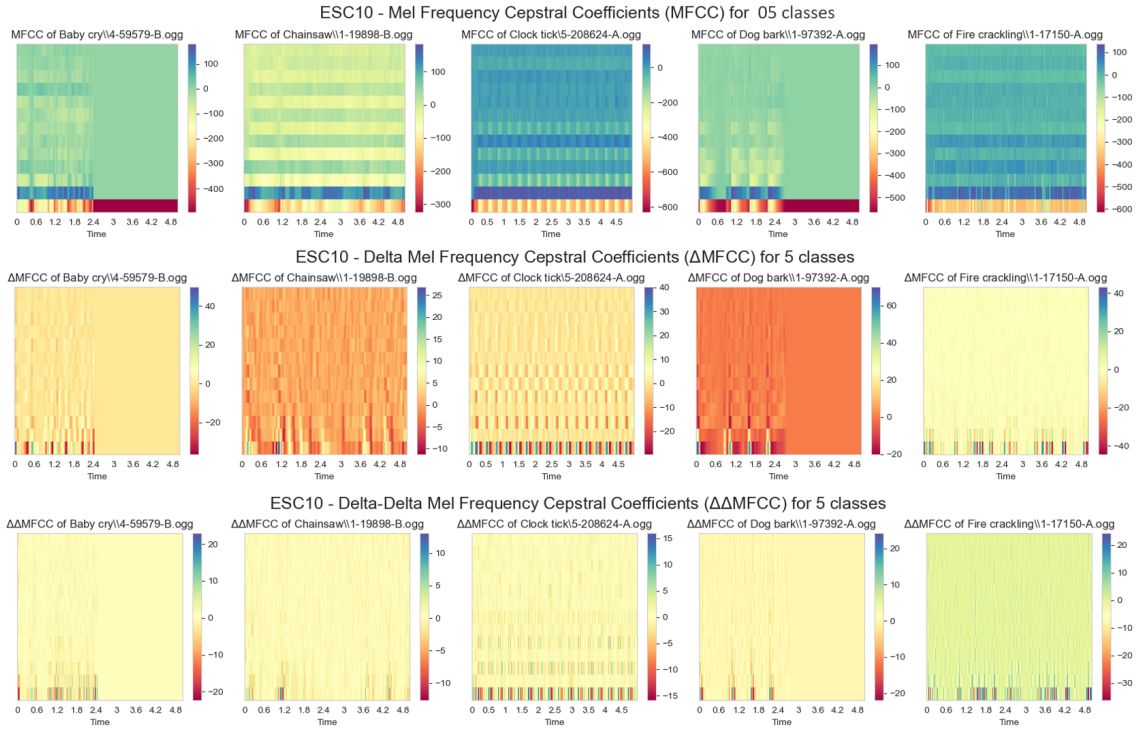
MFCC has one significant limitation which is its sensitivity to noise and channel variations. If the audio signal is corrupted with noise or transmitted through a poor-quality channel, the extracted MFCC coefficients may not accurately represent the underlying sound information. Another limitation is the lack of temporal information in the MFCC representation. Since the computation of MFCC is frame-based, it does not capture the dynamics and temporal variations of the speech signal. To alleviate this limitation, dynamic features such as delta and acceleration coefficients are commonly computed by taking the derivatives of MFCC (GOLD; MORGAN; ELLIS, 2011). To illustrate this concept, Figure 10 compares the spectrograms of the MFCC (first row), $\triangle$MFCC (second row) and $\triangle\triangle$MFCC (third row) set to 13 coefficients for five samples from the ESC-10 dataset,

Delta MFCC and **Delta-Delta MFCC**, also know as $\triangle$**MFCC** and $\triangle\triangle$**MFCC** are the first and second order derivative of the MFCC along the time axis.The first captures information about the velocity or speed of changes in the audio features and the latter captures the rate of change of the $\triangle$MFCC coefficients, capturing the curvature or acceleration of the spectral features. Both $\triangle$MFCC and $\triangle\triangle$MFCC are often concatenated with the original MFCC coefficients to form a feature vector that incorporates temporal dynamics (BOUNTOURAKIS et al., 2019) and (TANG et al., 2018).

2.1.3 Conclusion

Based on the detailed exploration of audio feature extraction techniques in the context of ESR, it became evident that the selection of appropriate audio features plays a critical role in achieving optimal performance and accurate recognition. The unique requirements

Figure 10 – Example of MFCC, △MFCC, and △△MFCC plot applied to random samples of the dataset ESC-10.



Source: Author

of ESR for autonomous vehicles, such as multi-kernel support, real-time operation, energy efficiency, and memory optimization, present challenges that must be carefully considered when choosing audio features. The ideal audio features for such applications should exhibit low computational complexity, fast response time, minimal energy consumption, and memory usage while maintaining acceptable recognition accuracy, all that can be achieved with the selected features using STFT. Additionally, the selected audio features should possess a small feature size to reduce computational costs, and therefore, MFCC, SCT, Chroma and Tonnetz must be carefully assessed regarding their influence in the classification metrics.

2.2 MICROPHONES

This section introduces relevant definitions, concepts and characteristics related to microphones, hearing capabilities and their applications in the automotive industry.

2.2.1 Definitions

In psychoacoustics, **sensitivity** is the ability of the human ear to detect and comprehend sounds at various frequencies and intensities, measuring how well an individual can perceive and differentiate sounds at various levels of volume or loudness (MOORE, 2013).

In a microphone, **sensitivity** represents the ability to convert sound waves into an electrical signal. It measures how well a microphone can collect and detect sound, especially low-level or quiet noises, which means, a microphone's sensitivity dictates how responsive it is to sound pressure levels and influences its ability to capture quiet or distant sounds accurately. Sensitivity is often evaluated as the SPL necessary to achieve a specific output level from the microphone and is usually represented in dB. The higher the sensitivity rating in dB, the more sensitive the microphone to sound (RAYBURN; EARGLE, 2004).

Frequency Response (FR) refers to how well the human ear can perceive sound at different frequencies in psychoacoustics. It is a measure of the sensitivity of the human ear to sound at different frequencies, and it plays an important role in determining how humans perceive and interpret different sounds. The human ear's frequency sensitivity does not remain constant across all frequencies, rather some frequencies are more responsive to the ear than others. A FR curve, which illustrates how well the ear responds to different frequencies, can be used to measure this sensitivity, and it typically depicts the amount of sound pressure required for a listener to perceive a sound at a specific frequency, measured in dB and plotted with frequency (in dB) on the horizontal axis and SPL on the vertical axis (MOORE, 2013).

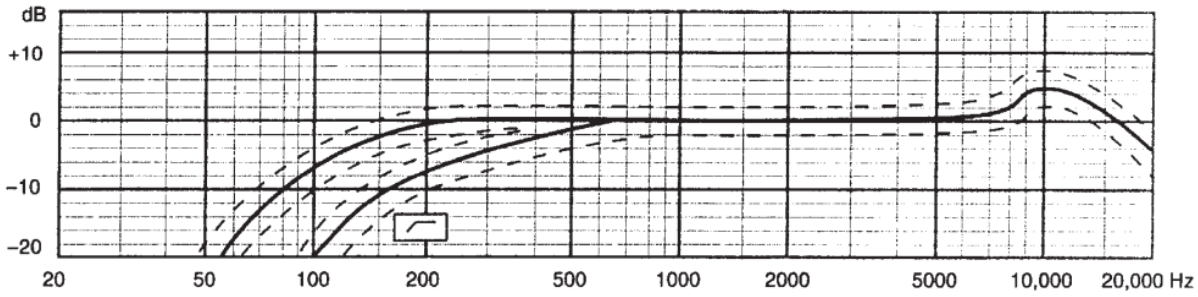
In a comparable manner, **FR** in a microphone refers to how it responds to different frequencies across the audible spectrum, in other words, it describes the microphone's ability to capture and reproduce sound accurately at various frequencies (RAYBURN; EARGLE, 2004).

A microphone's frequency response is often displayed as a graph or numerical specification that demonstrates how the sensitivity of the microphone fluctuates across the frequency range. According to Rayburn and Eargle (2004), it illustrates which frequencies are more responsive to the microphone and how it attenuates or highlights specific frequency ranges as shown in Figure 11 below.

2.2.2 Human hearing capabilities

The range of sound intensity (pressure) and frequency to which the ear responds is truly astonishing as presented in Figure 12 below. The intensity ratio between painful noises and the

Figure 11 – Amplitude response versus frequency with upper and lower limits for a capacitor vocal microphone; effect of lower frequency cut is also show.



Source: Rayburn and Eargle (2004), page 106

weakest sounds humans can hear is greater than 10^{12} and the frequency ratio between the highest and lowest frequencies humans can hear is roughly 1.000 times greater (20 Hz and as high as 20.000 Hz), or more than nine octaves - each octave represents a frequency doubling (ROSSING; MOORE; WHEELER, 2013). The auditory system's selectivity is another noteworthy feature, and also according to Rossing, Moore, and Wheeler (2013), a listener can discern the sound of a solo instrument among the blended sounds of a symphony orchestra, it is possible to distinguish a single speaker in a crowded room, a mother's conditioned ear can respond to an infant's cry even while she is sleeping and one may even teach himself to sleep through city traffic noise but wake up at the sound of an alarm clock or odd noise.

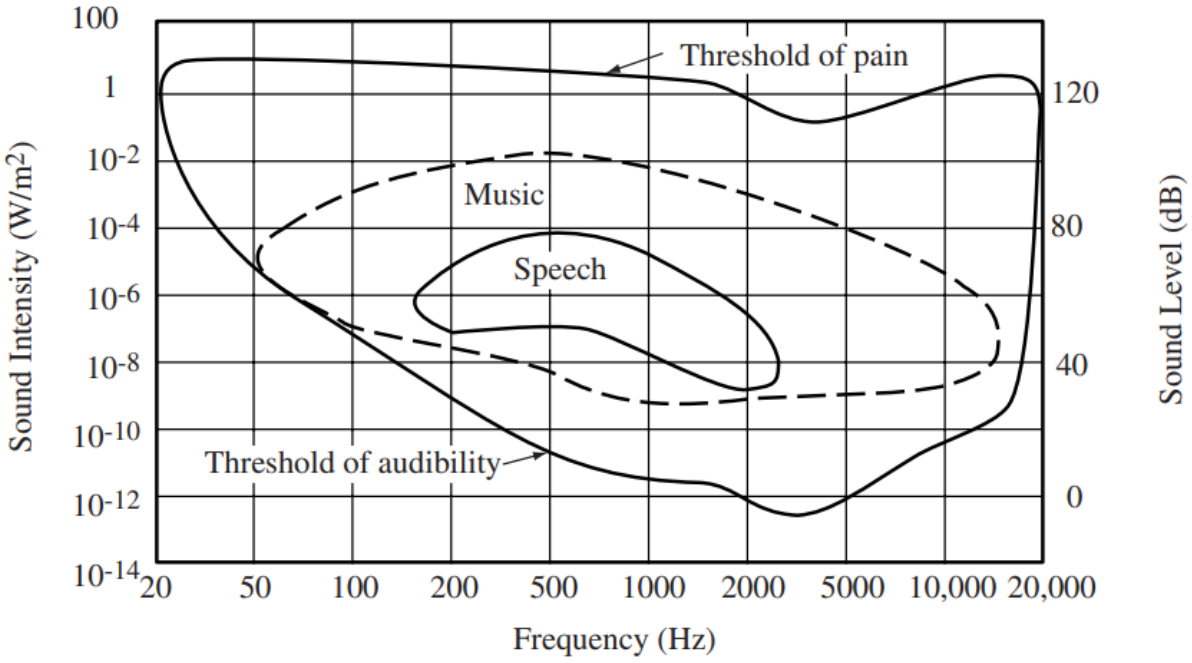
In general, the frequency response of the human ear is highest in the range of 2-4 kHz and gradually decreases at higher and lower frequencies as illustrated in the Figure 12 by the curves threshold of pain and threshold of audibility (MOORE, 2013).

2.2.3 Automotive vehicle microphones

There is a relatively small body of literature that is directly related to the types of microphones commonly used in regular passenger cars. In general, they are used in automotive vehicles for a variety of purposes, including hands-free calling, voice-activated controls, and in-car entertainment systems (NEWCOMB, 2008). The ones listed below are the microphone types commonly used in regular passenger cars according to Rumreich (2005):

a) Built-in microphones: many modern vehicles come with built-in microphones that are integrated into the dashboard or ceiling of the cabin. These microphones are

Figure 12 – Range of frequencies and intensities to which the auditory system (ear) responds.



Source: Rossing, Moore, and Wheeler (2013), page 80

often used for hands-free calling and voice-activated controls, and are typically designed to pick up sound from within the cabin;

b) Headset microphones: some vehicles come with headsets that have built-in microphones, which can be worn by the driver or passengers to communicate with other passengers or with the vehicle's audio system. These microphones are typically small and lightweight, and are designed to be comfortable to wear for extended periods;

c) External microphones: in some cases, external microphones may be used in automotive vehicles to pick up sound from outside the cabin, such as when using a hands-free calling system while driving with the windows down. These microphones are typically designed to be weather-resistant and durable, and may be mounted on the vehicle's exterior;

d) Bluetooth microphones: many modern vehicles are equipped with Bluetooth technology, which allows users to connect their smartphones or other devices wirelessly to the vehicle's audio system. Bluetooth microphones are commonly used for hands-free calling and voice-activated controls, and are designed to be small and unobtrusive.

Overall, the type of microphone used in a product will depend on the specific application and the design of the product's audio system. There seems to be a trend in the automotive industry to design audio systems using built-in and Bluetooth microphones, however, in some cases, external microphones may be necessary to achieve the desired level of sound quality and clarity (RAYBURN; EARGLE, 2004).

Built-in microphones are commonly used in the automotive industry as hardware modules that come equipped with interfacing circuitry and connectors. To detect acoustic signals, these microphones use either a single microphone element for voice sensing in hands-free calling or multiple elements for signal sensing including Active Noise Control (ANC) in the case of beam forming applications. In particular, Electret Condenser (ECM) and Microelectromechanical Systems (MEMS) microphone elements are widely utilized in automotive microphone modules and are essential in controlling the acoustic performance of the module (AUBAUER; LECKSCHAT, 2001). The advancements in microphone elements' signal-to-noise ratio and acoustic overload point specifications do not necessarily improve end-users' experience or Speech Intelligibility (SI) in an automotive environment. Even when a simulation model is developed to investigate the impact of microphone element sensitivity and phase mismatches on beam forming performance the results show significant dependency on the chosen array processing algorithm (DU et al., 2019).

Du, Varga, and Dobos (2022) investigated the SI of hands-free microphones used in vehicle audio systems, measured by a score (SI score), which is considered an important performance metric and cannot be directly obtained from the microphone datasheet. They categorized the three most common microphone types in the automotive industry:

a) Omnidirectional with a flat FR shape in the full speech band (e.g., 20-14.000 Hz);
b) Omnidirectional with a rising FR shape and a -3 dB cut-off frequency between 100 and 500 Hz;
c) Unidirectional (e.g., Cardioid) microphone.

They draw the conclusion based on their experiments that at low background noise levels, there are no noticeable differences in intelligibility among the three microphones. However, in the presence of medium to high-level and non-wind induced noises, the unidirectional or omnidirectional microphone with a rising frequency response shows certain advantages over the omnidirectional microphone with a flat FR. On the other hand, in the presence of wind

turbulence-induced noises, which are typically at high levels, the unidirectional microphone exhibits the greatest performance degradation.

2.2.4 Conclusion

Based body of literature investigated, it can be inferred that the maximum attainable sensitivity level of the microphones typically employed in the automotive industry is conventionally constrained by the caliber of the microphone's transducer and the electronic signal processing circuitry employed to analyze the acoustic beam forming and despite exhibiting noteworthy electroacoustic characteristics, they demonstrate limited potential in attaining the discernment and precision achievable by the human auditory system, however, it is imperative to note that these microphone types do exhibit sufficient attributes capable of detecting a substantial range of acoustic events. This inference imparts significant technological insights regarding the use of automotive microphones in environmental sound recognition.

2.3 MACHINE LEARNING

Machine learning plays a crucial role in enabling autonomous systems to acquire knowledge, make informed decisions, and adapt to dynamic environments. Within the context of ESR, the current literature predominantly focuses on supervised learning for classification tasks due to its effectiveness and practical applicability. However, semi-supervised and reinforcement learning approaches are also present, mostly for audio tagging tasks.

2.3.1 Classification

In a broader definition given by Russell and Norvig (2010), any component of an agent can be improved by learning from data, essentially depending on four major factors: which *component* is to be improved, what *prior knowledge* the agent already has, what *representation* is used for the data and the component, and what *feedback* is available to learn from.

There are three types of feedback that determine the three main types of learning, with one variation that could be classified as the fourth (RUSSELL; NORVIG, 2010):

Unsupervised learning is characterized by the absence of explicit feedback or labeled examples during the learning process. Instead, the agent focuses on recognizing patterns and structures within the input data. By detecting and clustering potentially meaningful groups or

clusters of input examples, the agent can uncover valuable insights without external guidance or supervision. An example of unsupervised learning in the domain of environmental sound recognition could involve an autonomous taxi agent gradually developing the concept of "good traffic days" and "bad traffic days" based solely on the audio patterns it observes, without being explicitly provided with labeled examples by a teacher.

Reinforcement learning revolves around the use of a series of reinforcements, which can take the form of rewards or punishments, to guide the learning process. The agent learns by interacting with its environment, receiving positive or negative reinforcement signals based on its actions. Through trial and error, the agent determines which actions led to desirable outcomes and which ones resulted in unfavorable consequences, for instance, in the context of an autonomous vehicle, conveying a rewarded thumbs up by the driver on the infotainment display when the agent detects an ambulance approaching from the rear of the vehicle would serve as an affirmative indication that a correct action was taken. It is the responsibility of the agent to determine the most appropriate actions that result in favorable reinforcement.

Supervised learning involves the agent observing pre-existing example input-output pairs to learn a function that maps input to output. In this learning paradigm, a teacher provides labeled examples, where the output is known or provided by an expert. The agent learns to generalize from these examples, acquiring the ability to make predictions or classifications on unseen data. In the context of environmental sound recognition for autonomous vehicles, supervised learning could manifest as the agent learning to associate specific environmental sound patterns with appropriate responses or actions, for instance, recognizing the sound of emergency sirens and responding accordingly.

In **semi-supervised learning**, the agent is provided with a limited number of labeled examples while also being presented with a vast collection of unlabeled examples. This learning paradigm falls in between supervised and unsupervised learning, as it attempts to make the best use of both labeled and unlabeled data. The labeled examples serve as a guide for the agent to learn a function mapping the inputs to outputs, similar to the supervised learning approach. However, the unlabeled examples pose a challenge as there is no explicit feedback or labels associated with them.

The presence of inaccuracies and uncertainties in the labeled examples adds complexity to the semi-supervised learning process, for example, considering a scenario where the task is to discern between the sound of a siren, animal, or children playing. In this case, individuals imitating these sounds might introduce intentional inaccuracies, thereby adding random noise and

systemic errors in the labeled dataset. Resolving and mitigating these inaccuracies would entail an unsupervised learning problem, wherein the system needs to analyze the relationships between the audio samples, the imitated sound sources, and the true (unknown) sound sources, thus both noise and lack of labels create a continuum between supervised and unsupervised learning. There are several approaches for this type of learning, namely the most utilized: self-training, semi-supervised variants of supervised algorithms, graph-based models and generative models.

2.3.2 Training process

The training process of traditional machine learning algorithms involves the utilization of labeled datasets to facilitate the learning of patterns and associations in the data. These algorithms aim to extract features from the input data and apply statistical methods to model the relationships and make predictions based on the learned patterns. In the context of the learning task defined (MITCHELL, 1997), which involves the classification of environmental sounds for autonomous vehicles, the training process would entail feeding the algorithm with an annotated dataset of environmental sound recordings along with their corresponding class labels, hence, an explicitly supervised learning. The algorithm would then analyze the input features, such as frequency content or temporal characteristics, and create a model that can classify new incoming sounds based on its learned patterns.

2.3.3 Commonly used classification algorithms

This section provides an overview of the classifiers examined in this study, starting with the famous k-Nearest Neighbors (k-NN) , following by Gaussian Naïve Bayes (GNB), Support Vector Machine (SVM), Logistic Regression (LR), Random Forest and a Voting classifier. Although this selection represents only a small fraction of the classifiers proposed and investigated in existing literature, they serve well to the purpose to focus on selected methods which are both popular and representative among the wealth of techniques that are available. Instead of delving into lengthy theoretical descriptions, this study emphasizes the fundamental concepts underlying the algorithms. Notably, these classifiers have been chosen for experimental and simulation testing in the implementation due to their prevalent usage in the literature, more specifically in Bountourakis, Vrysis, and Papanikolaou (2015), Silva et al. (2019) and Lhoest et al. (2021).

2.3.3.1 k-Nearest Neighbor (k-NN)

The concept underlying the **k-NN** classifier is based on the principle of a nonparametric models, also named **instance-based learning** or **memory-based learning**, which do not adhere to a predetermined set of parameters but rather utilize the entire training dataset to generate predictions (RUSSELL; NORVIG, 2010).

The basic approach uses as table lookup to store all training examples and retrieve the corresponding output for a given input query, however, table lookup often lacks generalization capabilities and can only provide a default output when a query is not present in the table.

To enhance the table lookup method, the k-NN identifies the k examples of a point ($\mathbf{x}_j$) closest to a given query ($\mathbf{x}_q$), counting how many of those belong to each class by employing a plurality vote among the nearest neighbors and odd values of k to avoid ties. The word "closest" entails a distance, and for that, several metrics could be applied, but the most common approach according Russell and Norvig (2010) is the **Minkowski distance** or L^p norm, defined in a multi-dimensional space D as:

$$L^p\left(\mathbf{x}_j, \mathbf{x}_q\right) = \left(\sum_i^D \left|x_{j,i} - x_{q,i}\right|^p\right)^{1/p} \tag{7}$$

Setting $p = 1$ results in the **Manhattan distance**, more fitting to dissimilar properties such as gender, weight and age of patient. Whereas $p = 2$ results in the well-know **Euclidean distance**, that performs better if the dimensions are measuring similar properties, such as the width, height and depth of parts on a transportation belt.

Let a set of patterns $\mathbf{x}_j$ represents an unknown feature vector, after calculating the numerical distances $L^p\left(\mathbf{x}_j, \mathbf{x}_q\right)$ for each x_q, the resulting distances are sorted in ascending order. The k closest neighbors of the unknown feature vector are identified as the k first values in the sorted list. Furthermore, let k_i denote the number of training vectors belonging to the i^{th} class among the k neighbors of the unknown feature vector, where i ranges from 1 to N_c. Based on this, the unknown vector is classified into the class that corresponds to the highest value of k_i.

It is important to note that by using the raw numbers from each dimension can lead to a distortion of the total distance calculation due to variations in scale across different dimensions. To avoid this, a simple approach according to Hastie, Tibshirani, and Friedman (2009) is to first standardize each of the properties or features to have mean equals to 0 and variance equals to 1 so that $x_{j,i}$ becomes $\left(x_{j,i} - \mu_i\right)/\sigma_i$.

In low-dimensional spaces with abundant data, the nearest neighbors approach tends to perform exceptionally well, as a sufficient number of nearby data points are available to achieve accurate results. However, in high-dimensional spaces, an inherent issue arises in which the nearest neighbors are often relatively distant and the amount of data needed to effectively cover the space grows exponentially, rendering their effectiveness diminished. This problem is described as the **curse of dimensionality** by Russell and Norvig (2010), Hastie, Tibshirani, and Friedman (2009) and it was originally defined by Bellman (1961).

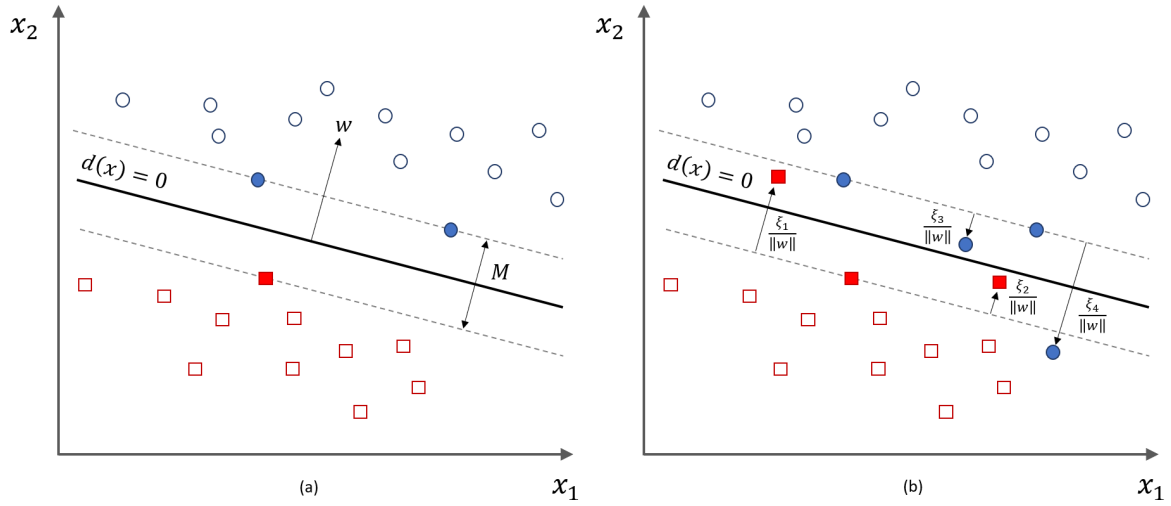
Similar to other memory-based learning classifiers, k-NN has some advantages though, such as simplicity and flexibility, as it can adapt to new data without the need for retraining the entire model. On the other hand, choosing the appropriate number of nearest neighbors is crucial for the algorithm efficiency. A small k value may cause overfitting, leading the classifier to be overly sensitive to noise. Conversely, a large k value may cause underfitting, resulting in poor generalization of the classifier, therefore, the number of k is typically adjusted through experimentation with the dataset at hand.

2.3.3.2 Support Vector Machine (SVM)

SVM is a powerful framework of supervised learning algorithms commonly employed for binary classification tasks, being very popular and recognized for their efficiency in terms of computational cost for prediction and their ability to yield accurate solutions even when training data is scarce. The primary objective of SVM is to determine the optimal hyperplane in an N-dimensional parameter-space $\mathbb{R}^N$ that possesses the maximum margin M between data samples belonging to different classes, facilitating the effective separation of different classes within the feature space (RUSSELL; NORVIG, 2010). A simple illustration of this concept is depicted in Figure 13 (a) where some data belonging to a bi-dimensional space, represented by circles and squares, are separated by a straight-bold line, and the margins are defined by the intersection of the solid circles and squares with an offset of the boundary line.

In essence, the classification of data points into classes $(y = +1)$ or $(y = -1)$ in a multidimensional space can be accomplished by determining their position in relation to a hyperplane. Specifically, data points located on one side of the hyperplane $(d > 0)$ are associated with one class, whereas those on the other side $(d < 0)$ are attributed to the other class. The weight vector w, perpendicular to the hyperplane, determines the orientation of the hyperplane

Figure 13 – Classification example of SVM with the maximum margin hyperplane defined by the support vectors represented as filled circles and squares. (a) without soft margin, (b) with soft margin.



Source: Author

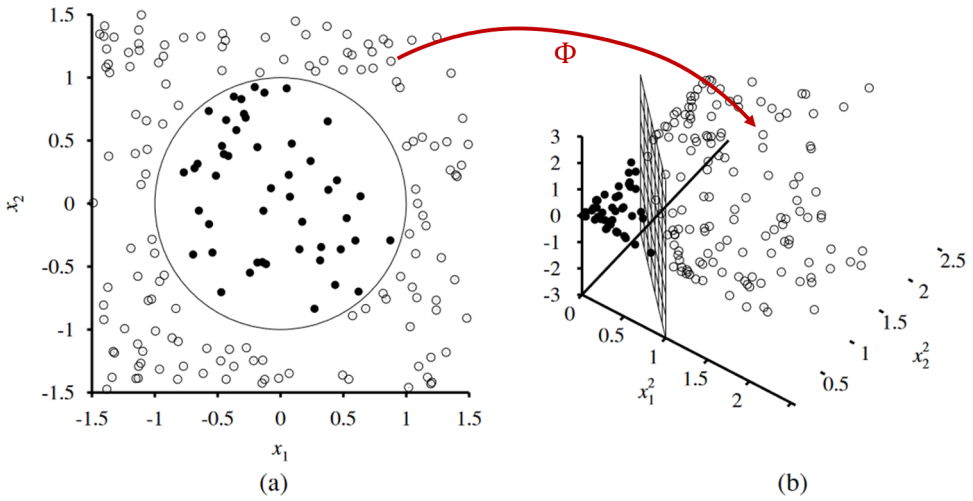
in the multidimensional space $\mathbb{R}^N$, while the bias w_0 shifts its position from the origin $\frac{w_0}{\|w\|}$. To maximize the margin M, the constraints are defined given all training examples $x^{(i)}, y^{(i)}$.

In practice, it is enough to retain the support vectors $x(i)$ with non-zero weights α_i as illustrated in Figure 13 by the filled circles and squares. Considering the linearity of the decision boundary, standard SVMs can only classify linearly separable data, however, SVMs have the ability to embed the data into a higher-dimensional space by applying a non-linear function $\Phi : \mathbb{R}^{N_1} \rightarrow \mathbb{R}^{N_2}$ to the input x, converting it to a non-linear space $\mathbb{R}^{N_2}(N_2 > N_1)$, where non-linearly spaced data can be separated using a linear decision function. Figure 14 depicts this concept, where (a) is a two-dimensional training set with positive examples as black circles and negative examples as white circles. The true decision boundary is defined by the circumference, and (b) is the the same data after mapping into a three-dimensional input space. The circular decision boundary in (a) becomes a linear decision boundary in three dimensions.

At this point, Russell and Norvig (2010) highlights that the transformation $\Phi(x)$ to the input data presents significant computational challenges, and in some cases, it may be impracticable (e.g., with infinite N_2). However, the so called **kernel trick** offers a solution by simplifying the transformation to a higher-order feature space through kernel evaluations.

There are several kernel functions to be plugged in the SVMs, namely: linear, polynomial, quadratic, and perceptron, but according to Goodfellow, Bengio, and Courville (2016), the most

Figure 14 – Applying a non-linear transformation Φ to the input feature space to separate non-linearly spaced data with a linear hyperplane



Source: Author "adapted from" Russell and Norvig (2010), page 747

commonly utilized kernel is the Gaussian kernel, also know as the **Radial Basis Function (RBF) kernel**.

When dealing with inherently noisy data, a linear separator in a high-dimensional space might not be desirable. Instead, one seek a decision surface in a lower-dimensional space that acknowledges the lack of clean separation between classes, reflecting the noisy nature of the data. This is achievable with the **soft margin** classifier, permitting examples to be situated on the incorrect side of the decision boundary while imposing a penalty proportional to the distance needed to relocate them to the correct side (RUSSELL; NORVIG, 2010).

According to Rothmund (2018), the penalty parameter C essentially regulates the number of support vectors situated within the margin or on the 'wrong' side of the decision function ($\xi i \neq 0$). When C is small, the decision boundary becomes smoother, reducing the risk of overfitting. Conversely, a larger penalty C results in a more intricate decision boundary that closely represents the training data. The objective is to determine a suitable parameter C that captures the underlying trend in the training data without being excessively complex (leading to overfitting) or overly simplistic (resulting in underfitting). An example of soft margin is depicted in 13 (b) by the filled circles and squares within the margin or even on the wrong side of the decision boundary.

SVM can be used for **multi-class discrimination** by combining several SVMs thus creating a multi-class classifier for $N_c > 2$. Two methods are frequently available in the literatures, the One-vs-Rest and the One-vs-One.

In the **One-vs-Rest** or **One-vs-All**, each class c is trained individually, where samples belonging to class c are labeled as positive $(y = 1)$, while the remaining samples are labeled as negative $(y = -1)$. The output class is determined by the maximum output score, such as the distance from the decision boundary.

The **One-vs-One** approach is more computationally demanding. In this method, $\frac{N_c(N_c-1)}{2}$ binary classifiers are trained, where each classifier corresponds to a unique pair of classes $\{c_i, c_j\}$. The binary output predictions from these classifiers are then aggregated by tallying the number of positive predictions for each class.

Although SVMs are highly sophisticated, they are not flawless: kernel machines encounter a significant computational burden during training, particularly with large datasets, and generic kernels often struggle to generalize well. According to Goodfellow, Bengio, and Courville (2016), the modern incarnation of deep learning was designed to overcome these limitations of kernel machines and its very renaissance began when Hinton, Osindero, and Teh (2006) demonstrated that a neural network could outperform the RBF kernel SVM on the MNIST benchmark.

2.3.3.3 Naïve Bayes

The naïve Bayes classifier is a supervised machine learning algorithm that uses the principles of probability and Bayes' theorem to perform classification tasks. It is called naïve because it makes two simplifying assumptions: that the features are conditionally independent given the class, and that all features have equal importance. Despite these unrealistic assumptions, the naïve Bayes classifier often works well in practice and is widely used for text classification, spam filtering, sentiment analysis, and more (BARBER, 2012).

Depending on the choice of distribution, the naïve Bayes classifier can have different variants, such as the Bernoulli naïve Bayes can be used for binary features, the multinomial naïve Bayes can be used for categorical features, and the Gaussian naïve Bayes can be used for continuous features (FRIEDMAN; GEIGER; GOLDSZMIDT, 1997). This study implemented the **Gaussian Naïve Bayes (GNB)** algorithm from the Python library scikit-learn.

The naïve Bayes classifier has several advantages and disadvantages. Some of the advantages are that it is easy to implement, fast to train and test, robust to noise and irrelevant

features, and scalable to large datasets. Some of the disadvantages are that it can suffer from zero-frequency problems, where a feature value that does not occur in the training data leads to a zero posterior probability, and that it can perform poorly when the independence and equal importance assumptions are violated. To overcome these limitations, various smoothing techniques and feature selection methods can be applied (WICKRAMASINGHE; KALUTARAGE, 2021).

As a remark, Generalized Linear Model (GLM) is a probabilistic model that assumes all the data points are generated from a mixture of several Gaussian distributions with unknown parameters and Gaussian Mixture Model (GMM) is a framework for modeling relationships between a response variable and one or more explanatory variables, which includes linear regression, logistic regression, Poisson regression, etc. While they all involve probability and modeling, including GNB, they differ in their underlying assumptions, methodologies, and applications.

2.3.3.4 Logistic Regression

Logistic Regression (LR) is a statistical method used for binary classification tasks, where the goal is to predict the probability that an instance belongs to one of two classes. Despite its name, logistic regression is a classification algorithm rather than a regression one.

The literature on LR model is extensive as evidenced by works such as (MITCHELL, 1997), (BOUGUILA; FAN, 2020) and (RUSSELL; NORVIG, 2010), therefore it suffices to state the basic idea, where the model applies the logistic function (also known as the sigmoid function) to the linear combination of input features. This function maps any real-valued number to the range between 0 and 1, which makes it suitable for representing probabilities.

LR is trained using optimization algorithms, namely some of them: maximum likelihood estimation, gradient descent and limited-memory BFGS, or an online algorithm, such stochastic gradient descent. The training aims to find the optimal values for the coefficients that minimize a cost function, typically the log loss or cross-entropy loss, which measures the difference between the predicted probabilities and the actual class labels in the training data.

LR is inherently a binary classification algorithm, however, it can be extended to handle multiclass classification tasks through several strategies, namely:

a) **One-vs-Rest** or **One-vs-All**: by training one classifier per class and treating that class as the positive class, and all other classes as the negative class. During

prediction, the class with the highest probability from the individual classifiers is the winner;

b) **Multinomial (Softmax)**: it handles multiple classes directly without the need for multiple binary classifiers. Instead of predicting a binary outcome, it predicts the probabilities of each class using the softmax function, which normalizes the output into a probability distribution over multiple classes. During training, it minimizes the cross-entropy loss, which measures the difference between the predicted probabilities and the actual class labels;

c) **Multiclass Hinge Loss**: this function penalizes predictions that are not sufficiently correct, aiming to maximize the margin between classes.This approach from SVM is often used when the emphasis is on maximizing margins between classes rather than directly optimizing probabilities.

This algorithm offers simplicity and interpretability, and furthermore, it has computational efficiency and resistance to overfitting, especially with smaller datasets. However, logistic regression assumes a linear relationship between features and outcomes, limiting its ability to capture complex patterns in the data. It struggles with non-linear relationships, large feature spaces, and outliers, which can affect its performance adversely (RUSSELL; NORVIG, 2010).

2.3.3.5 Random Forest

Random Forest is a popular ensemble learning method in machine learning. It's essentially a collection of decision trees, where each tree is trained on a different subset of the training data, and the final prediction is made by averaging (for regression) or voting (for classification) the predictions of all the individual trees. The method has essentially the following steps below according to Breiman (2001), Hartshorn (2016) and Genuer and Poggi (2020):

Bootstrap sampling: Random Forest builds multiple decision trees based on random subsets of the training data. This sampling technique, known as bootstrap sampling, involves selecting k random samples with replacement from the original dataset, notably, some instances may appear multiple times in a subset, while others may not appear at all. Theoretically, k samples will cover 2/3 of the original dataset, while the rest is called Out-Of-Bag.

Feature randomness and decision tree construction: When building each decision tree, Random Forest also introduces randomness in the features that can be considered at each split. Instead of considering all features at every split, it randomly selects a subset of m features

from M total features and chooses the best split from that subset. This helps to decorrelate the trees, making them more diverse and less likely to overfit the training data. It is recommended to start with $m = \sqrt{M}$ number of selected features and then reduce or increase m value until the minimum error for the Out-Of-Bag dataset is achieved. The best split is determined by assessing its quality using metrics from the "information theory" such as Gini impurity or entropy.

Voting or averaging for predictions: Once all the trees are constructed, predictions are made for new instances by each individual tree. For regression tasks, the final prediction is usually the average of the predictions of all trees, whereas for classification tasks, the final prediction is determined by majority voting among all trees.

According to Genuer and Poggi (2020), Random forest is among the preferred methods in the toolbox of statisticians and other data scientists due to its high accuracy, robustness to noise and irrelevant features, and ability to handle large datasets. It is also relatively easy to use and requires minimal parameter tuning, however, it can be computationally expensive and may not perform well on imbalanced datasets.

2.3.3.6 Voting classifier

Voting classifier, also known as a probabilistic voting classifier, is an ensemble learning method in machine learning that combines multiple base classifiers and predicts the class label based on the probabilities of each class output by the individual classifiers.

Each base classifier in the ensemble independently predicts the class label for a given input. Instead of directly taking a majority vote, the voting classifier calculates the probabilities assigned to each class label by each base classifier, then it averages these probabilities for each class across all the base classifiers, and finally, the class label with the highest average probability is chosen as the final prediction (SARKAR; NATARAJAN, 2019).

This approach is particularly effective when the base classifiers are capable of providing probability estimates for their predictions, such as in the case of many classification algorithms like LR, SVMs with probability outputs, or decision trees with probability outputs.

There are two types of voting classifiers: hard voting and soft voting. In hard voting, the final prediction is the class that receives the most votes from the individual models. In soft voting, the final prediction is the class with the highest average probability across all the individual models. Soft voting typically outperforms hard voting (where the majority vote is taken directly)

because it considers the confidence levels of individual classifiers, resulting in more nuanced and often more accurate predictions.

According to Sarkar and Natarajan (2019), this ensemble technique is robust against overfitting and outliers, as it averages out individual errors and biases. However, voting classifiers can be computationally intensive, especially when using a large number of diverse base classifiers. Additionally, interpretability may be compromised compared to single-model approaches like LR, as the decision-making process involves multiple models. Nonetheless, voting classifiers are highly flexible and effective in a wide range of classification tasks, particularly when the individual base classifiers complement each other's strengths and weaknesses.

2.3.4 Conclusion

The comprehensive analysis of the machine learning classifiers described in this section highlighted key strengths and weaknesses of them with common themes including considerations such as computational efficiency, generalization capability, robustness to noise, scalability to large datasets, and ease of implementation. Additionally, the trade-offs between model complexity and interpretability are evident across different classifiers. While some of them excel in specific aspects like adaptability to new data k-NN, others prioritize accuracy and robustness (Random Forest). Moreover, challenges such as overfitting, computational burden, and assumptions violations are recurrent issues that need to be addressed when selecting an appropriate classifier for ESR tasks.

2.4 NEURAL NETWORKS

Neural network, as initially proposed by psychologist Frank Rosenblatt, can be defined as a computational model inspired by the structure and function of the human brain, composed of interconnected artificial neurons, organized in layers, each responsible for performing specific computational tasks. These networks are capable of approximating complex non-linear relationships between inputs and outputs through a process called training, where the network adjusts its internal weights based on a given set of example data. This adaptive learning enables neural networks to recognize patterns, make predictions, and solve various types of problems (ROSENBLATT, 1958).

2.4.1 Artificial Neural Network (ANN)

The cornerstone of Artificial Neural Network (ANN) lies in a fundamental mathematical model of an artificial neuron introduced by McCulloch-Pitts in 1943. This model serves as the groundwork for computational methodologies in the realm of learning, describing the properties of a single neuron by forming a linear combination of the outputs of other neurons, which is then transformed using a nonlinear function (BISHOP; BISHOP, 2023). An artificial neuron aggregates the input values $x_1, x_2, \ldots, x_M$ through a linear combination process illustrated by Figure 15 (a). Each input x_i is assigned a weight w_i, which is used to scale or weight the input value. The total of the weighted inputs, along with the bias b, is defined as the activation a, and after it passed through a non-linear activation function $\sigma(a)$, it produces the output state y_j of the neuron.

$$a = \sum_{i=1}^{M} w_i \cdot x_i + b \qquad y_j = \sigma(a) \tag{8}$$

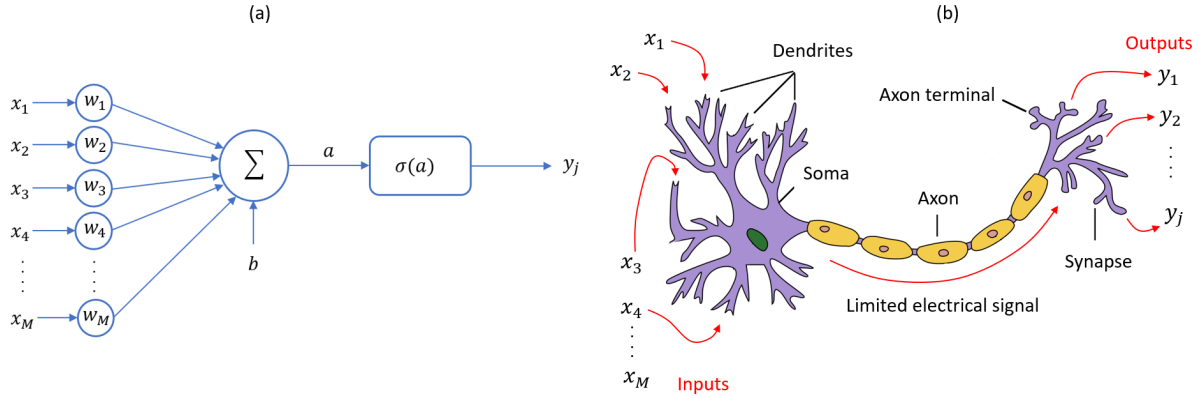
This linear combination cab be expressed by the dot product of a weighted vector W and the input vector X, given by:

$$y_j = \sigma\left(W^T X + b\right) \tag{9}$$

The trainable weight vector w plays a crucial role in determining the significance of an input to a neuron's output. Similarly, the trainable bias b sets the operational point within the non-linear activation function $\sigma(a)$. The activation function $\sigma(a)$ mimics its biological counterpart by limiting the strength of electrical spikes traveling through the axon to a certain range as illustrated in Figure 15 (b).

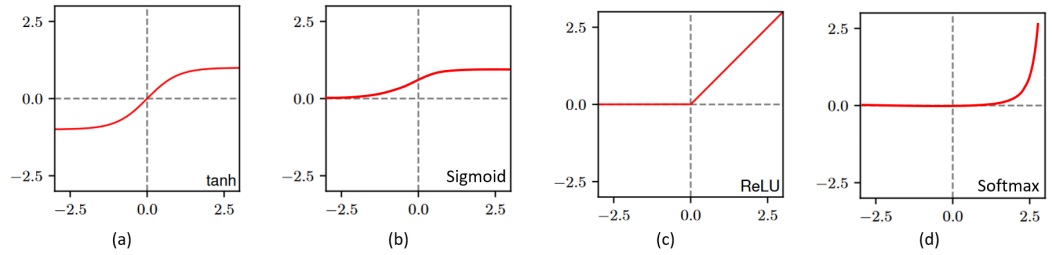
According to Bishop and Bishop (2023), frequently utilized general-purpose non-linear activation functions to constrain the neuron's state y within a reasonable range include hyperbolic tangent (Tanh) and Rectified Linear Unit (ReLU). Sigmoid and softmax are commonly employed at the neural network's output layer for classification tasks, transforming the input into a probability-like range between 0 and 1 as shown in Figure 16 (a) to (d). Specifically, sigmoid is suitable for binary classification, while softmax performs better for multi-class classification scenarios. in both cases, they are preferred for their regularization benefits and faster training compared to hyperbolic tangent activations. To obtain a discrete class from these continuous probability outputs, a decision function is typically implemented. In single-label multi-class classification, the argmax function is commonly used to select the class with the highest probability value.

Figure 15 – Artificial neuron mimicking a biological neuron



Source: Author

Figure 16 – Common activation functions



Source: Author "adapted from" Bishop and Bishop (2023), page 184

One of the most popular ANN is the **Multilayer Perceptron (MLP)**, a feedforward neural network model that consists of multiple layers of nodes (neurons), each layer fully connected to the next layer, organized consecutively as input, hidden and output layer as depicted in Figure 17. The MLP is characterized by its ability to learn nonlinear models by using an activation function that introduces non-linearity into the network's decision-making process (MITCHELL, 1997).

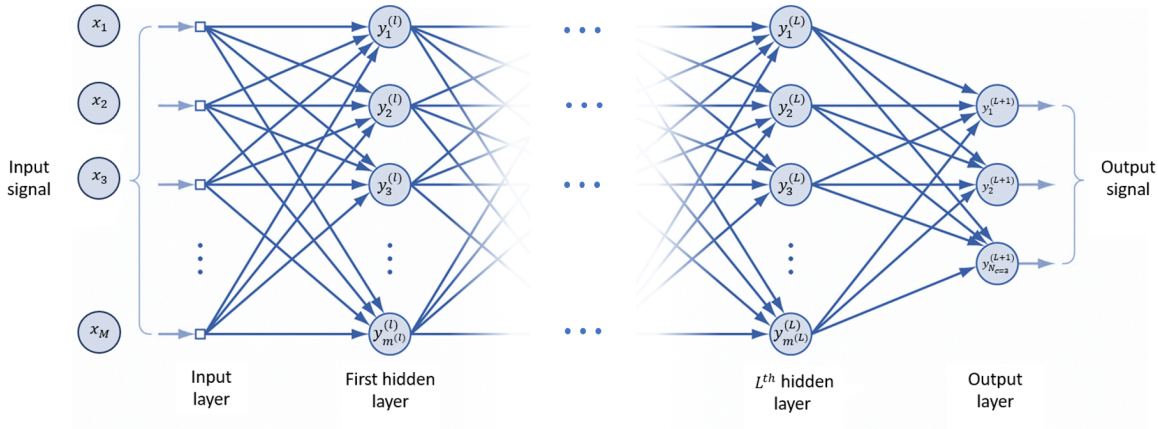
According to Russell and Norvig (2010), the first layer, or the input layer, receives input data features where each node represents a feature of the input data. Followed by hidden layers L that are the intermediate layers between the input and output layers where each node in a hidden layer l consisting of $m^{(l)}$ parallel nodes receives input from all the nodes in the previous layer and applies an activation function $\sigma^{(l)}(a_i^{(l)})$ to the weighted sum of those inputs and fed to the inputs of the next layer $(l+1)$. The last layer, or output layer $(L+1)$, produces the network's output vector. The number of nodes in this layer depends on the problem type, for instance, for

binary classification, one might have one output node representing the probability of belonging to one class, and its complement representing the probability of belonging to the other class.

MLPs are trained using supervised learning methods like backpropagation, where the network adjusts its internal parameters (weights and biases) iteratively to minimize the difference between its output and the desired output for a given set of input data. The training process undergo by optimizing a specified objective function, also known as a loss function. In the context of supervised learning, the predominant technique utilized is mini-batch **Stochastic Gradient Descent (SGD)** coupled with **Backpropagation**, particularly for classification tasks, the cross-entropy function, commonly referred to as log-loss, is frequently employed. The log-loss function exhibits a tendency towards infinity as the predicted probability of the true class approaches zero , imposing significant penalties for wrong predictions. **Categorical cross-entropy** serves as an extension of binary cross-entropy tailored for scenarios involving multi-class, and according to the problem and the classification tasks, the literature provides additional loss functions such as Logistic Loss, Mean Squared Error, and Mean Absolute Error. The process of generating predictions through a feedforward neural network and subsequently evaluating the loss function facilitates the assessment of the model's performance, and to enhance it, an optimization algorithm is applied to determine necessary adjustments to the model parameters. There are several options for the optimization algorithm such as SGD, Adam, RMSprop, etc, but they all generate a gradient of a function that quantifies the rate and direction of its output variations in response to incremental modifications in its inputs, computed through partial derivatives of the function (BISHOP; BISHOP, 2023).

The fundamental mechanism for computing gradients in a MLP is through the utilization of backpropagation (RUMELHART; HINTON; WILLIAMS, 1986), which involves iteratively propagating the error signals from the output layer back to the input layer by calculating the partial derivatives with respect to the inputs from the preceding layer. This process enables the determination of gradients for each weight and each biases in the network, and subsequently, once the gradients are obtained, weight adjustments are made by moving in the opposite direction of the gradient. To control the magnitude of this adjustment, each step of this process is scaled by a hyperparameter known as the **learning rate**, which typically falls within the range of 10^{-7} to 10^{-2}. Selecting an appropriate learning rate is crucial as excessively small values may lead to convergence issues, or local minima, whereas excessively large values can hinder training convergence. There are no shortcuts in this process, and the "rule of thumb" for defining the learning rate is to assess the model results using cross-validation, a systematic and

Figure 17 – MLP with M inputs, L hidden layers, each with $m^{(l)}$ nodes and $N_c = 3$ output nodes. Arrows represent weighted connections between nodes



Source: Author

rigorous technique for evaluating the performance of a learning model and for selecting the best hyperparameters.

Optimization algorithms that utilize the complete training set for updating each parameter is referred to as batch training or deterministic gradient methods, where the batch size is equivalent to the total number of observations in the training set. In **mini-batch** Gradient Descent, the training set is divided into fixed-size batches, and the loss function and model parameter updates are calculated for each batch, allowing for processing large datasets without the need to store all the training data in memory simultaneously. The batch size, a hyperparameter, must be chosen carefully to ensure that the batch loss provides a reasonable approximation of the overall training set loss while still fitting into memory (BISHOP; BISHOP, 2023). Each complete iteration through the entire training dataset is referred to as an **epoch**, with multiple epochs typically being conducted during training.

While Gradient Descent does not ensure discovery of a globally optimal minimum, appropriate hyperparameter selections typically lead to satisfactory local minima. According to Choromańska et al. (2014), achieving a global optimum on the training set may not be advantageous, as it is unlikely to result in strong generalized model.

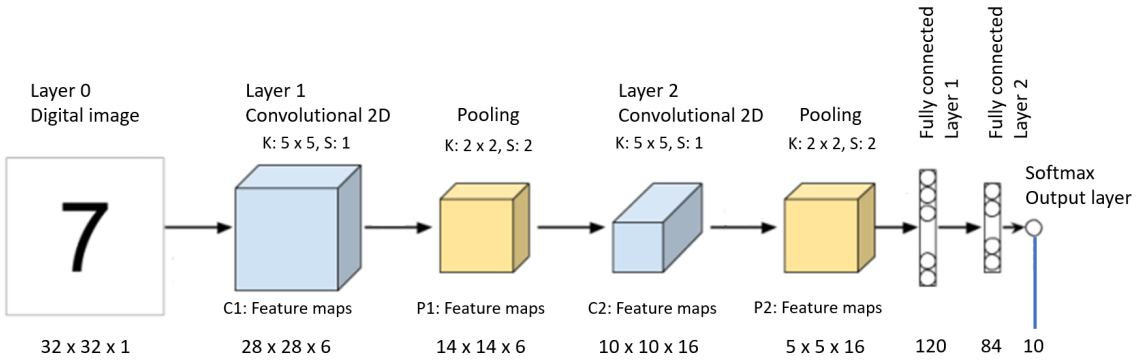
2.4.2 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN), as introduced by Lecun et al. (1998), is a specialized type of neural network designed for processing data with a structured, grid-like

topology, and it has demonstrated significant success in real-world applications. This can include various forms of data such as time-series data represented as a 1D grid with samples taken at regular intervals, and image data represented as a 2D grid of pixels. The term "convolutional neural network" reflects the utilization of the mathematical process known as convolution, which is a specialized form of linear operation within the network architecture. Goodfellow, Bengio, and Courville (2016) defined convolutional network as a simple neural network that uses convolution in place of general matrix multiplication in at least one of its layers.

Essentially, CNNs are complex architectures comprised of multiple convolutional layers, followed by pooling/down-sampling layers, and fully-connected layers of artificial neurons. The primary function of convolutional layers is to correlate segments of input data with pre-trained filter kernels to identify patterns, resulting in a high output score for matching features. By incorporating additional convolutional layers, increasingly intricate patterns can be extracted from the input data. Figure 18 illustrates the architecture of a CNN with two convolutional layers, pooling layers, and fully-connected layers serving as a classifier as proposed by Lecun in the LeNet-5.

Figure 18 – Example of CNN (LeNet-5) proposed by LeCun



Source: Author "adapted from" Lecun et al. (1998), page 2283

According Goodfellow, Bengio, and Courville (2016) and Rothmund (2018), the convolution operation is typically denoted with an asterisk (*), and a discrete convolution for functions $x(t)$ and $k(t)$ is expressed as:

$$s(t) = (x * k)(t) = \int x(\tau)k(t-\tau)d\tau \tag{10}$$

Given that x and k are defined solely at discrete intervals, the discrete convolution, denoted as s_i, can be defined as:

$$s_i = (x * k)_i = \sum_{n=-\infty}^{\infty} x_i \cdot k_{i-n} \tag{11}$$

Filter kernels may vary in dimensionality, being one-dimensional, two-dimensional, or three-dimensional, depending on the format of the input data. In the context of image recognition, the convolution process enables the representation of various beneficial transformations on two-dimensional data through the selection of different weights within the kernel. Examples include edge detection (horizontal, vertical, diagonal) and smoothing filters (mean, Gaussian). As the weights of the kernels in a convolutional layer are trained, they become adept at identifying local features specific to the training dataset. In cases involving multiple input channels, a single kernel is applied across all input channels, with the results at a given location and the bias being aggregated to generate the output (BISHOP; BISHOP, 2023).

Given a two-dimensional kernel K with dimensions $h_1 \times h_2$ and an input image X with dimensions $m_1 \times m_2$, the convolution operation can be expressed according to Rothmund (2018) as:

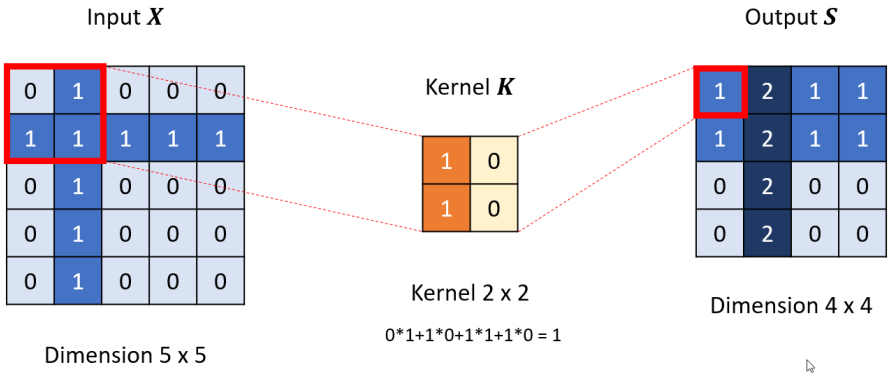
$$s_{i,j} = (X * K)_{i,j} = \sum_{m=1}^{h_1} \sum_{n=1}^{h_2} x_{i-m,j-n} \cdot k_{m,n} \tag{12}$$

With input X and the kernel K yielding the two-dimensional output S.

$$K = \begin{bmatrix} k_{0,0} & k_{0,1} & \cdots & k_{0,l_2} \\ k_{1,0} & k_{1,1} & \cdots & k_{1,l_2} \\ \vdots & \vdots & \ddots & \vdots \\ k_{l_1,0} & k_{l_1,1} & \cdots & k_{l_1,l_2} \end{bmatrix} \quad \text{and} \quad X = \begin{bmatrix} x_{0,0} & x_{0,1} & \cdots & x_{0,m_2} \\ x_{1,0} & x_{1,1} & \cdots & x_{1,m_2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m_1,0} & x_{m_1,1} & \cdots & x_{m_1,m_2} \end{bmatrix} \tag{13}$$

Figure 19 illustrates the process of a discrete two-dimensional convolution using a small 2×2 kernel designed to detect vertical edges. The kernel is applied to a larger 5×5 dimension input image and computed according to equation 12. The resulting output image highlights the vertical edge with a darker color, while the horizontal edge appears less pronounced. It is important to note that zero-padding was not utilized in this example, leading to a reduction in dimensions for the output image S (4×4). If zero-padding, also named *same padding*, was applied, zeros would be added to the input image's borders in such a way that the output image S would have the same spatial dimensions as the input image X.

Figure 19 – 2D discrete convolution performed on input image X with dimensions of 5×5 using a filter kernel K with dimensions of 2×2, yielding an output image of size 4×4. The filter kernel is designed to detect and enhance vertical lines in the image.



Source: Author

The input of a two-dimensional **convolutional layer** l consists of a three-dimensional array containing $n_1^{(l)}$ two-dimensional feature maps with dimensions $n_2^{(l)} \times n_3^{(l)}$. For spectrograms or black and white images, the number of channels or feature maps is denoted as $n_1^{(1)} = 1$, while for a colored RGB image (The RGB color space is an additive color model, describing a color in three channels, representing the colors red, green and blue) the number of channels would be $n_1^{(1)} = 3$. Each convolutional layer comprises $m_1 \cdot n_1$ filter kernels $K_{i,j}$, each with dimensions $h_1^{(l)} \times h_2^{(l)}$, resulting in $m_1^{(l)}$ two-dimensional feature maps with dimensions $m_1^{(l)} \times m_2^{(l)}$ at the output. These filter kernels, denoted as $K_{i,j}$, are responsible for identifying specific patterns across the input data X_j.

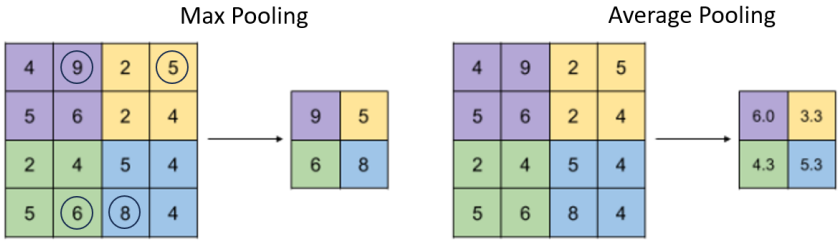
The output feature maps $Y_i^{(l)}$ for layer l are obtained by summing the discrete convolutions of the trainable filter kernels $K_{i,j}$ with the input feature maps X_j and a trainable bias term $b_i^{(l)}$. With $i = 1, 2, \ldots, m_1, j = 1, 2, \ldots, n_1$ and $\sigma^{(l)}$ as the non-linear transformation function, it compute as:

$$Y_i^{(l)} = \sigma^{(l)} \left(b_i^{(l)} + \sum_{j=1}^{n_1^{(l)}} K_{i,j}^{(l)} * X_j^{(l)} \right) \tag{14}$$

Subsampling the data as it progresses through the convolutional layers is crucial for enhancing the model's ability to learn larger and more intricate features. This process is achieved through **pooling layers**, such as the 2D pooling operation with a f_1 x f_2 sized **filter** and a **stride** matching the operation size. Each channel is independently processed in the pooling operation,

which outputs either the average value of the input in *average pooling* or the maximum value in *max pooling* as depicted in Figure 20. In other words, pooling layers don't learn any pattern in the network, rather reduce the resolution of the output feature maps from preceding convolutional layers ($Y_i^{(l-1)}$) by applying an filter window of size f_1 x f_2 with a stride of s_1 x s_2.

Figure 20 – Pooling examples on a two-dimensional input image X of size m_1 × m_2 = 4×4 with pool size f_1 × f_2 = 2×2 and stride s_1 × s_2 = f_1 × f_2 (no overlap).



Source: Author "adapted from" IndoML (2018)

As a remark, Long Short-Term Memory (LSTM) networks have recently emerged as a promising alternative to CNNs in the field of environmental sound recognition. Unlike CNNs which are primarily designed for image processing tasks, LSTMs are well-suited for sequential data analysis due to their ability to capture long-range dependencies and temporal dynamics in audio signals (BUBASHAIT; HEWAHI, 2021). By modeling the temporal relationships within sound sequences, LSTMs can effectively learn intricate patterns and nuances in acoustic data, making them a valuable tool to be further investigated for tasks in the field of ESR.

2.4.3 Conclusion

In the context of ESR, neural networks are able to offer distinct advantages. MLPs provide a strong foundation for understanding basic sound patterns and relationships using the interconnection of hidden layers and neurons, while CNN 1D excels in capturing temporal dependencies within sound data. On the other hand, CNN 2D is particularly effective in extracting spatial features from spectrogram representations of environmental sounds. By leveraging the unique strengths of each model, a more robust and accurate ESR algorithm can be developed for autonomous vehicles.

2.5 ELECTRONIC CONTROL UNIT

An Electronic Control Unit (ECU), is an embedded system found in automotive electronics that regulates one or more electrical systems or subsystems within a motor vehicle. To address the requirements of these systems \ subsystems, a clear trend in the automotive sector involves the growing prevalence of electronic components within vehicles that are vital for controlling the propulsion, chassis, body functions, in-vehicle networks, and driving information systems. Depending on the system, the ECU receives an specific acronym, for example: engine control module (ECM), powertrain control module (PCM), transmission control module (TCM), brake control module (BCM or EBCM), central control module (CCM), body control module (BCM), etc. Although colloquially referred to as the car's computer, it is important to note that these ECUs are distinct individual computers running System on Chip (SoC), rather than a single unit (APTIV, 2020).

2.5.1 Vehicle EE architecture

This section will present a brief overview about where the ESR algorithm could be deployed or embedded within the vehicle EE architecture considering the findings of Zhu et al. (2021) and it will not dwell in the intricacies and details of the architecture itself, rather will focus on the application deployment considering a regular passenger car and an autonomous vehicle.

Assuming a **centralized gateway architecture** with the ESR algorithm embedded on a specific ECU would have the advantage of improve functional safety given that the gateway converts different protocols and regulates network traffic, on the other hand, the number of ECUs have been increasing constantly which lead to more complexity in the gateway and ultimately, it may be harmful for security. A more updated architecture such a **domain-based** would integrate portions of the ECU, containing specific software components thus reducing material cost and weight in manufacturing processes. Additionally, the overhead in the gateway and bottleneck of multiples ECUs will be reduced, conversely, the potential downsize are delays that may be introduced between sensors and actuators. **Zone-base architecture** is currently a trend, and many automakers are developing their future products based on this concept. The main advantage is that the electronic components can be integrated based on their physical location in the vehicle, reducing cabling, especially for Ethernet, with the combination of communication and power. The disadvantage is that the architecture entails higher requirements for software platforms owing to the location cluster.

Shifting to autonomous vehicles, it's a "common sense" that Tesla is on the vanguard of hardware and software development, and according to Lindholm et al. (2008), the underlying structure of the Tesla architecture relies on a flexible processor array based on the GeForce 8800 Graphics Processing Unit (GPU), with 128 streaming-processor (SP) cores, organized into 16 streaming multiprocessors (SMs) that are further grouped into eight separate processing units referred to as texture/processor clusters (TPCs). Notably, this is just a small part of the so called HW4 infotainment system hacked by North (2009) on the Tesla Model. For obvious reasons, Tesla doesn't disclaim their infotainment technical specifications and therefore, the information that the HW4 infotainment system has 128 GB of storage capacity and 8 GB of Random Access Memory (RAM) is assumed as valid.

The Tesla architecture, developed by NVIDIA, is considered to be the pioneering platform for ubiquitous supercomputing. Its success is attributed to its compatibility with C programming language and the CUDA software development environment, which allows for the efficient deployment of computationally intensive parallel-computing and graphics applications. As transistors become more densely packed in the future, this architecture will be able to effortlessly scale processor parallelism, memory partitions, and overall performance (LINDHOLM et al., 2008). Based on this conclusion, it's plausible to assume that ESR algorithms could be deployed in autonomous vehicles using only embedded software in the current EE architecture. This approach would bring added value, but not extra costs on hardware. It may be necessary to considered additional sensors like external microphone arrays though.

2.5.2 High-end general-purpose embedded platform

In the realm of computing, specialized architectures and general-purpose platforms play distinct roles in catering to various computational needs. FPGA and TPU are examples of specialized architectures tailored for accelerating machine learning tasks, particularly those involving tensor operations. On the other hand, high-end general-purpose platforms like Raspberry Pi offer versatile computing capabilities at a more accessible price point. Considering the current platform utilized in the C-Bot, this study will focus on the high-end general-purpose platform Raspberry Pi 4, model B (RASPBERRY PI, 2023), however, this section will briefly explain the most common architectures for machine learning tasks, given that ECUs for the automotive industry, specially the ones designed for autonomous vehicles, can have a FPGA or TPU technology embedded in its hardware.

FPGA is a customizable integrated circuit that can be reconfigured after manufacturing, making it highly flexible for various applications, including accelerating specific algorithms and tasks. It's often used in specialized computing tasks, including machine learning acceleration, digital signal processing, and hardware emulation because they offer parallelism and low-latency processing, making them suitable for real-time applications where performance is critical. Latest technologies in FPGA include Xilinx Versal AI Core series, advancements in high-level synthesis tools for easier programming, integration with high-speed interfaces like PCIe Gen4 for faster data transfer, and the incorporation of hardened IP blocks for common functions, such as memory controllers and DSP blocks (INTEL, 2024).

TPUs Google's custom-built ASIC (Application-Specific Integrated Circuit) is designed specifically for accelerating machine learning workloads, more specifically, optimized for TensorFlow (TENSORFLOW, 2023), Google's open-source machine learning framework, but can also be used with other frameworks like PyTorch and JAX. They excel at both training and inference tasks and are known for their high performance and energy efficiency. Recent advancements in TPU technology include the development of second and third-generation TPUs, which feature higher computational capabilities, increased memory bandwidth, and improved support for mixed-precision computing (Coral Dev Board). Additionally, there's ongoing research into integrating TPUs with Google's Cloud AI infrastructure to provide scalable and efficient machine learning solutions for various industries (GOOGLE CLOUD, 2024).

Raspberry Pi is a series of small single-board computers developed by the Raspberry Pi Foundation, and although not as powerful or specialized as FPGA or TPU, Raspberry Pi offers a cost-effective solution for various computing tasks, from basic programming to IoT (Internet of Things) projects. The latest Raspberry Pi models feature significant improvements in processing power, memory, and connectivity options compared to earlier versions, for example, Raspberry Pi 5, introduced a quad-core ARM Cortex-A72 processor, up to 8GB of RAM, USB 3.0 ports, Gigabit Ethernet, and support for dual 4K displays. Recent developments in the Raspberry Pi ecosystem include the release of Raspberry Pi OS, a Debian-based operating system optimized for Raspberry Pi hardware, and the expansion of software and hardware peripherals compatible with the platform (RASPBERRY PI, 2023).

2.5.3 Conclusion

Upon initial examination, it appears more reasonable, regardless of whether using a domain-based or zone-based architecture, to leverage the capabilities of current sophisticated electronic systems available in regular passenger vehicles, like infotainment devices or head units, in order to integrate the ESR algorithm. These systems are already equipped with ample storage and computational power to effectively execute the algorithms, moreover, they often possess advanced embedded audio processing algorithms and beam forming features, further enhancing their suitability for this purpose.On the other hand, high-end general-purpose platforms like Raspberry Pi offer versatile computing capabilities at a more accessible price point, specially for prove-of-concepts or advanced prototypes like the C-Bot.

3 RELATED WORK

This chapter provides a comprehensive examination of the existing literature that establishes the relevant research pertaining to this study. The parameters utilized in this process are clearly defined and outlined in Appendix A.

The field of research associated with the recognition of acoustic scenes and environmental sounds has gained significant attention over the last decade, and currently holds immense practical value across diverse domains. Initiatives like challenges on Detection and Classification of Acoustic Scenes and Events (DCASE), an international competition that focuses on the development and evaluation of computational methods for the analysis of environmental sounds, aims to advance the state-of-the-art in acoustic scene classification and sound event detection serving as a platform for researchers and practitioners to benchmark their methods, share ideas, and contribute to the advancement of audio processing technologies (MESAROS et al., 2019). To portrait a few potential applications of this field of research: in urban planning, it may aid in traffic management and noise pollution control, optimizing city infrastructure, by combining inexpensive hardware components (such as Raspberry Pi devices) with deep learning algorithms for event identification, and custom planar antennas for communication between nodes within an ubiquitous sensor network framework (VIDAÑA-VILA et al., 2020). ESR ensures safety in different settings by monitoring noise levels and triggering emergency alerts as proposed by Sharma, Granmo, and Goodwin (2021) by developing a generic emergency detection system based on environmental sound using multiple audio feature extraction techniques and a deep CNN 2D for classification. The non-embedded model was evaluated using well established benchmark datasets, where emergency sound-related classes were combined into one category, making it a binary classification problem with outstanding accuracy results. The incorporation of ESR in smart homes can offer numerous advantages by enabling automation triggered by the identification of sounds that are commonly overlooked by the occupants of the house. These sounds may include tap water leakage, flush water leakage, television noise, shower running, radio playing, and many others which can lead to wastage of energy resources and potential accidents. By using microphone sensing units and machine learning classifiers, Pandya and Ghayvat (2021) proposed a sophisticated framework to accurately identify and interpret environmental sounds, ultimately taking actions to mitigate wastage or warn the house residents. A very simple commercial example of this application is the Alexa Guard. Further more, ESR finds applications in assistive technologies for the visually impaired (HUANG; CHHABRIA; JAIN, 2023), wildlife

monitoring for conservation (JEANTET; DUFOURQ, 2023), in healthcare, it may assists in patient monitoring (FUKUYAMA et al., 2022) and elderly care (SARAUBON; ANURUGSA; KONGSAKPAIBUL, 2018), and in agriculture for pest detection (BRANDING et al., 2023) and equipment monitoring (JEONG; AHN; PARK, 2022).

Considering the complexity of this field of research, a few authors have created a review on essential techniques associated with sound recognition, for example, Alías, Socoró, and Sevillano (2016) published an extensive article to review the most relevant audio feature extraction techniques for analyzing speech, music, and environmental sounds, while Abayomi-Alli et al. (2022), assessed the recent progress in research concerning the utilization of knowledge databases, small datasets, classification methods, classification metrics, audio features and data augmentation.

In the context of research and academic endeavors, several authors have investigated different approaches to overcome the scarcity of publicly available dataset using machine learning techniques for evaluation (SALAMON; JACOBY; BELLO, 2014), (BOUNTOURAKIS; VRYSIS; PAPANIKOLAOU, 2015), and (PICZAK, Karol J, 2015a), while others utilized machine learning as baseline for comparison with more complex architecture such as LSTM-CNN utilized by Pandya and Ghayvat (2021) or the hybrid ensemble of machine learning classifiers proposed by Bansal and Garg (2022). Achieving state-of-art classification metrics using small datasets is considered a typical problem, and a common method to overcome this situation involves data augmentation techniques. Unlike the ones well established for speech recognition such as warping the features, masking blocks of frequency channels, and masking blocks of time steps (PARK et al., 2019), ESR requires a different approach given the audio characteristics in frequency and spectrum. Time stretching, pitch shifting, dynamic range compression and background noise addition was utilized by Salamon and Bello (2017) with log-mel-spectrogram as input for a CNN 2D model. Mushtaq and Shun Feng Su (2020) considered time stretching, pitch shifting, white noise addition and silence trimming in his deep CNN 2D model using mel-spectrogram, MFCC and Mel-spectrogram (feature aggregation), Bountourakis et al. (2019) utilized only time stretching and pitch shifting but improved the features vector with additional statistical metrics to capture the temporal information of successive features, while Chu, Zhang, and Chiang (2023) proposed, a different type of augmentation using mel filters to input MFCC-spectrograms in the CNN 2D model. Drawing inspiration from the approach employed in the field of visual recognition, several authors have explored the concept of feature aggregation to construct images from various spectrograms, for example, Yu Su et al. (2020) investigated the outcome of using

different combinations of spectrograms from the audio features MFCC, gammatone, log-mel, chroma, spectral contrast and tonnetz. Luz et al. (2021) adopted a comparable strategy, but also added handcrafted features and compared the influence of feature selection in the process. Overall, the researches employment of diverse methods and approaches showcases a spectrum of creativity, ranging from light CNN 2D model with MFCC-spectrogram (SHREYAS et al., 2020), CNN 2D model utilizing multi-channel mel-spectrograms along with their deltas in both the time and frequency domains (TANG et al., 2018), deep learning models to separate music from environmental sound (ROTHMUND, 2018), stacked CNN 2D models using either log-mel-spectrograms, raw data or ensemble of both CNNs utilizing the Dempster-Shafer theory (LI et al., 2018), two-streams deep CNN 2D model using chroma, spectral contrast, tonnetz aggregated with MFCC-spectrogram in one stream and log-mel-spectrogram aggregated in the other (SU, Y. et al., 2019), two-streams deep CNN 2D model using aggregated MFCC-spectrogram and log-mel-spectrogram in one stream and raw data in the other (TRAN; TSAI, 2020), to highly complex architectures using self supervised learning as pretext to recognize the type of data augmentation that renders the best representative features as input for a deep CNN 2D model (TRIPATHI; MISHRA, 2021).

In the realm of embedded devices, researchers have also explored several applications for ESR. Notably, Abreha (2014), in his master's thesis, delved into the intricacies of this field by developing an environmental audio–based context recognition system using smartphones and machine learning classifiers (k-NN, SVM, and GMM) with MFCC, spectral centroid and spectral entropy as input. Expanding the scope of applications using microcontrollers, Nordby (2019) made significant contributions on city noise monitoring in his master's thesis by conducting noise classification on a sensor node utilizing a low-cost microcontroller and embedded a CNN 2D model that processed mel-spectrograms and delta-mel-spectrograms as input. Likewise, Vidaña-Vila et al. (2020), proposed a highly scalable low-cost distributed infrastructure that features a ubiquitous acoustic sensor network to monitor real-time urban sounds performed on microcontrollers. While high-end general-purpose platforms, FPGAs and TPUs have not been extensively investigated by researchers, a few authors have made significant contributions to this topic, starting with Silva et al. (2019) that evaluated machine learning classifiers (GNB, k-NN, SVM and decision trees) and an ANN model using perceptual and physical features summarized across time using statistics (mean, median, minimum, maximum, variance, skewness and kurtosis), comparing the metrics obtained between a notebook and a Raspberry Pi 3. Lhoest et al. (2021) followed the same approach, but proposed a method to improve these classical

machine learning techniques by combining multiple classifiers and create, what was named, a "mosaic" of classifiers (MosAIc), a method based on the one-vs-all approach to reduce the number of recognizable classes to a binary problem. Finally, Vandendriessche et al. (2021) developed a tool flow to evaluate a CNN 1D model using perceptual and physical features deployed in different platforms, namely, Raspberry Pi 4 model B, USB Coral TPU, Coral DevBoard, Zynq Z-7020 SoC and Zynq UltraScale+ XCZU7EV MPSoC. More recently, Lamrini, Chkouri, and Touhafi (2023) proposed a method utilizing pre-trained models from Yet another Audio Mobilenet (YAMNet), achieving state-of-art results in terms of accuracy and inference time. Their approach leveraged the knowledge and capabilities captured by a pre-trained model that has been trained on large datasets, and applied to resource-constrained embedded devices (Raspberry Pi 4 model B and USB Coral TPU), by freezing the last dense layer and replacing it with an ANN or CNN 1D.

Finally, within the context of the automotive industry, there are already strong indications that stakeholders are interested in implementing sound recognition technology in cars, effectively giving them "ears". A good example is the EU-funded I-SPOT project which aims to enhance the environmental awareness of smart cars by incorporating acoustic sensing technology. The project focuses on efficient sensor placement, signal processing technologies, and smart, low-power hardware aiming to achieve industrial and economical ambitions, including the training of young scientists, strengthening Europe's position in the smart car electronics business, and transferring academic knowledge to the industry (I-SPOT, 2020). Bosch is known in the automotive market as a cutting-edge supplier, and it comes as no surprise that they have been actively working on the development of a smart acoustic sensor specifically designed to detect sirens for autonomous vehicles. Their main objective is to enable cars to recognize the sound of sirens and respond appropriately, complying with legal requirements of many countries to give way to emergency vehicles. The sensor includes a microphone array and a microcontroller embedding artificial intelligence algorithms trained with more than 44.000 minutes of audio data. Once a siren is detected, the information is transmitted to the vehicle's onboard computer, allowing appropriate actions to be taken (BOSCH, 2024).

The papers associated with this research field in the industry are limited. One notable paper by Marchegiani and Newman (2022) presented a novel approach in urban environments for identifying and locating horns and sirens of emergency vehicles. This approach utilized an external microphone array installed in the vehicle roof panel to overcome the limitations of conventional filtering techniques. Spectrograms were employed as images and image processing techniques were utilized to enhance noise detection and background-foreground separation.

Additionally, the authors included acoustic event classification to differentiate alerting sounds and achieved accurate localization on the ground plane through the use of denoised signals. Although the study demonstrated exceptional performance in highly challenging scenarios with low signal-to-noise ratios, it lacked evaluation in real-time scenarios and embedding within any device. Sun et al. (2021) built upon the foundational framework established by Tran and Tsai (2020), employing their two-streams deep CNN 2D model augmented with a MLP that produced a binary outcome indicating the presence or absence of a siren. Simultaneously, a parallel CNN 1D model was implemented for both regression and classification purposes, coupled with two additional MLP outputs, one determining the direction angle and the other specifying the distance to uniquely establishes the position of the sound source. They achieved latency to produce all three results online in the range of 50ms, nevertheless, the model was again not embedded in any device. Shabtai and Tzirkel (2019) explored the use of microphones (internal and external) as sensors in autonomous vehicles, specifically for detecting the direction of emergency vehicle sirens. Using 4 omnidirectional MEMS microphones selected from a array of 32, they implemented a Multiple Signal Classification (MUSIC)-based algorithm with time smoothing techniques to improve the reliability of the estimated Direction-of-Arrival (DoA) values while for internal microphones, they utilized a transfer function projection for rough estimation of DoA. Their findings confirmed the possibility of determining the direction of an approaching emergency vehicle in both experiments, however, using internal microphones array proved to be less effective compared to the external array. Although none of the aforementioned papers conducted experiments on embedded devices, WAYMO, a former subsidiary of Alphabet Inc. (Google's parent company), has been focused on developing autonomous vehicle technology, encompassing both hardware and software. It appears to be a few steps ahead, as more recently they published a web article showing their prototype learning to recognize emergency vehicles in Arizona, using sound and light, fully embedded in the vehicle EE architecture (WAYMO, 2023).

A broader scope of ESR for autonomous vehicles was explored by Veeraraghavan and Ranga Charan (2020) by proposing a system that involved the utilization of a microphone to capture ambient sound, and processes it by a CNN 2D implemented on the Xilinx Zynq FPGA SoC. Additionally, the integration of a fuzzy logic Proportional-Integral (PI) controller enabled the reception of commands from the FPGA, as well as signals from manually-operated systems such as steering, braking, and acceleration. The resulting output commands generated by the controller were then transmitted to the ECU via the Controller Area Network (CAN) bus interface. It is noteworthy to mention that this framework was purely theoretical, implemented on Simulink,

and lacked empirical evidence regarding the accuracy of the CNN model. Nevertheless, the study highlighted the potential for utilizing an acoustic-based architecture in the development of a level 3 semi-autonomous vehicle. Likewise, Yin et al. (2023) published a conference paper with the ongoing results of the second stage of the I-SPOT project (I-SPOT, 2020) where a road acoustics simulator was designed, sound event detection and localization algorithms were developed, and a sensor array for vehicles was designed. A hybrid approach combining traditional signal processing and deep learning techniques was used for audio signal processing to ensure better interpretability of results in safety-critical situations like autonomous driving.

3.1 DISCUSSION

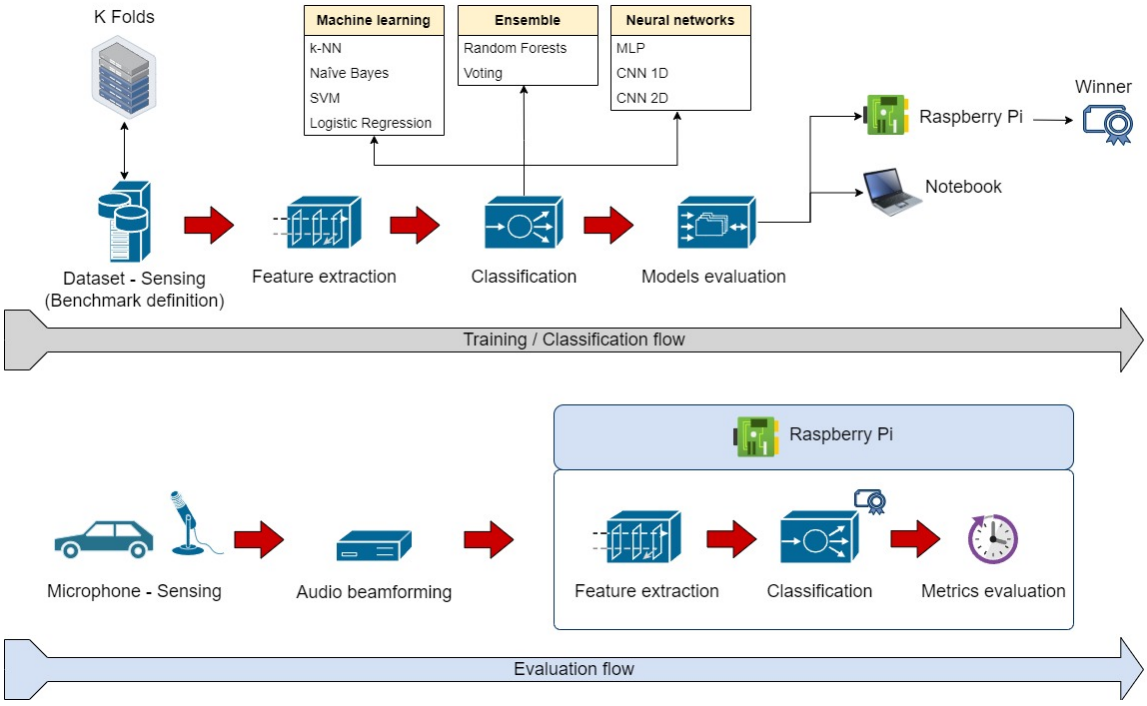
In general, despite the limited literature specifically addressing sound recognition in embedded systems for autonomous vehicles, significant contributions have been made to advance this field. Prominent research efforts have been directed towards the detection of emergency vehicle sounds and their precise localization, which is crucial in the context of the C-Bot. However, it is necessary to incorporate other sounds associated with road users which will limit the applicability of certain technologies such as filtering techniques. The utilization of neural networks, particularly those with a lightweight architecture and fast response time, continues to show promise in this domain. Additionally, it is worth noting that while many authors argued that internal microphone arrays are ineffective for detecting external sounds, there is evidence contradicting this claim, specifically in the case of sound source localization. This presents an opportunity to further explore the use of existing internal microphones in vehicles for various applications, including environmental sound recognition.

4 METHODS AND MATERIALS

The following methodology aims to compare classical machine learning techniques, ensemble methods, and neural networks classifiers in order to assess their responsiveness for integration within the framework of autonomous vehicles. Similar to other pattern classification tasks, audio classification consists of three essential elements: sensing, audio signal processing, and classification. **Sensing** involves the measurement of sound events or signals, while **audio signal processing** focuses on extracting characteristic features from the recorded sound, and **classification** belongs to the recognition of the contextual information associated with the sound event.

These elements are illustrated in Figure 21, namely **Training / Classification flow**, representing the initial stage of the experiment. The subsequent phase, titled as the **Evaluation flow**, also encompasses the same fundamental elements, however, it involves real-time simulation, employing a commonly available automotive microphone equipped with speech recognition capabilities, along with the selected classifier (winner) embedded in the Raspberry Pi module to emulate the condition of the vehicle's ECU, with the objective of verifying and confirming the findings obtained during the initial experiment.

Figure 21 – Illustration of the implemented methodology.



Source: Author

The organization of this chapter is structured as follows: section 4.1 describes the software and hardware utilized in the experiments. In section 4.2, an overview of well-known datasets used in the systematic review is presented, along with a tailored proposal specifically for environmental sounds in the context of autonomous vehicles. Section 4.3 introduces the concept of sample normalization within these datasets. The techniques for data augmentation are discussed in section 4.4. Section 4.5 provides a comprehensive explanation of the methods employed for feature extraction. The hyperparameters used in the training process of the classifiers are outlined in section 4.6, and finally, section 4.7 elucidates the evaluation procedures conducted on both the notebook and Raspberry Pi platform.

4.1 HARDWARE AND SOFTWARE

The implementation of the algorithms in this study utilized the Anaconda software platform (ANACONDA, 2023), an open-source tool for library management that facilitates the creation of multiple virtual environments. Anaconda is particularly geared towards scientific computing and finds applications in various fields such as data science, machine learning, predictive analysis, and general scientific projects. The algorithms were written using Python version 3.9, either in PyCharm or in Jupyter notebooks, along with the scikit-learn library for machine learning techniques and ensemble methods (SCIKIT-LEARN, 2023), as well as the Keras library for neural networks construction (KERAS, 2023). The Keras library is built on the Tensorflow framework (TENSORFLOW, 2023), serving as a high-level open-source Application Programming Interface (API) for neural networks, and it is specifically designed to allow for fast implementation of both simple and highly complex neural networks, offering a modular, intuitive, and extensible interface.

The training and testing of the models were performed on a notebook equipped with an Intel® Core™ i7-10850H CPU @2,70 GHz, 80 GB of RAM, and a Quadro T2000 graphics card with 4 GB of memory. The operational system in use was Windows 10 Enterprise.

Notably, the GPU resources of the Quadro T2000 were instrumental in training the neural networks. However, it is worth mentioning that the scikit-learn library's algorithms lack support for GPU acceleration, resulting in a slower training process overall. On the other hand, the Tensorflow framework was designed specifically to leverage the extensive capabilities offered by GPU resources, leading to training speeds approximately four times faster than when exclusively utilizing the CPU.

4.2 DATASET

The next subsections provide an overview of the prominent open-source datasets focusing on urban sound and ESR. These datasets are instrumental in this methodology for selecting suitable sound recognition systems for embedding purposes, while encompassing various categories that may not be exclusively urban-centric, they nonetheless encompass a sufficient repertoire of urban sounds that warrant their consideration for the purpose of this study, specially to establish a comparison baseline.

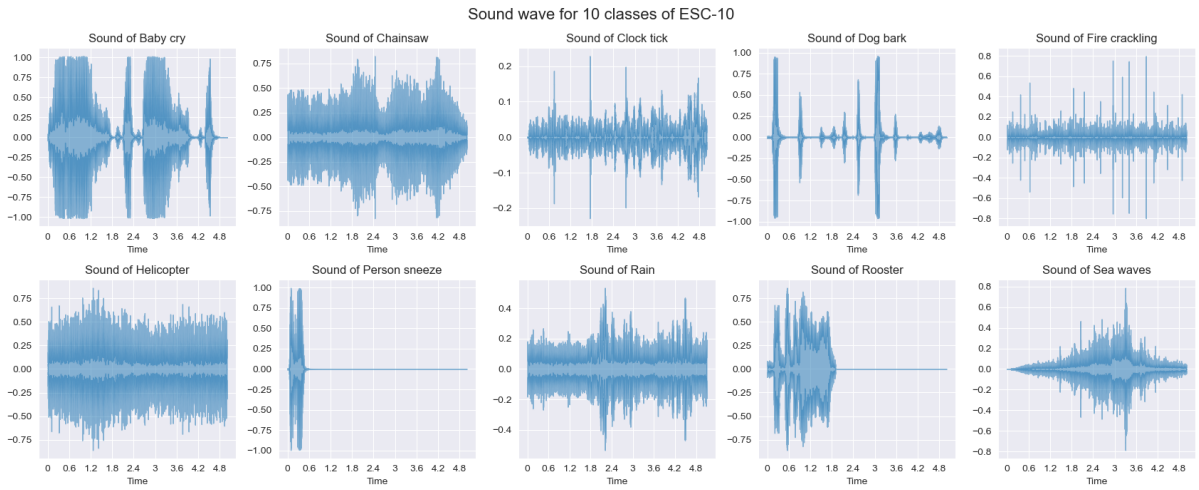
4.2.1 ESC-10

The compiled dataset in Karol J. Piczak (2015b) is divided into three sections: the primary labeled set encompasses 50 classes of diverse environmental sounds named ESC-50, a smaller proof-of-concept subset of 10 classes (ESC-10) was selected from the main dataset to serve as a simplified benchmark, and an additional dataset of unlabeled excerpts was included for unsupervised learning experiments (ESC-US). All datasets were constructed using sound clips sourced from publicly available recordings through the Freesound project (FONT; ROMA; SERRA, 2013). The selection of classes for the labeled part of the dataset was done arbitrarily, with the aim of maintaining balance among major sound event types while considering limitations in the availability of diverse source recordings and subjectively evaluating the usefulness and distinctiveness of each class.

The ESC-10 dataset is a labeled set of 400 environmental recordings with 10 classes, 40 clips per class, 5-second-long recordings reconverted to a single channel (monophonic), sampling rate of 44.100 Hz, using Vorbis/Ogg compression @ 192 kbit/s, namely (Figure 22):

a) 001 - Dog bark;
b) 002 - Rain;
c) 003 - Sea waves;
d) 004 - Baby cry;
e) 005 - Clock tick;
f) 006 - Person sneeze;
g) 007 - Helicopter;
h) 008 - Chainsaw;
i) 009 - Rooster;
j) 010 - Fire crackling.

Figure 22 – Wave form of randomly samples of each one of the classes within the dataset ESC-10.



Source: Author

The author propose the extraction of two distinct types of features, specifically, the ZCR and MFCC, from each audio clip. The ZCR presents itself as a straightforward yet informative characteristic, while the application of MFCC is widespread in the realm of speech processing and harmonic content analysis. The computation of MFCCs was undertaken using the Librosa package (MCFEE et al., 2023) with the default settings, giving rise to a frame length of 11,6 ms. Following the exclusion of the 0^{th} coefficient, the initial 12 MFCCs and the ZCR were aggregated for each clip via their respective mean and standard deviation across frames, resulting in a vector dimension of 26. These aggregated feature vectors were subsequently utilized as input for three distinct classifiers, namely, k-NN, Random Forest ensemble, and SVM with a linear kernel.

To assess the classifier performance, a 5-fold cross-validation regime was employed on the dataset ESC-10 during the learning phase, but unlikely other fold techniques, the author made it explicitly in his dataset metadata the fold split from 1 to 5. It was also confirmed in his code the for loop using these folds and therefore the same approach was carried out for the classifiers in this study. For the k-NN, the hyperparameter value for "k" was not specified in the article however, in the code was defined as 8. The Random Forest ensemble consisted of multiple decision trees, where each tree was trained on a randomly selected subset of features and samples from the dataset, but the number of trees in the ensemble was not mentioned. Again, analyzing the code made it possible to see the author's decision for 500 estimators. For SVM with a linear

kernel, no specific hyperparameters were mentioned either, but in the code, the regularization parameter C was set to 0,5 (PICZAK, Karol J., 2015b).

The results showed that for the ESC-10 dataset:

a) k-NN achieved an average classification accuracy ranging from 66,7%;

b) Random Forest ensemble achieved an average classification accuracy of 72,7%;

c) SVM had an average classification accuracy at 67%.

Considering the amount of classes involved in this study, the ECS-10 presents itself as better option to establish a baseline for the classifiers due its compact size and its original purpose as proof of concept.

4.2.2 BDLib2

The BDLib2 Environmental Sound Dataset created by Bountourakis, Vrysis, and Papanikolaou (2015) was designed to compare existing machine learning techniques for recognizing environmental sounds and determine the most effective one, focusing on analyzing discrete sound events rather than general acoustic environments, with the long-term goal of applying this knowledge to soundscapes recognition.

The dataset was constructed by identifying and extracting 10-second-long audio clips that represent distinct sound categories originally sourced from the BBC Complete Sound Effects Library (BBC, 2023) and Freesound project (FONT; ROMA; SERRA, 2013). The authors took utmost care during the clip selection process to ensure the absence of background noise and prevent any overlap between the sound categories.

All recordings in the dataset are uncompressed single channel files in WAV format, captured at a sampling rate of 44.100 Hz and analyzed with a precision of 16 bits, summing 180 labeled files, structured into 10 classes, namely (Figure 23:

a) Airplanes;

b) Alarms;

c) Applause;

d) Birds;

e) Dogs;

f) Motorcycles;

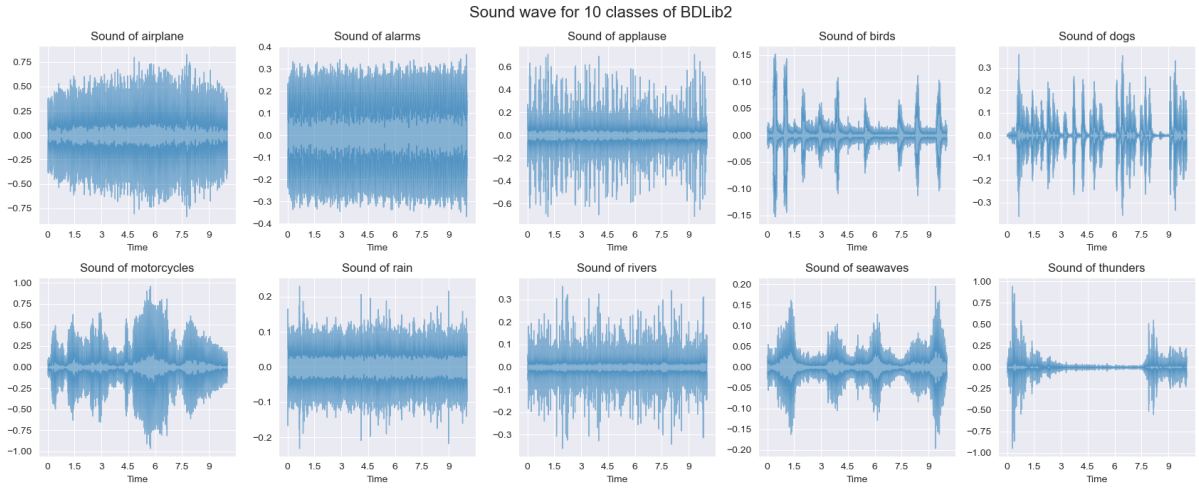
g) Rain;

h) Rivers;

i) Sea waves;

j) Thunders.

Figure 23 – Wave form of randomly samples of each one of the classes within the dataset BDLib2.



Source: Author

While the original dataset experimental results (BOUNTOURAKIS; VRYSIS; PAPANIKOLAOU, 2015) showed promising performance in classifying 7 different sound classes, it performed poorly in the recognition problem for 12 classes. Therefore, Bountourakis et al. (2019) changed the library structure into new classes deliberately chosen to encompass a comprehensive range of sounds encountered in both indoor and outdoor soundscapes, aligning with established recognition schemes. To maintain diversity and authenticity, each class was uniformly represented in the database with 18 audio files, showcasing significant variations that reflect real-life scenarios.

Considering the total duration of the collected files amounts to 1.800 s, which may be considered relatively small for the intended classification task, two commonly data augmentation techniques were applied to address the problem of limited data, namely time stretching with factors of 0,85 and 1,15, and pitch shifting by -4 and +4 semitones, carefully defined to ensure the semantic integrity of the transformed data.

Following augmentation, the total duration of the complete database increased to 8.820 s, which is comparable to the duration of similar reliable databases. It is worth noting that these augmentation techniques have been associated with improved model accuracy for environmental sound classification tasks (SALAMON; BELLO, 2017) and are also presented in the systematic review of Abayomi-Alli et al. (2022) for data augmentation and deep learning methods in sound classification.

In order to compare the frame-based method with temporal integration methods, several features showed in Table 3 were extracted using two different window configurations: long windows for the frame-based method and short windows for the temporal integration methods. The short window had a duration of approximately 46 ms with 50% overlapping over 2.048 samples, while the long window had a duration of approximately 1,48 s with 50% overlapping over 65,536 samples. By extracting aggregated features over 64 subsequent short windows with 50% overlapping, both methods achieved the same temporal resolution. For the sake of definition, texture window is commonly referred as an early integration technique when the process is carried out at the feature extraction level by consolidating short-time features across a larger frame (BOUNTOURAKIS et al., 2019).

Table 3 – Extracted audio features, including their respective dimensions (with a default dimension of 1 when not explicitly stated) and their acronym.

Feature	Symbol
Zero Crossing Rate	ZCR
Root Mean Square	RMS
Relative Difference Function	RDF
Spectral Centroid	CEN
Spectral Spread	SPR
Spectral Flux	FLU
Root Mean Square	RMS
Spectral Roll-Off	ROL
Spectral Skewness	SKEW
Spectral Kurtosis	KURT
Spectral Entropy	ENTR
Spectral Variability	VAR
Spectral Smoothness	SMO
Spectral Flatness Measure (24)	SFM
Spectral Crest Factor (24)	CSF
Brightness	BRI
Roughness	ROU
Irregularity	IRR
MFCC (13)	MFCC
Delta MFCC (13)	DMFCC
Linear Predictive Cepstral Coefficients (LPCC) (12)	LPCC

Source: Bountourakis et al. (2019), page 413

Beside the mean, standard deviation (std), skewness, and kurtosis used in most temporal integration methods, 4 additional metrics were proposed:

a) Mean Sequential Difference (MSD): This metric quantifies the amount of variation and frequency of changes in successive feature values within a texture window;
b) Mean Crossing Rate (MCR): Inspired by ZCR, MCR estimates the alternations of successive feature values with respect to their mean value within a texture window;
c) Flatness (FLA): Similar to Spectral Flatness, this metric calculates temporal flatness by dividing the geometric mean of feature values by their arithmetic mean within a texture window;
d) Crest Factor (CRF): CRF is calculated by dividing the maximum value by the mean value of feature values within a texture window, similar to how it is used for wave-forms.

To determine the most effective machine learning technique for the aforementioned dataset, 3 different methods for feature aggregation were proposed to generate feature vectors for sound classification, namely:

a) Standard Frame Based (SFb): a vector with dimension of 100 considering all features extracted in the Table 3, discarding the MFCC-0 that is fundamentally related to the signal's energy;
b) Standard Temporal Integration (STi): a vector with dimension of 96 build through the mean, standard deviation, skewness, and kurtosis of the following 12 baseline features: MFCCs, CEN, SPR, SKEW, KURT, ROL, ENTR, BRI, VAR, RMS, ZCR, and RDF;
c) Standard Temporal Integration (ETi): identical to STi, but including the 4 additional metrics previously described (MSD, MCR, FLA and CRF), adding 96 dimensions to the vector leading to a final dimension of 192.

Feature ranking algorithms were employed to select the most relevant features for each method, followed by the utilization of LR, ANN, and GLM classifiers for classification purposes. There were no mention of hyperparameters for LR and GLM, on the other hand, the ANN architecture consisted of 2 hidden layers, with the number of neurons in each hidden layer being equivalent to half the sum of the input and output dimensions, and a dropout layer was then incorporated. The remaining parameters followed a conventional configuration: the ReLU activation function was utilized for the intermediate layers, SoftMax function for the output layer, Categorical Cross-Entropy employed as the loss function, and Adam utilized as the optimizer. The learning rate was set at 0,01 with a dropout rate of 25%.

The training process involved splitting the sound files into a training set with 12 files per class and a testing set with 6 files per class, ensuring that the algorithms were tested on unseen data. To prevent overfitting and ensure unbiased results, a 3-fold cross-validation regime was employed during the learning phase and although a metadata was not available, the same principle used in the ESC-10 dataset was used since there are 3 distinct folds available after unzipping the downloaded files. The ETi method demonstrated higher accuracy than the STi method, and among the classifiers, ANN performed the best with 81,5% accuracy, followed by GLM with 80,4% and LR with 74,8%. Nevertheless, in light of the objective of conducting comparisons with the other datasets in this study, it is more appropriate to utilize the findings from the STi:

a) ANN achieved an average classification accuracy of 75,2%;
b) GLM following, with an average classification accuracy of 75,0%;
c) LR by last, with an average classification accuracy at 73.6%.

4.2.3 UrbanSound8K

The authors of this dataset developed a taxonomy based on a subset of a previous taxonomy specific to urban acoustic environments, focusing on low-level sound sources like "car horn" and "jackhammer" instead of broader categories. They also analyzed noise complaints filed through New York City's 311 service to identify frequently complained about sound categories relevant to urban sound research (SALAMON; JACOBY; BELLO, 2014).

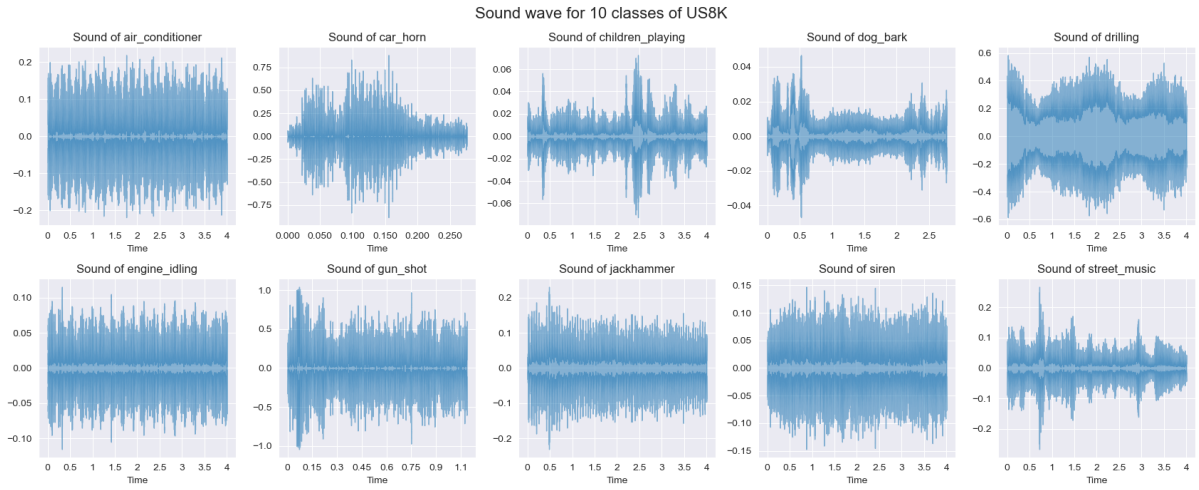
The UrbanSound dataset was created by downloading field recordings from the Freesound repository (FONT; ROMA; SERRA, 2013) and meticulously verifying each recording to ensure the presence of the sound of interest. After the filtering process, 1.302 recordings totaling over 27 hours of audio were obtained. In the next step, the recordings were labeled at the start and end times of every occurrence of a sound, and a salience description to indicate foreground or background perception was added. This resulted in 3.075 labeled occurrences amounting to 18,5 hours of labeled audio. Finally, for sound source identification research, an additional subset called US8K was created, encompassing short audio snippets segmented into 4 s slices using a sliding window approach with a 2 s overlap, totaling 8.732 labeled files.

Using the open-source audio editing software Audacity (TEAM AUDACITY, 2024), it was possible to identify that all recordings in the dataset are uncompressed multi channel files (stereo) in WAV format, captured at a sampling rate of 44.100 Hz and analyzed with a precision

of 32 bits, pre-arranged into ten folds (labeled fold1 to fold10) to facilitate the replication and comparison of the automatically generated classification results, namely (Figure 24):

a) air_conditioner;
b) car_horn;
c) children_playing;
d) dog_bark;
e) drilling;
f) enginge_idling;
g) gun_shot;
h) jackhammer;
i) siren;
j) street_music.

Figure 24 – Wave form of randomly samples of each one of the classes within the dataset US8K.



Source: Author

The sound classification process was performed in the US8K aiming to learn about the characteristics of the dataset rather than an optimal combination of feature/classifier parameters. Considering the reasonable amount of recording hours in the dataset, the authors extracted MFCC from the audio clips with no previous augmentation technique implemented. The features were extracted on a per-frame basis with a window size of 23,2 ms and 50% frame overlap using the first 25 MFCC coefficients from the initially 40 Mel bands computed. Summary statistics such as minimum, maximum, median, mean, variance, skewness, kurtosis along with mean and

variance of first and second derivative coefficients were computed to create a feature vector of dimensionality equal to 225 per audio clip.

In order to experiment with different classification algorithms and different audio clip lengths, a 10-fold cross-validation was utilized through a random allocation process to ensure that clips from the same original recording were not used for both training and testing, thus avoiding artificially high classification accuracy results. The machine learning classifier considered were: Decision Tree, k-NN with "k" set to 5 neighbors, Random Forest with 500 trees, SVM with a RBF kernel, and a majority vote classifier named ZeroR. The authors analyzed the classifiers performance considering a window of 1 s to 10 s for the audio clips, and most notably, it was noticed a uniform pattern in behavior across all classifiers, with performance maintaining stability from 10 s to 6 s, followed by a gradual decline. Yet, when focusing on the top-performing classifiers (SVM and Random Forest), there was no statistically significant distinction in performance between 6 s clips and 4 s clips, although the difference becomes significant when considering clips below 4 s. As the dataset was published with a 4 s window, the following results for each classifier were estimated below (values were not explicit, rather shown by a chart):

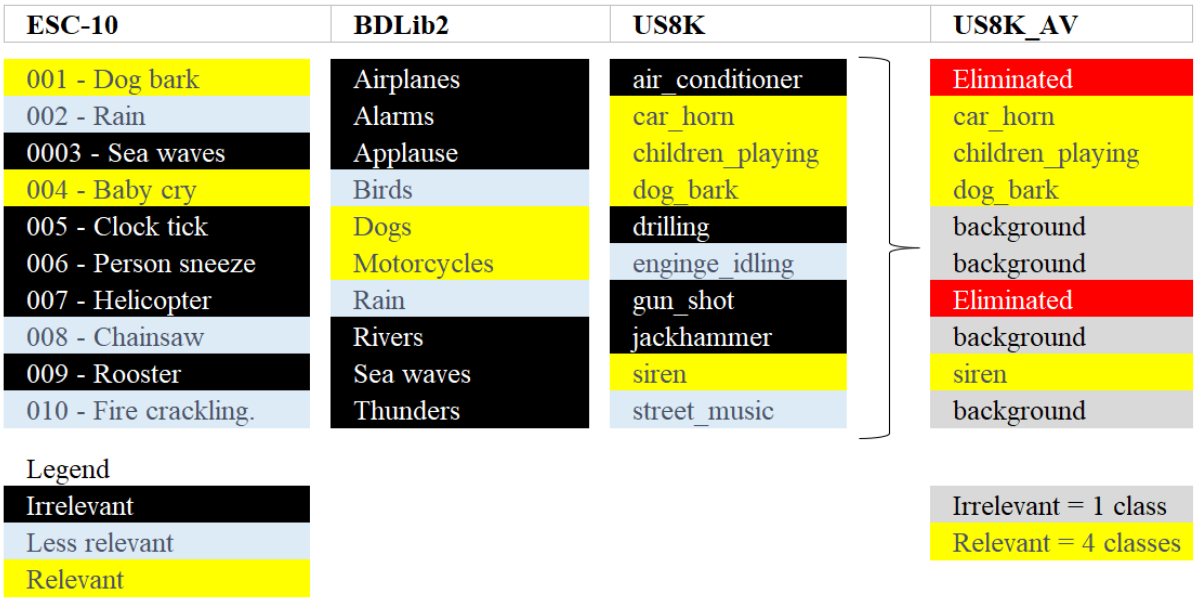
a) SVM was on top with an estimated classification accuracy of 69,0%;
b) Random Forest following, with an estimated classification accuracy of 66,0%;
c) k-NN achieved estimated classification accuracy of 56,0%;
d) Decision Tree achieved estimated classification accuracy of 48,0%;
e) Voting classifier by last, with an estimated classification accuracy of 10,0%.

4.2.4 Tailored dataset (US8K_AV)

The available datasets may not encompass the specific categories relevant to the context outlined in this study, and generating new datasets is a labor-intensive and time-consuming endeavor. To overcome this situation, the evaluation of the existing dataset was carried out subjectively, taking into account the relevance of classes that could potentially be integrated into the C-Bot during the third phase (Figure 25). A new dataset, denoted as US8K_AV, was constructed under specific conditions to optimize performance: irrelevant classes were eliminated, less relevant urban sounds were consolidated into new classes, and contextually relevant classes were preserved.

To maintain the same class distribution as the original dataset, the new class "background" was populated with 1.000 randomly selected samples from the original classes. Upon completing this process, the US8K_AV dataset comprised 4.358 files, distributed among 5 classes preserving the 10-fold split for cross-validation, with a total duration of 4,84 hours of recorded sounds.

Figure 25 – Subjective evaluation of the classes within the datasets ESC-10, BDLib2 and US8K related to autonomous vehicles and proposal for a tailored dataset based on the US8K classes.



Source: Author

4.3 NORMALIZATION

The concept of audio normalization entails the process of standardizing audio signals by harmonizing their amplitudes to a unified scale. Typically, this is achieved by manipulating the gain or volume of the audio while preserving its relative dynamics in order to establish a consistent level across all audio recordings, thereby minimizing the influence of factors such as disparate recording devices, microphone sensitivities, and environmental circumstances, bringing as consequences a consistency across different datasets, robustness to recording conditions and improved generalization (MUELLER, 2016).

There are multiple audio normalization techniques, and the following compilation outlines the most pertinent approaches according to Mueller (2021):

a) Peak amplitude normalization: it involves adjusting the amplitude so that the highest point in the audio signal reaches a specified target level, ensuring that no part of the signal is clipped;
b) RMS normalization: this method takes into account the overall energy distribution averaging its power signal and is less sensitive to short-term amplitude fluctuations;
c) Perceptual loudness normalization: essentially identical do the RMS normalization but based on perceived loudness which considers human auditory perception.
d) Frame-level normalization: in involves clipping the audio recording at a smaller time scale, such as individual frames or segments to address variations within the audio signal over time, thus capturing dynamic patterns more effectively;
e) Duration adjustment: while not strictly a normalization technique, adjusting the duration of audio segments to a common length is often performed in conjunction with amplitude normalization to ensures that the classifiers receive input of consistent duration in the training process.

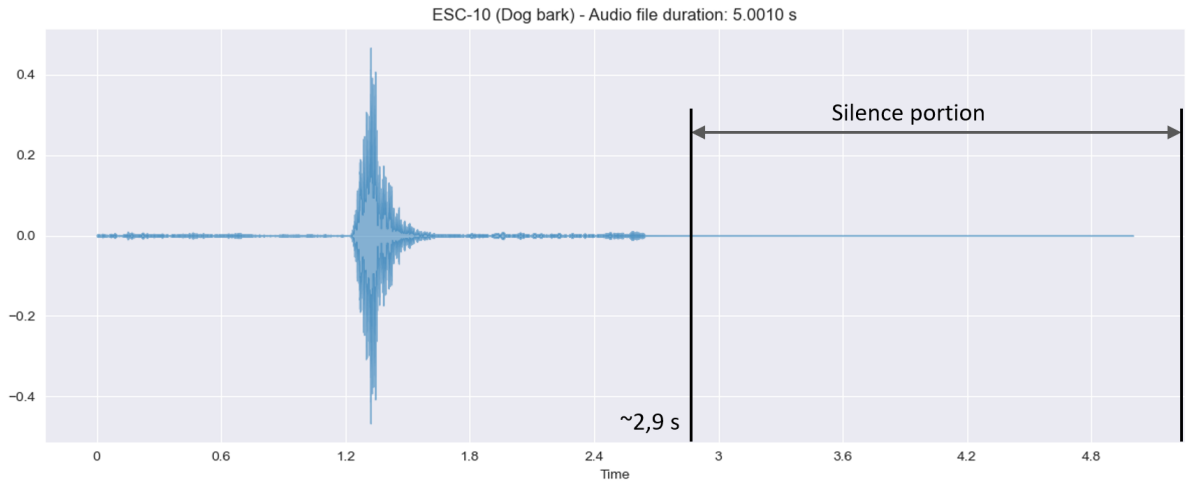
Fundamentally, the authors of the aforementioned datasets stated they were already normalized to reach an amplitude peak of 80 dB, nonetheless, such statement was also confirmed during the data exploration in the Jupyter notebooks. RMS and perceptual loudness normalization were not explicitly mentioned in the related work, and though conceptually relevant, they were not utilized in this study.

During the preliminary classifiers experiments, the concept of frame-level normalization was investigated following the procedures of Silva et al. (2019) and Lhoest et al. (2021). This involved dividing the audio signal into n short audio clips, each lasting 1 s and overlapping with a 50% margin (comprising 43 frames at a sampling rate of 22.050 Hz and a hop length of 512). These shorter audio clips were only temporarily stored in an array, until the relevant features were extracted from them. This technique was not considered in this study as an augmentation technique *per se*, rather was used in the section 4.5 as supporting tool for one of the feature extraction methods.

Several classes in the datasets, particularly ESC-10, ESC-50, and BDLib2, contain audio files where the original sound constitutes only 30% to 50% of the total duration, with the remaining portion being silence, as illustrated in Figure 26 using a random sample.

This type of audio data significantly reduces the accuracy of the classifiers, and therefore, the technique of duration adjustment was merged with trimming and implemented in this study. According to Mushtaq and Shun-Feng Su (2020), when applying the trim silence technique as a

Figure 26 – Wave form of an audio sample from the dataset ESC-10, class "001 - Dog bark", file name: 3-170015-A.ogg, illustrating most of the audio duration as silence or near-silence.

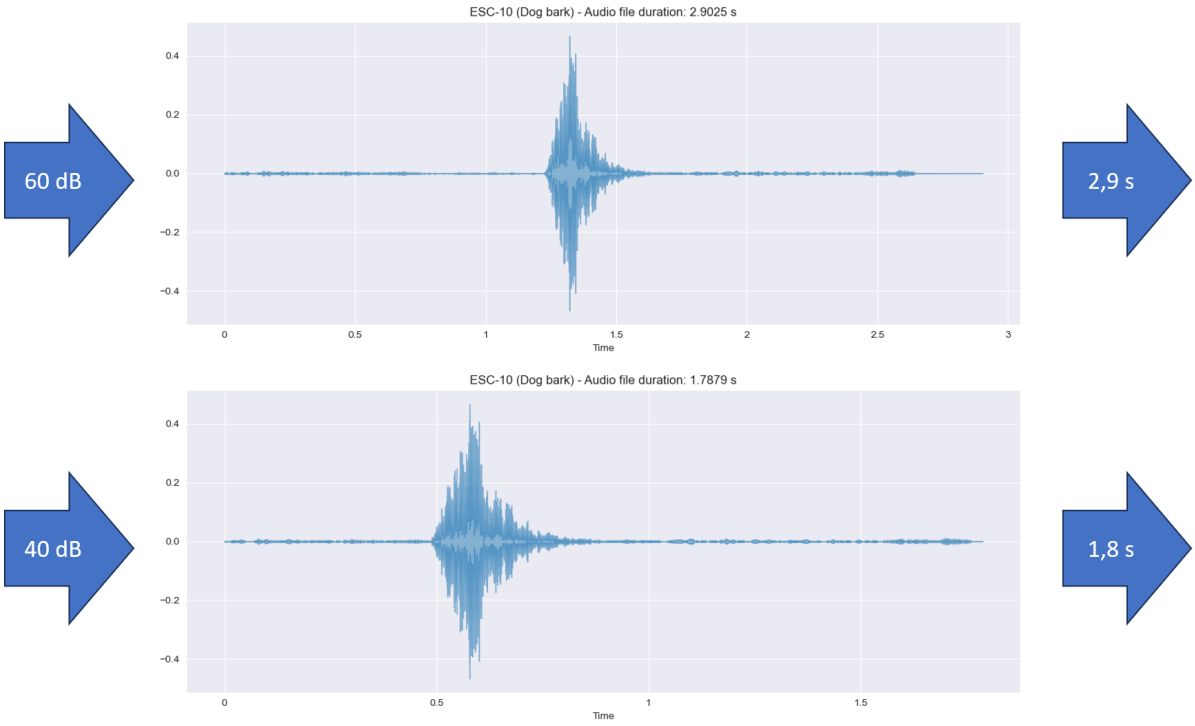


Source: Author

pre-processing method to the entire audio dataset, a common challenge arises in determining the precise threshold value in dB below the reference point (80 dB confirmed by peak amplitude normalization), which indicates silence. The improper selection of this threshold value has the potential to lead to the loss of crucial information contained within the audio data and although stated in the article that the threshold of 40 dB has been settled as the most effective silence trimming value for the used datasets, this value was not suitable when merged with the duration adjustment technique as shown in Figure 27 where approximately 1,1 s of the audio data was lost when compared with the threshold of this study.

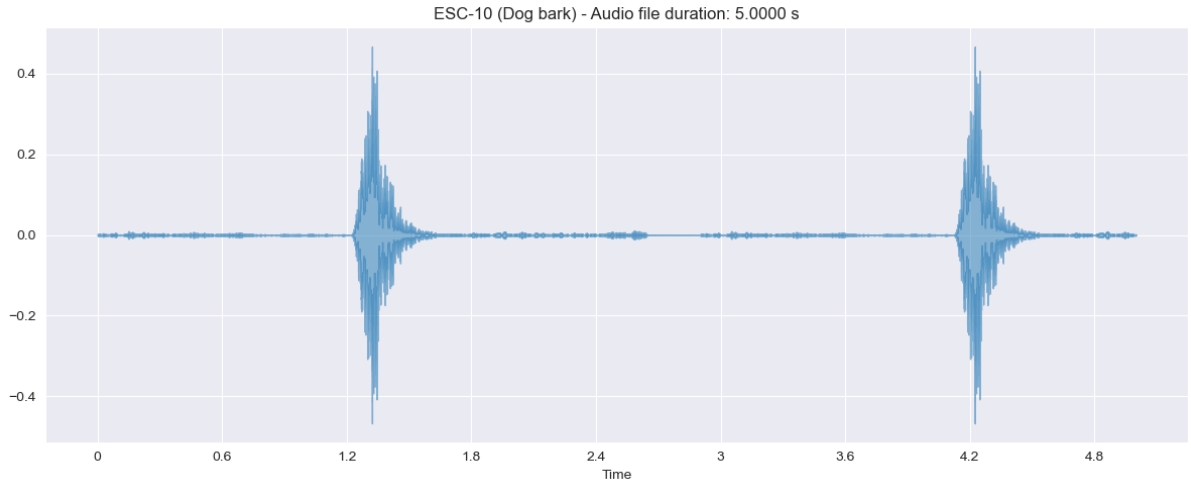
The value of 60 dB was settled as threshold after several investigations using the split function of the package Librosa (MCFEE et al., 2023), which took an audio signal as input and returned a NumPy array of intervals, where each interval is represented as a tuple of start and end times (in samples) of non-silent audio. The for loop continues, concatenating the non-silent audio frames until the target duration is exactly achieved for each dataset (ESC-10: 5 s, BDLib2: 10 s and UrbanSound 8k: 4s). The final audio wave form of the normalized audio data is shown in the Figure 28. Significantly, the normalization process primarily focused on the duration adjustment rather than trimming in the datasets, as trimming was observed to occur infrequently (after the process, approximately 20% of the audio data exhibited some form of trimming).

Figure 27 – Wave form of the audio file 3-170015-A.ogg trimmed to 40 dB and 60 dB resulting in approximately 1,1 s of data loss.



Source: Author

Figure 28 – Wave form of the audio file 3-170015-A.ogg trimmed to 60 dB and concatenated to reach the target duration of 5 s.



Source: Author

4.4 AUGMENTATION

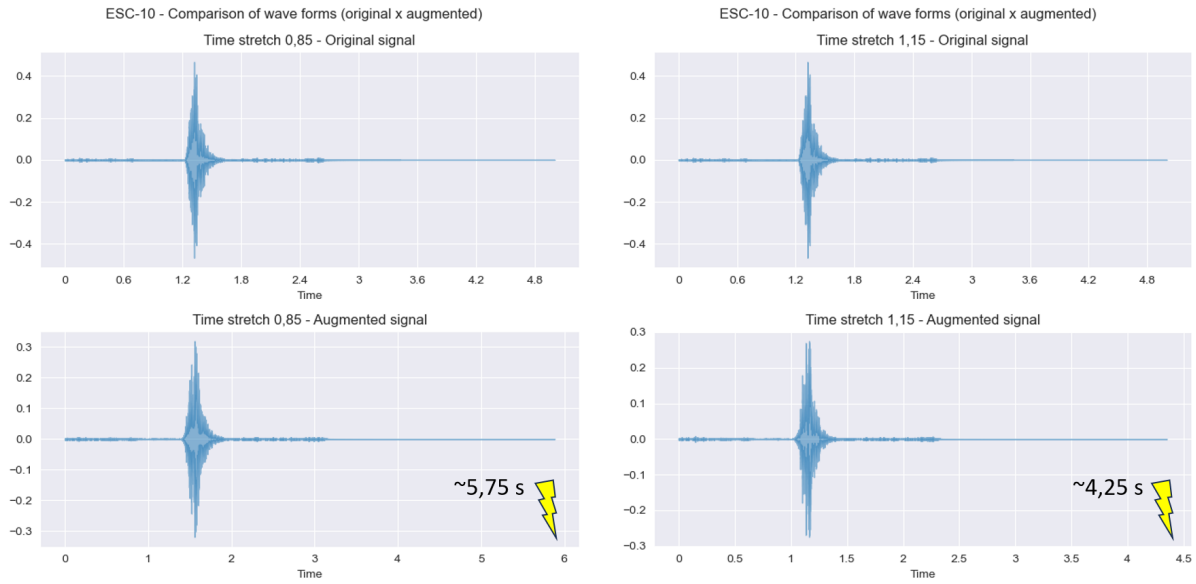
In general, augmentation techniques can help overcome the problem of limited labeled data, which is quite common in many application domains, and audio recognition is no exception. By artificially introducing variations in the audio data during training, audio augmentation expands the dataset and allows the models to learn more robust and discriminative features, thus improving their generalization capabilities, resulting in better equipped models to handle real-world scenarios where the sounds may exhibit variations in pitch, duration, and amplitude. Additionally, audio augmentation helps the models become less sensitive to background noise by training them on a wider range of acoustic conditions.

In the systematic review for deep learning methods and data augmentation in sound classification, Abayomi-Alli et al. (2022) ranked the technique of noise addition with the largest number of occurrences, accounting for 22 publications (39,2%). The subsequent highest number of occurrences was attributed to the time shift method, encompassing 15 publications (26,7%), following closely behind were the Generative Adversarial Network (GAN) based models and pitch shifting, each accounting for 12 publications (21,4%). Additionally, alternative methods, namely time stretching, mix-up, and background noise, summed 10 (17,8%), 9 (16,1%), and 8 (13,6%) publications, respectively.

Time stretching, pitch shifting, dynamic range compression and background noise were also used by Salamon and Bello (2017) in their deep convolutional neural network classifier. Mushtaq and Shun-Feng Su (2020) utilized time stretching, pitch shifting, white noise addition and silence trimming (that dwells between normalization and augmentation techniques) in their regularized deep convolutional neural network while Bountourakis et al. (2019) considered only time stretching and pitch shifting. Techniques commonly used in the speech recognition helm such as warping the features, masking blocks of frequency channels, and masking blocks of time steps (PARK et al., 2019) were not found explicitly in the related work and therefore were not considered in this study.

After experimenting the data augmentation within all the datasets, five techniques were selected and implemented in this study, namely **time stretches quickly** where the time of each audio clip of the dataset was stretched by a factor of 1,15 and **time stretch slowly**, acting in the opposite direction, the audio clip was slowed down by a factor of 0,85 (Figure 29).

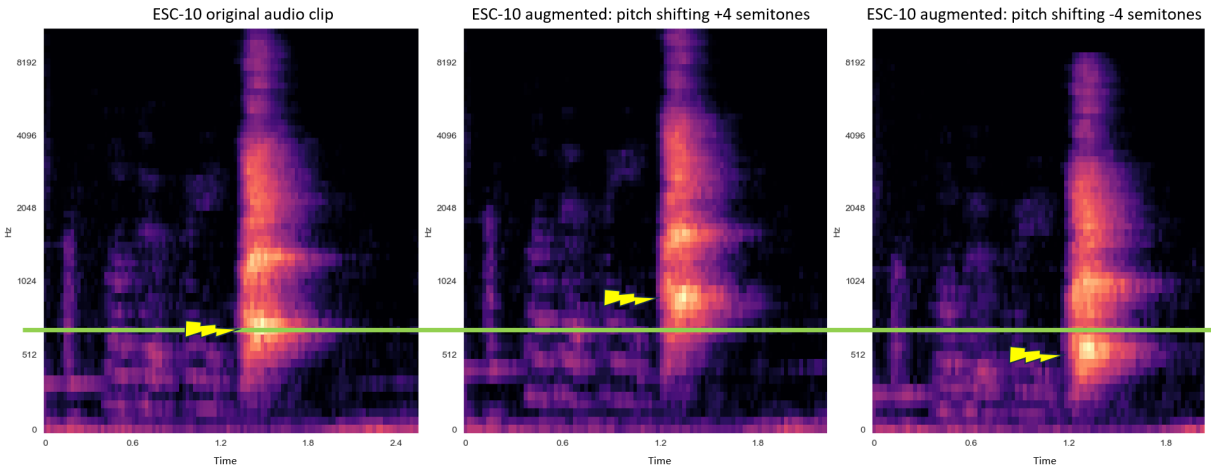
Figure 29 – Wave form of the audio file 3-170015-A.ogg augmented using the time stretch technique with factor 0,85 and 1,15.



Source: Author

Pitch shifting positively that increased the pitch of the audio clip in 4 semitones, and again, inversely, **pitch shifting negatively** that decreased the pitch of the audio clip in 4 semitones, keeping the total duration of the audio file the same as shown in the spectrograms of Figure 30.

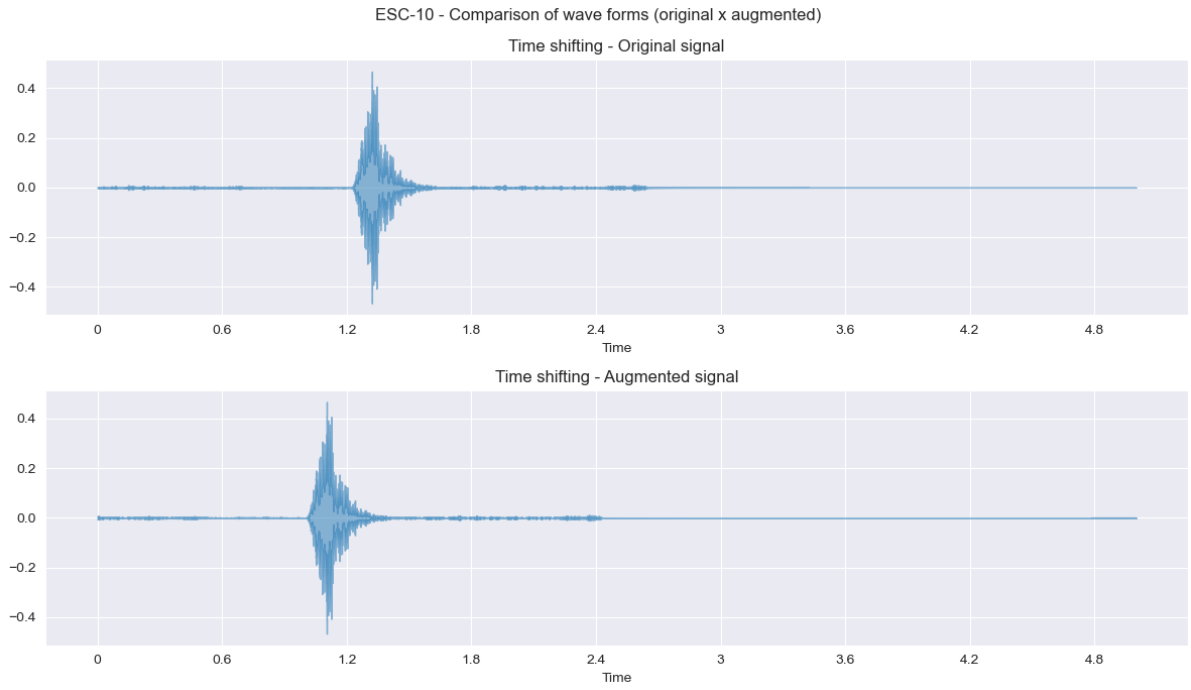
Figure 30 – Mel frequency spectrogram of the audio file 3-170015-A.ogg augmented using the pitch shifting technique with +4 and -4 semitones.



Source: Author

And finally, **time shifting randomly**, where the algorithm initialize a random integer (start_) within the range [-4800, 4800] using Numpy random.uniform. Depending on the value of start_, a time shift is applied to the audio clip. If start_ is negative, the audio is shifted to the right by start_ samples, and random noise is added at the beginning. If start_ is positive, the audio is shifted to the left by -start_ samples, and random noise is added at the end (Figure 31).

Figure 31 – Wave form of the audio file 3-170015-A.ogg augmented using the time shifting technique when start_ was initialized with +4800 and random noise was added in the end.



Source: Author

Table 4 – Characteristics of the datasets before and after the implementation of the augmentation process, including the total size of the compiled features file in format .PKL (Python library Pickle).

Dataset	**Samples** (units)	**Duration** (h)	**Features** (MB)	**Samples** (units)	**Duration** (h)	**Features** (MB)
	Before Augmentation			**After Augmentation**		
ESC-10	400	0,55	2,2	2.400	3,33	1.213
BDLib2	180	0,50	155,6	1.080	3,00	1.088
UrbanSound8K	8.732	8,75	3.034	52.392	52,20	21.215

Source: Author

After the implementation of the five augmentation techniques, the dataset characteristics were changed as described in Table 4 comparing the values before and after the process.

4.5 FEATURE EXTRACTION

There is a wide range of audio features that can be broadly categorized into perceptual and physical features based on their semantic interpretation as described in section 2.1.2. Perceptual features, such as pitch, loudness, rhythm, and timbre, approximate properties that are perceived by human listeners. Conversely, physical features describe audio signals in terms of mathematical, statistical, and physical properties, and are further classified as temporal features and spectral features, based on the domain of representation (LHOEST et al., 2021). Classical machine learning classifiers typically require the extraction of diverse features from the raw audio signal. While it is possible to input raw audio into classifiers based on CNN models, such classifiers tend to have numerous layers and a substantial number of parameters. This is because CNN models must learn to extract features from the raw audio in the early layers (CHU; ZHANG; CHIANG, 2023).

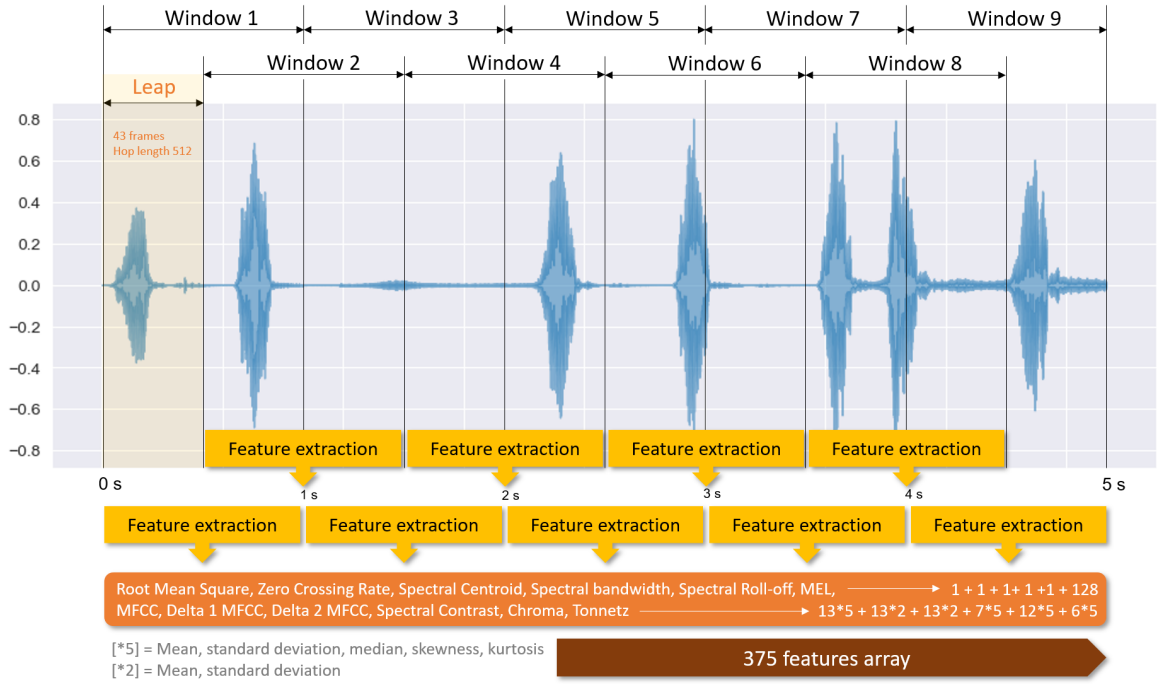
This section introduces the physical features and their extraction process utilized during the implementation of the experiments. The process of selecting features tailored to specific classification task is referred in the literature as feature engineering. All of the hereafter audio features are extracted using Librosa (MCFEE et al., 2023) due its variety of tools for feature extraction, easy-to-use implementation, supporting for ARM-based processors, and seamless integration with the Python library scikit-learn (SCIKIT-LEARN, 2023).

Once the audio features have been thoughtfully selected, these "handcrafted features" can be used in both machine learning and neural network classifiers instead of the raw audio, allowing the utilization of a smaller model that is more suitable for embedded devices.

For the purposes of comparison, the feature extraction process of the audio files in the datasets was divided into two methods. The first method (Figure 32) employed a sliding or leap window, following the procedures described by Silva et al. (2019) and Lhoest et al. (2021) in section 4.3, as a normalization technique (frame-level normalization). The choice of a sampling rate of 22.050 Hz was based on the work of Salamon and Bello (2017), and although Karol J. Piczak (2015b) and Bountourakis, Vrysis, and Papanikolaou (2015) utilized a sampling rate of 44.100 Hz, preliminary experiments showed no significant benefits of using a higher sampling rate, and the downturn of such choice was a larger compiled features file that ultimately consumed

more memory allocation and longer training time. The choice of a 1 s leap window was based on the fact that the sounds within the datasets, while non-stationary, did not exhibit abrupt changes over time. Larger leap window sizes, although slightly beneficial in terms of accuracy (SALAMON; BELLO, 2017), were not considered.

Figure 32 – Frame level normalization and feature extraction for a sample of the dataset ESC-10 (5 s duration, with 9 windows and 43 frames on each window).

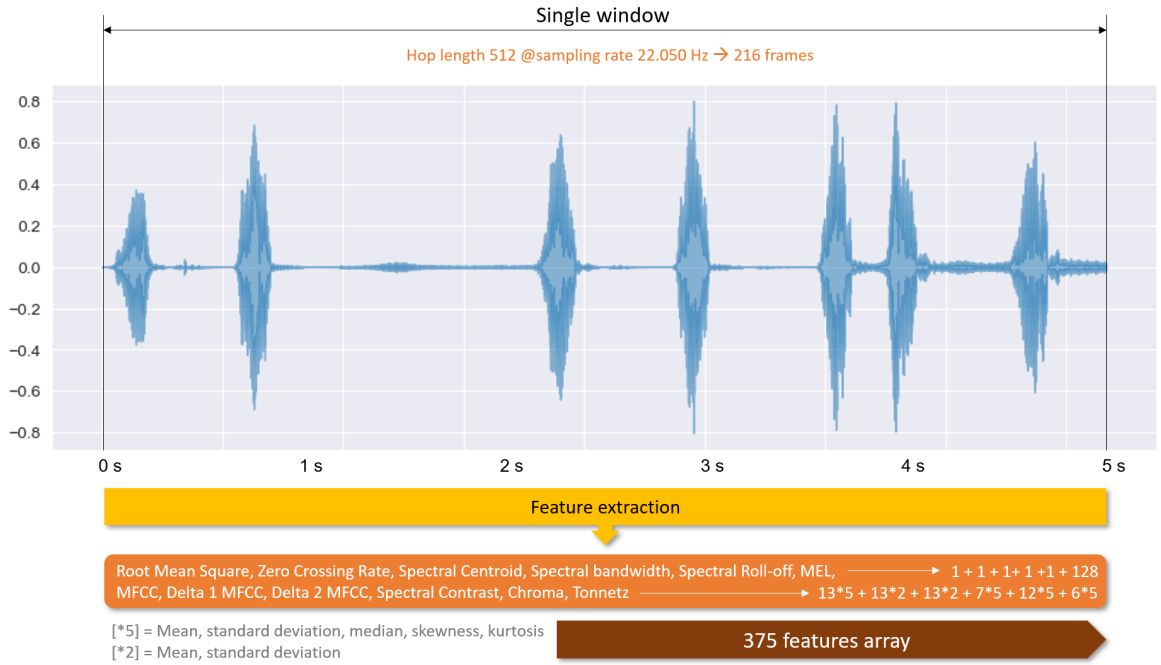


Source: Author

The second method, involved using the entire audio file as input for the feature extraction process (Figure 33), as proposed by the authors of the datasets ESC-10 (PICZAK, Karol J., 2015b), BDLib2 (BOUNTOURAKIS; VRYSIS; PAPANIKOLAOU, 2015), and UrbanSound8k (SALAMON; BELLO, 2017), but for normalization purposes, utilized the same sampling rate of 22.050 Hz of the first method. Vandendriessche et al. (2021) also utilized the same method in his CNN classifier embedded in FPGAs and TPUs.

The features selected were defined in the section 2.1.2, and the feature extraction process utilized a Hann window (BLACKMAN; TUKEY, 1958), with a hop length of 512, sampling rate of 22.050 Hz and frame size of 1.024 (around 46 ms) for RMS and ZCR to achieve a higher time resolution, and n_fft (number of frames for the STFT) of 2.048 for the remaining features to increase the frequency resolution. The number of Mel was 128 and MFCC was 13.

Figure 33 – Single window or complete audio feature extraction for a sample of the dataset ESC-10 (5 s duration, with 1 window and 216 frames).



Source: Author

In the first method, since the number of frames and hop length is fixed, resulting in a window of 1 s with 50% overlapping, the number of windows for the feature extraction on each dataset is calculated by $N_w = audioduration * 2 - 1$, leading to 9 windows for the ESC-10, 19 windows for the BDLib2 and 7 windows for the UrbanSound8K.

In both methods, the features with only one coefficient had their per-frame values calculated by mean, on the other hand, the features containing more than one coefficient, had their per-frame values for each coefficient summarized across time using statistics, namely: mean, median, std, skewness and kurtosis. The exceptions were Mel with only mean, Delta MFCC and Delta-Delta MFCC with mean and std calculated. The result was an array containing 375 features as shown in Table 5.

Finally, a separated algorithm to extract the MFCCs spectrograms for the CNN 2D was also implemented....to be continued

4.6 TRAINING THE CLASSIFIERS

The initial step on training the models involved finding the best hyperparameters using grid search and built-in cross-validation (SCIKIT-LEARN, 2023) to optimize model performance. Grid

Table 5 – Composition of the features array utilized in the machine learning, ANN, and CNN 1D classifiers after the feature extraction process.

Feature	**Coefficients**	**Mean**	**Mean, std**	**Mean, std, median, skewness, kurtosis**	$\sum$
RMS	1	1	—	—	1
ZCR	1	1	—	—	1
SC	1	1	—	—	1
SB	1	1	—	—	1
SRP	1	1	—	—	1
Mel	128	1	—	—	128
△ MFCC	13	—	2	—	26
△△ MFCC	13	—	2	—	26
MFCC	13	—	—	5	65
SCT	7	—	—	5	35
Chroma	12	—	—	5	60
Tonnetz	6	—	—	5	30
Total					**375**

Source: Author

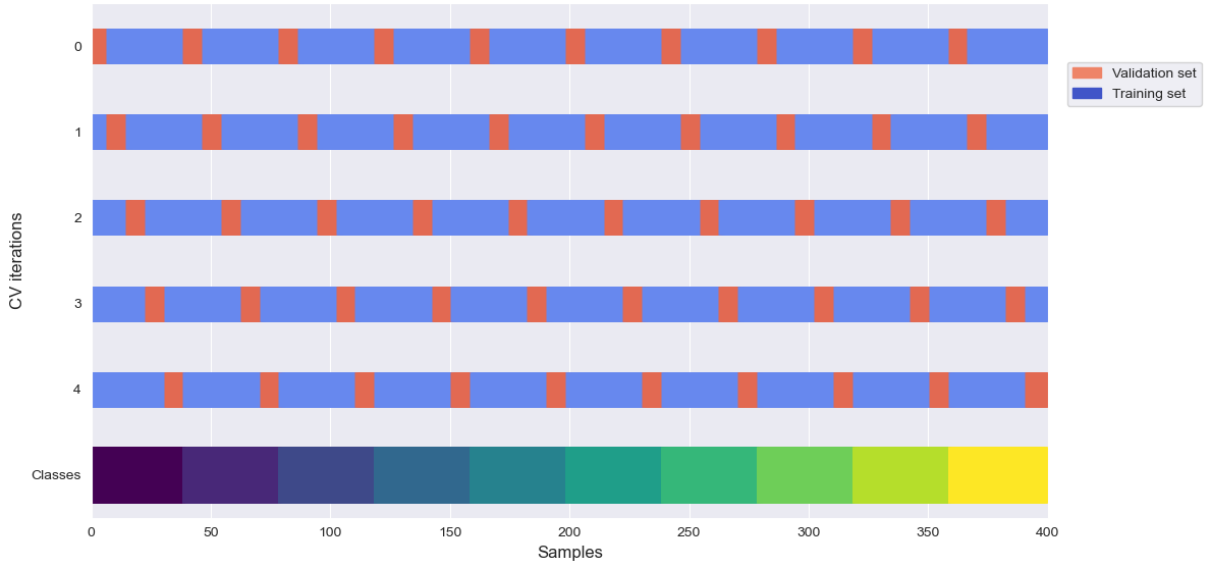
search systematically explored a predefined hyperparameter grid, testing various combinations to identify the set that yields the best performance while built-in cross-validation was employed to assess model generalization by dividing the dataset into multiple subsets. The use of built-in cross-validation helps mitigate overfitting by providing a more robust estimate of the model's performance across different data subsets, ensuring that the selected hyperparameters are likely to perform well on unseen data.

By default, the grid search splits the dataset into training and testing sets according to the parameter cv (number of cross-validations), and the model is trained on the former and evaluated on the latter, repeatedly, for each combination of hyperparameters in the grid. Depending on the number of hyperparameters to be searched, this process may take a long time of execution, specially in neural networks, and therefore, although recommended in the general literature to run the built-in cross-validation with 10 sets, it was established the number of 5 sets for all models, hence 80% for training and 20% for testing. Once the set of optimal hyperparameters was found for each model, the final step of the training started, validating the model on a separate, unseen validation set, using the method of k-fold cross-validation, to assess its true generalization performance.

Each of the aforementioned datasets was initially intended to undergo validation using k-fold cross-validation methods, with each dataset considering a distinct value for k: ESC-10

employs 5 folds, BDLib2 utilizes 3 folds, and UrbanSound8K employs 10 folds. Figure 34 illustrates the process of the k-fold cross-validation using the ESC-10 dataset as example, demonstrating that each k-fold (0 to 4) assigned in the dataset metadata was properly separated within the 10 classes. In order to compare the classifier's results accordingly, the number of k-folds for each dataset was respected.

Figure 34 – Process of k-fold cross-validation using the metadata of the dataset ESC-10 with 10 classes and 5 folds.



Source: Author

In the end of this process, the classifier's models using machine learning techniques and ensemble methods were configured with their standard parameters according to the scikit-learn library, except for the following hyperparameters:

a) GNB: $priors$ = none (Prior probabilities of the classes);

b) SVM: $kernel$ = 'linear', C = 0,50 (Regularization parameter);

c) LR: $solver$ = 'saga', C = 0,50 (Regularization parameter), max_iter = 500 (Maximum number of iterations taken for the solver to converge);

d) k-NN: $n\_neighbors$ = 3, $metric$ = 'minkowski', p = 2 (Power parameter for the Minkowski metric, 2 means euclidean distance), $leaf\_size$ = 20 (Leaf size passed to BallTree or KDTree algorythm to calculate the nearest neighbors considered that 'auto' is the standard);

e) Random Forest: $criterion$ = 'gini', $n\_estimators$ = 500 (The number of trees in the forest), $bootstrap$ = True (Whether bootstrap samples are used when building trees);

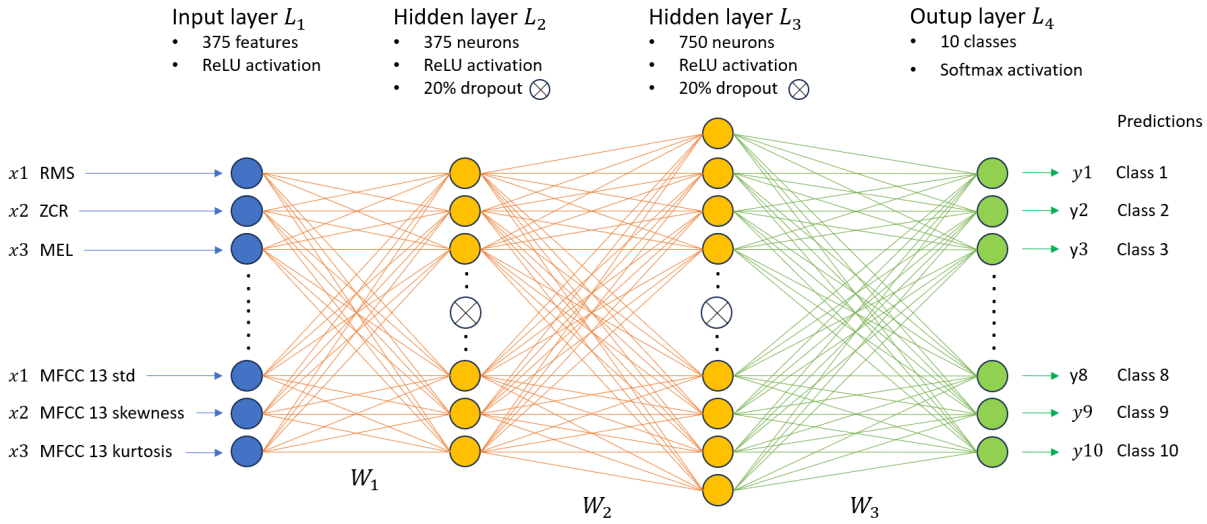
f) Voting: $voting$ = 'soft' (Predicts the class label based on the argmax of the sums of the predicted probabilities).

The ANN was originally created using Kolmogorov's theorem where given any continuous function $\phi : I^n \longrightarrow \mathbf{R}^m, \phi(\mathbf{x}) = \mathbf{y}$, where I is the closed unit interval $[0,1]$ representing the classes, and therefore I^n is the n dimensional unit cube, ϕ can be implemented exactly by a 3 layer neural network having n processing elements in the first input layer (x), followed by $(2n + 1)$ processing elements (neurons) in the hidden layer, and m processing elements in the output layer (y) (HECHT-NIELSEN, 1987). The network was built using Keras sequential model to define a MLP with the first dense layer receiving as input an array with 375 features (parameterized with Principal Component Analysis (PCA) when requested) and ReLU activation function. The second and third dense layers were hidden, with 375 neurons and 750 neurons respectively, both using ReLU activation function. To improve the model's accuracy proposed by Hecht-Nielsen, one additional hidden dense layer was added based on the grid search and built-in cross-validation results. A dropout layer with 20% rate was also added after each hidden layer to improve generalization, prevent overfitting, and make the model more robust by introducing randomness during training.

The final dense layer with 10 neurons (one for each class) is utilized for the classification task by using softmax activation function to assign probabilities to each class in the output layer. The hyperparameters used in the grid search to achieve this model configuration (Figure 35 and Figure 36) were $loss$ = 'categorical_crossentropy', and the Adan optimizer with $learning\_rate$ = 0,0001, $beta\_1$ = 0,5 (The exponential decay rate for the 1st moment estimates), $beta\_2$ = 0,999 (The exponential decay rate for the 2nd moment estimates), $epsilon$ = 1e-07 (A small constant for numerical stability), $amsgrad$ = True (Whether to apply AMSGrad variant (REDDI; KALE; KUMAR, 2018) of this algorithm).

A second round of grid search was performed to find the best parameters for the batch size and epochs, implemented together with a early stop process. Ultimately, the values considered were: $batch\_size = 20$, $epochs$ = 350 and $EarlyStopping$ with $monitor$ = 'val_accuracy', min_delta = 0,0001, $patience$ = 150, and $restore\_best\_weights$ = True. As illustrated in Figure 37, the ANN model achieved a stable and rather well generalized behavior during the built-in cross-validation process.

Figure 35 – ANN architecture utilized in the experiments.



Source: Author

Figure 36 – ANN model summary from Jupyter notebook.

```
model_ANN = build_ANN_model('ANN_1', neurons = n_dim)
model_ANN.summary()
```

```
Model: "ANN_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Input (Dense)                (None, 375)               141000
_________________________________________________________________
Hiden_1 (Dense)              (None, 375)               141000
_________________________________________________________________
Dropout_1 (Dropout)          (None, 375)               0
_________________________________________________________________
Hiden_2 (Dense)              (None, 750)               282000
_________________________________________________________________
Dropout_2 (Dropout)          (None, 750)               0
_________________________________________________________________
Output (Dense)               (None, 10)                7510
=================================================================
Total params: 571,510
Trainable params: 571,510
Non-trainable params: 0
_________________________________________________________________
```

Source: Author

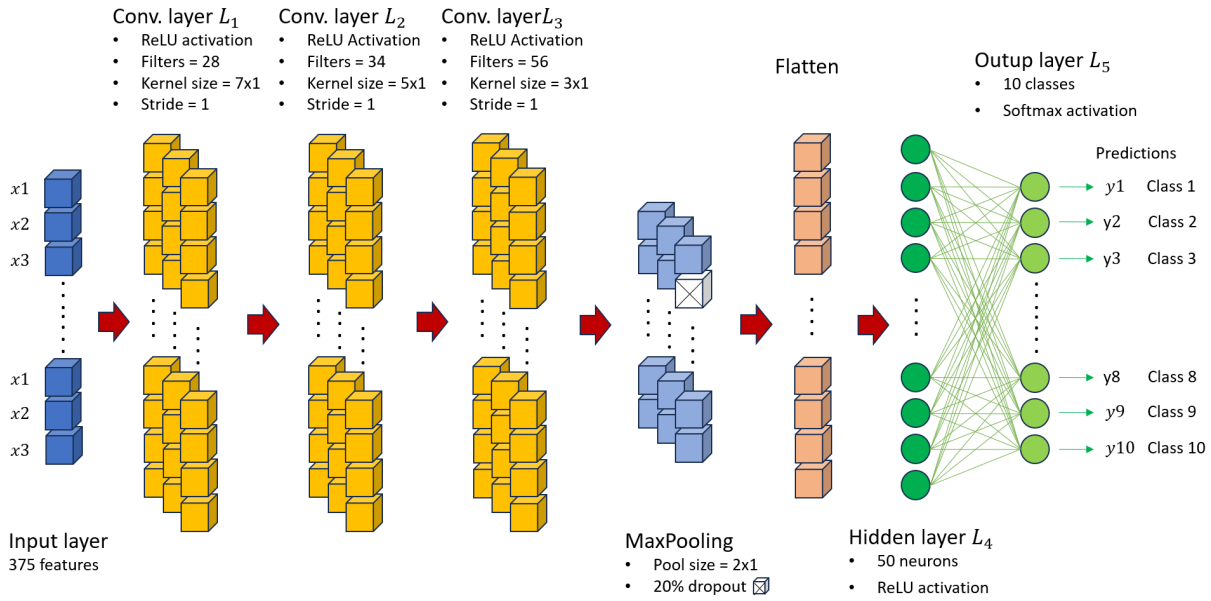
The architecture of the CNN 1D model illustrated in Figure 38 and summarized by the Jupyter notebook in Figure 39 consisted of several layers, compiled with a categorical cross-entropy loss function, adamax optimizer, and accuracy as the evaluation metric:

a) Conv1D_1: The first convolutional layer has 28 filters with a kernel size of 7 x 1, and it uses the ReLU activation function, receiving as input an array with 375 features;

Figure 37 – Loss and accuracy graphs of the implemented ANN model

Source: Author

Figure 38 – CNN 1D architecture utilized in the experiments.



Source: Author

b) Conv1D_2: This is the second convolutional layer with 34 filters, and kernel size of 5 x 1. It also uses the ReLU activation function and applies L2 regularization to the kernel with 0,001 and bias with 0,01. $Padding$ is set to 'same' with $stride$ as default (1) to ensure the output size matches the input size;

c) Conv1D_3: This is the last convolutional layer. It has 56 filters with a kernel size of 3 x 1. The same parameters of the second convolutional layer are used for the activation function and L2 regularization to the kernel and bias;

d) MaxPooling1D_3: This layer performs max pooling with a pool size of 2 x 1, reducing the dimensionality of the data by taking the maximum value within each window of size 2 x 1;

e) Dropout_1: To reduce overfitting, this layer randomly sets 20% of the input units to 0 at each update during training;

f) Flatten_5: To prepare the data for the next step, this layer reshapes the output of the previous layer into a 1-dimensional vector;

g) Dense: This layer is a fully connected to the previous layer with 50 neurons;

h) Output: Finally, the classification task is performed in this layer. It has a number of neurons equal to the number of output classes (10), and it uses the softmax activation function to output probabilities for each class.

Figure 39 – CNN 1D model summary from Jupyter notebook.

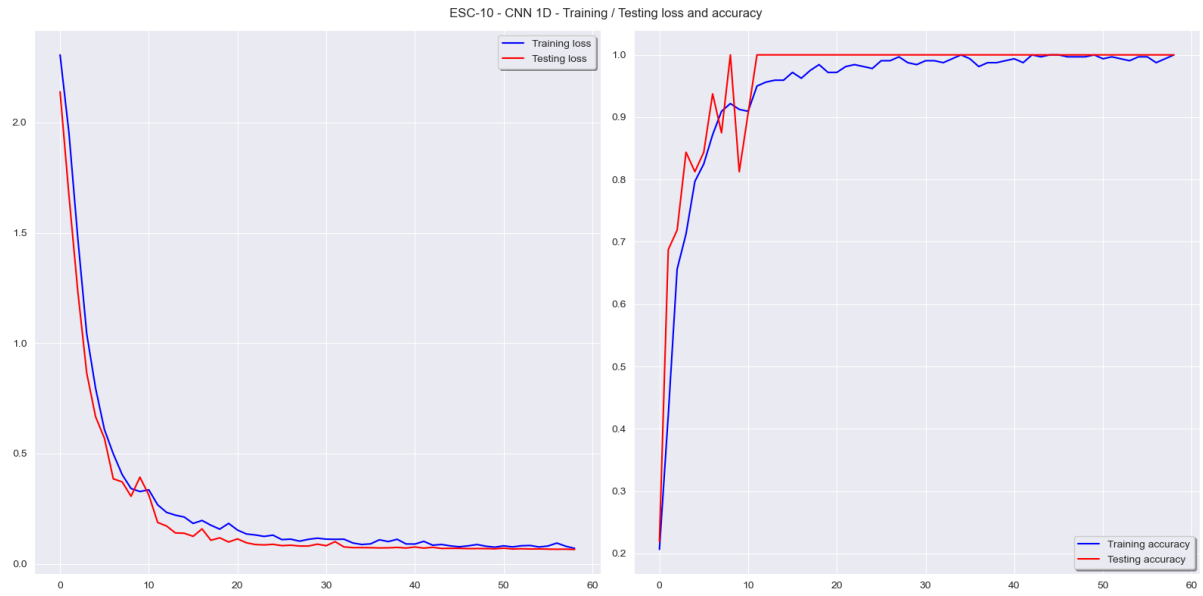
```
model_CNN_1D = build_CNN_1D_model('CNN_1D', neurons = n_dim)
model_CNN_1D.summary()
```

```
Model: "CNN_1D"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Conv1D_1 (Conv1D)            (None, 369, 28)           224
_________________________________________________________________
Conv1D_2 (Conv1D)            (None, 369, 34)           4794
_________________________________________________________________
Conv1D_3 (Conv1D)            (None, 369, 56)           5768
_________________________________________________________________
MaxPool1D_3 (MaxPooling1D)   (None, 184, 56)           0
_________________________________________________________________
Dropout_1 (Dropout)          (None, 184, 56)           0
_________________________________________________________________
flatten_5 (Flatten)          (None, 10304)             0
_________________________________________________________________
Dense (Dense)                (None, 50)                515250
_________________________________________________________________
Output (Dense)               (None, 10)                510
=================================================================
Total params: 526,546
Trainable params: 526,546
Non-trainable params: 0
_________________________________________________________________
```

Source: Author

In the same way as the ANN, a grid search was performed to find the batch size and epochs that yield the highest accuracy, implemented together with a early stop process. In the final stages, the values considered were: $batch\_size = 20$, $epochs$ = 150 and $EarlyStopping$ with $monitor$ = 'val_accuracy', min_delta = 0,0001, $patience$ = 50, and $restore\_best\_weights$ = True. Figure 40 depicts a robust model that achieved the highest accuracy with just a few epochs. Early stop was triggered during the built-in cross-validation between 50 to 70 epochs.

Figure 40 – Loss and accuracy graphs of the implemented CNN 1D model



Source: Author

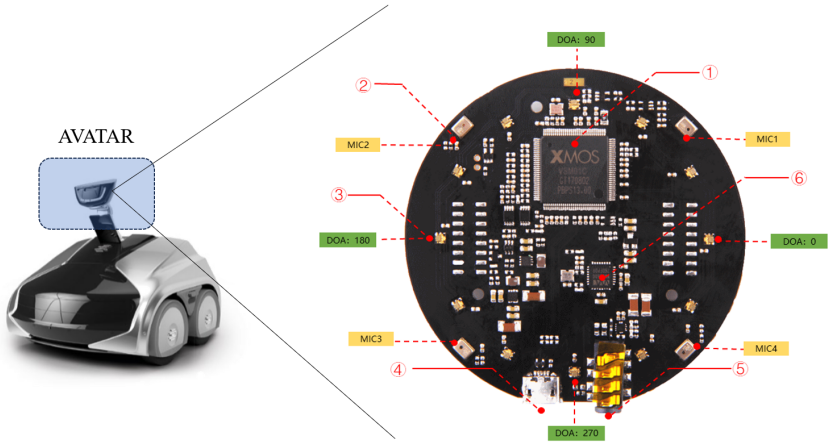
4.7 EVALUATION

Once the training is completed, the evaluation phase started, initially stationary as illustrated in the methodology diagram (Training / Classification flow), and lastly mobile (Evaluation flow), in the practical scenario of a regular passenger vehicle, considering factors like ambient noise, changing environmental conditions, and the need for instantaneous responses.

The C-Bots utilized in the living laboratory are equipped with ReSpeaker Mic Array v2.0 (SEEED STUDIO, 2021) installed in the avatar head (Figure 41), connected to the AVATAR-ECU which is used for speech recognition among other functions. The ReSpeaker contains 4 high performance digital microphones (ST MP34DT01TR-M), with sensitivity of -26 dBFS (Omnidirectional), acoustic overload point of 120 dBSPL, maximum sampling rate of 16 kHz, and speech recognition algorithm on-chip, including support to far-field voice capture. The

AVATAR-ECU is a Raspberry Pi connected to the vehicle EE architecture via CAN and Ethernet, and according to the C-Bot system engineer, the ESR algorithm could be implemented on this module or in another high-performance computing module of the vehicle, and its output information published via ROS2.

Figure 41 – Illustration of the ReSpeaker Mic Array v2.0 installed in the AVATAR of the C-Bot.



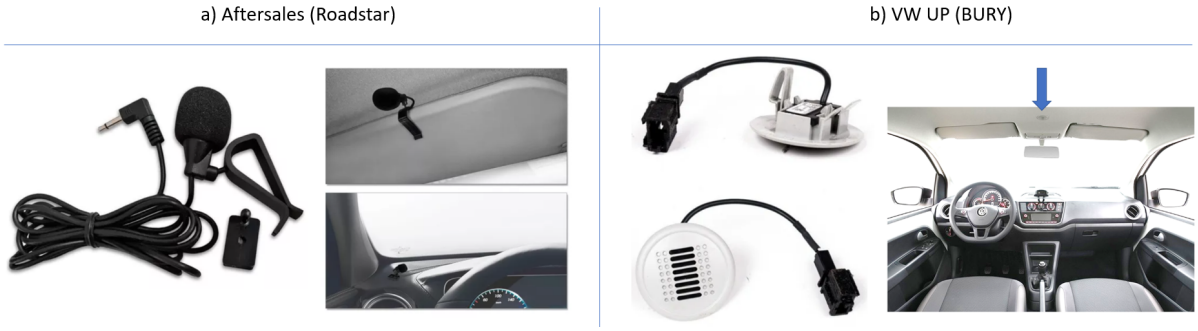
Source: Author, "adapted from" Seeed Studio (2021)

As described in subsection 1.2.1, the C-Bot hardware is not available in Brazil for testing therefore, the best-performing model was deployed in a high-end general-purpose platform identical to the AVATAR, namely Raspberry Pi 4 model B (RASPBERRY PI, 2023), hereinafter named only as Raspberry Pi, with 4 GB SDRAM and Quad core Cortex-A72 (ARM v8) 64-bit SoC @1,8 GHz.

Two types of microphones were evaluated, one commonly used in the aftersales market, model RS-120MIC (ROADSTAR, 2021), condenser type, with frequency response of 50 Hz to 20 kHz and sensitivity of -32 dB ±3 dB (Figure 42 a). The second one is utilized in the VW UP (Figure 42 b), manufactured by the company BURY, with hypercardioid characteristics, electret type, output level Ua (Meß) of 67 mVeff ±3 dB, acoustic pressure pMeß of 0,1 Pa, and frequency response not informed (BURY, 2024).

It is important to note that the Raspberry Pi does not possess an integrated microphone or an input jack, nevertheless, it is possible to utilize the USB ports or Bluetooth capability to establish an external microphone connection. Due to the discrepancy of the Bluetooth connection when compared to both the C-Bot and a regular passenger vehicle scenario, the USB option was chosen for experimentation.

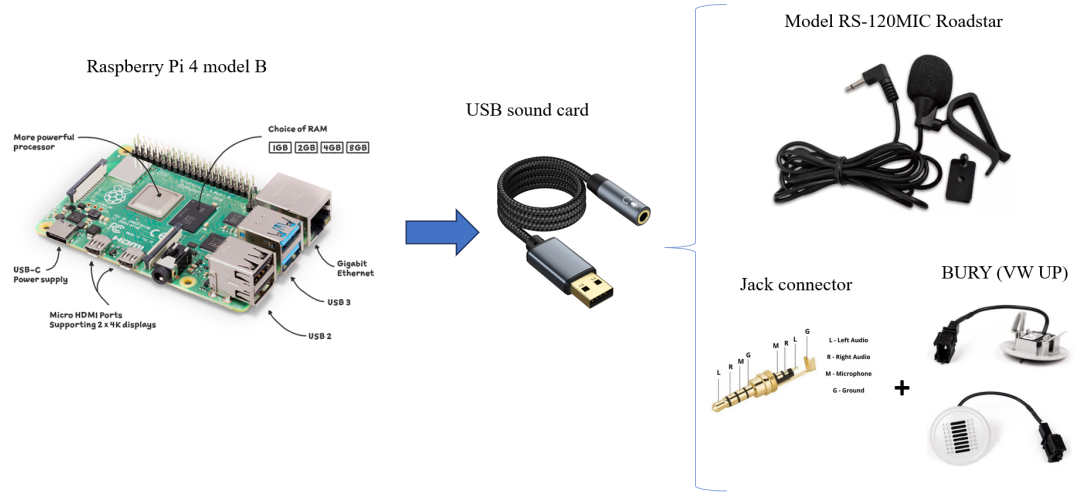
Figure 42 – Microphones utilized in the experiments.



Source: Author, "adapted from" Roadstar (2021) and BURY (2024)

To enable seamless switching between the two types of microphones (Roadstar and BURY) during the evaluation trials, the original connector of the BURY microphone was substituted with a 3,5 mm jack connector consistent with the one originally supplied by Roadstar. The connection to the Raspberry Pi was established using a jack adapter, which emulates a sound card easier than a audio hat for Raspberry Pi with a jack or RCA connector. The schematic of this proposal is presented on Figure 43 below.

Figure 43 – Connection schematic among the Raspberry Pi, the sound card and the different microphones utilized during the training / classification flow and evaluation flow.



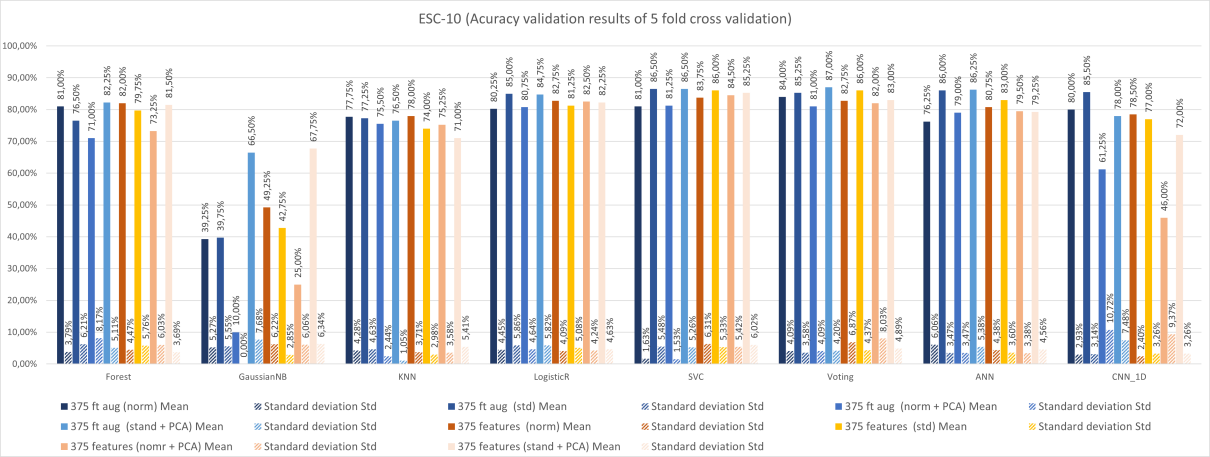
Source: Author, "adapted from" Raspberry Pi (2023), Roadstar (2021) and BURY (2024)

The evaluation trials were performed inside of VW UP using the the ESC-AV dataset, and the audio sound was reproduced through a sound box following as close as possible the same SPL of the real events.

5 RESULTS

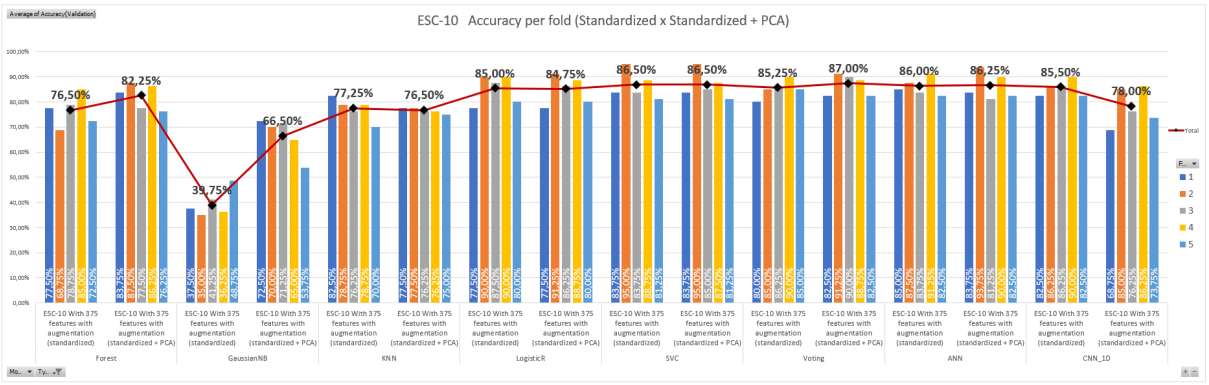
In addition to employing grid search and built-in cross-validation to optimize the classifiers described in Chapter 4, the impact of two methods to rescale the features was evaluated: normalization and standardization. Further more, the influence of PCA by considering 98% of the explained variance was also investigated, resulting in approximately a 50% reduction in the number of features. The results were evaluated in Excel, with examples using the ESC-10 dataset depicted in Figures 44 and 45. The results are compiled in the subsequent sections only as tables.

Figure 44 – Overview of the classifiers results during the training / classification flow - Example with the ESC-10 dataset.



Source: Author

Figure 45 – Comparison of the best option for the classifiers results per fold during the training / classification flow - Example with the ESC-10 dataset standardized x standardized + PCA



Source: Author

5.1 BENCHMARK

This section presents the basic benchmark metrics gathered within the body of literature (Chapter 3). Starting with human accuracy, according to Karol J. Piczak (2015b) an average accuracy rate of 95,7% was attained for the ESC-10 dataset, while the ESC-50 dataset achieved an accuracy rate of 81,3%. The recall rates for individual sound event classes displayed significant variation, ranging from 34,1% for washing machine noise to nearly 100% for crying babies and barking dogs. These experiments indicate that proficient and attentive listeners have the potential to achieve perfect scores on the smaller dataset and are likely to reach accuracy levels of approximately 90% on the main dataset, albeit with some uncertainties when categorizing more ambiguous mechanical noises and soundscapes.

To ensure the soundness of the ESR algorithm, accuracy rates of benchmark datasets were initially compiled from the literature, but only the ones that fully comply with the dataset specifications, more specifically, with k-fold cross-validation setting of 5 for ESC-10, 3 for BDLib2 and 10 for US8K. Thus the values were collected from (PICZAK, Karol J., 2015b), Bountourakis, Vrysis, and Papanikolaou (2015)[1] with STi method, Salamon, Jacoby, and Bello (2014), and Vandendriessche et al. (2021) in the Table 6.

Table 6 – Compilation of the accuracy results to establish a benchmark on the utilized datasets.

Classifier	**ESC-10**	**BDLib2**	**US8K**
k-NN	66,7%	—	56,0%
GNB	—	—	—
SVM	67,0%	—	69,0%
LR	—	73,6%	—
Random Forest	72,7%	—	66,0%
Voting soft	—	—	10,0%
ANN	—	75,2%	—
CNN 1D	83,2%	74,4%	60,1%
GLM	—	75,0%	—
Decision tree	—	—	48,0%
CNN 2D	???	???	???

Source: Author

[1]The ETi method achieved higher results, with 81,5% ANN, 80,4% GLM, and 74,8% LR.

5.2 TRAINING AND CLASSIFICATION FLOW

In this initial stage of the experiments, the main objective is to evaluate the ESR algorithm implemented in several classifiers regarding its feature selection, feature extraction, classification metrics, processing memory, allocation memory and response time. The evaluation was performed both in a notebook environment and embedded in the Raspberry Pi 4.

To evaluate feature selection, a Python script was developed to generate various models for k-fold cross-validation. These models included normalization, standardization, normalization + PCA, and standardization + PCA. Each model was applied to both the original and augmented datasets. On average, the accuracy rates of the top-performing classifiers increased among the datasets: 4,7% in the ESC-10, 11,3% in the BDLib2, and 1,7% in the US8K. It could also be observed that the US8K dataset was the least impacted by the augmentation, which is understandable considering its substantial total duration of 8,75 hours (8.732 samples), in contrast to the 0,55 hours (400 samples) of the ESC-10 and 0,5 hours (180 samples) of the BDLib2 datasets. Notably, Random Forest was less affected while the neural networks, ANN and CNN 1D, benefited the most from the augmentation. Further more, as depicted in Table 7, the winning classifiers were mostly the models **augmented standardized** and **augmented standardized + PCA**. The differences between the exceptions and these models were marginal.

Table 7 – Accuracy rates overview using the benchmark datasets - The color gradient is focused on the classifiers utilized in the models augmented and original, line by line.

	Classifiers	Augmented								Original							
Model		norm		stand		norm + PCA		stand + PCA		norm		stand		norm + PCA		stand + PCA	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
ESC10	Forest	81,00%	3,79%	76,50%	6,21%	71,00%	8,17%	82,25%	5,11%	82,00%	4,47%	79,75%	5,76%	73,25%	6,03%	81,50%	3,69%
	GNB	39,25%	5,27%	39,75%	5,55%	10,00%	0,00%	66,50%	7,68%	49,25%	6,22%	42,75%	2,85%	25,00%	6,06%	67,75%	6,34%
	KNN	77,75%	4,28%	77,25%	4,63%	75,50%	2,44%	76,50%	1,05%	78,00%	3,71%	74,00%	2,98%	75,25%	3,58%	71,00%	5,41%
	LR	80,25%	4,45%	85,00%	5,86%	80,75%	4,64%	84,75%	5,82%	82,75%	4,09%	81,25%	5,08%	82,50%	4,24%	82,25%	4,63%
	SVM	81,00%	1,63%	86,50%	5,48%	81,25%	1,53%	86,50%	5,26%	83,75%	6,31%	86,00%	5,33%	84,50%	5,42%	85,25%	6,02%
	Voting	84,00%	4,09%	85,25%	3,58%	81,00%	4,09%	87,00%	4,20%	82,75%	6,87%	86,00%	4,37%	82,00%	8,03%	83,00%	4,89%
	ANN	76,25%	6,06%	86,00%	3,47%	79,00%	3,47%	86,25%	5,38%	80,75%	4,38%	83,00%	3,60%	79,50%	3,38%	79,25%	4,56%
	CNN 1D	80,00%	2,93%	85,50%	3,14%	61,25%	10,72%	78,00%	7,48%	78,50%	2,40%	77,00%	3,26%	46,00%	9,37%	72,00%	3,26%
BDLib2	Forest	63,89%	5,36%	72,78%	3,47%	55,56%	7,88%	73,33%	4,41%	70,56%	6,74%	73,89%	0,96%	63,89%	5,36%	72,22%	11,71%
	GNB	17,22%	3,85%	20,00%	0,00%	10,56%	0,96%	60,00%	6,67%	22,22%	2,55%	21,11%	1,92%	26,67%	3,33%	52,22%	3,47%
	KNN	66,67%	7,26%	68,33%	7,26%	66,67%	4,41%	66,67%	8,66%	62,22%	9,18%	63,33%	6,01%	61,11%	5,85%	55,00%	7,64%
	LR	70,00%	8,82%	78,89%	2,55%	69,44%	9,18%	77,78%	2,55%	68,33%	5,00%	76,67%	1,67%	68,89%	5,36%	77,78%	2,55%
	SVM	67,22%	4,81%	75,00%	1,67%	67,78%	6,74%	72,78%	0,96%	75,00%	5,00%	74,44%	2,55%	70,56%	3,85%	71,67%	3,33%
	Voting	67,22%	3,85%	78,33%	0,00%	69,44%	6,94%	76,67%	5,00%	59,44%	8,22%	73,33%	2,89%	57,78%	8,39%	72,78%	4,19%
	ANN	61,67%	13,64%	77,78%	3,47%	64,44%	7,52%	76,11%	2,55%	62,22%	11,10%	66,11%	9,18%	55,56%	11,82%	73,89%	12,06%
	CNN 1D	63,89%	8,22%	73,33%	5,77%	30,56%	2,55%	64,44%	6,74%	64,44%	13,57%	64,44%	6,94%	34,44%	11,10%	45,56%	10,84%
US8K	Forest	58,15%	5,59%	57,07%	6,30%	55,77%	5,76%	65,36%	2,35%	60,88%	5,92%	62,08%	3,07%	55,06%	6,81%	65,13%	3,60%
	GNB	20,33%	3,32%	37,64%	5,32%	7,87%	3,01%	37,36%	3,98%	26,31%	4,04%	37,33%	5,75%	12,80%	1,27%	38,82%	4,49%
	KNN	59,16%	7,30%	59,82%	4,35%	58,01%	7,05%	58,13%	5,28%	57,99%	6,37%	56,98%	4,68%	58,51%	6,21%	55,66%	4,89%
	LR	62,16%	6,37%	68,91%	3,46%	61,78%	6,24%	68,67%	3,05%	65,74%	4,89%	68,08%	3,69%	64,79%	5,43%	68,19%	3,57%
	SVM	61,21%	6,16%	64,91%	4,53%	61,70%	5,55%	65,14%	3,89%	64,10%	6,34%	63,90%	3,07%	63,81%	6,46%	63,68%	3,58%
	Voting	63,48%	6,61%	68,08%	3,03%	63,87%	6,66%	66,87%	3,36%	66,09%	6,34%	66,98%	4,25%	66,55%	7,14%	66,48%	4,28%
	ANN	57,10%	8,14%	69,83%	4,70%	54,74%	8,07%	69,23%	4,93%	58,97%	8,78%	69,48%	4,31%	60,19%	6,52%	69,20%	3,90%
	CNN 1D	49,46%	6,18%	71,34%	4,61%	52,92%	6,02%	67,28%	5,51%	54,00%	7,06%	69,36%	5,34%	50,24%	5,06%	64,51%	3,48%

Source: Author

When analyzing the entire datasets with their respective classifiers, notable performance was observed among classical machine learning techniques, with **LR** and **SVM** emerging as the top-performers. Among ensemble methods, **Voting soft** proved effective, while in the realm of neural networks, both **ANN** and **CNN 1D** demonstrated strong performance, albeit with some variability across datasets. Consistently, a pattern emerged wherein models **augmented standardized** and **augmented standardized + PCA** yielded the best results, as illustrated dataset by dataset with color gradient in Table 8.

Table 8 – Accuracy rates overview using the benchmark datasets - The color gradient is focused on the classifiers utilized in the models augmented and original, dataset by dataset.

	Classifiers	Augmented								Original							
Model		norm		stand		norm + PCA		stand + PCA		norm		stand		norm + PCA		stand + PCA	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
ESC10	Forest	81,00%	3,79%	76,50%	6,21%	71,00%	8,17%	82,25%	5,11%	82,00%	4,47%	79,75%	5,76%	73,25%	6,03%	81,50%	3,69%
	GaussianNB	39,25%	5,27%	39,75%	5,55%	10,00%		66,50%	7,68%	49,25%	6,22%	42,75%	2,85%	25,00%	6,06%	67,75%	6,34%
	KNN	77,75%	4,28%	77,25%	4,63%	75,50%	2,44%	76,50%	1,05%	78,00%	3,71%	74,00%	2,98%	75,25%	3,58%	71,00%	5,41%
	LogisticR	80,25%	4,45%	85,00%	5,86%	80,75%	4,64%	84,75%	5,82%	82,75%	4,09%	81,25%	5,08%	82,50%	4,24%	82,25%	4,63%
	SVC	81,00%	1,63%	86,50%	5,48%	81,25%	1,53%	86,50%	5,26%	83,75%	6,31%	86,00%	5,33%	84,50%	5,42%	85,25%	6,02%
	Voting	84,00%	4,09%	85,25%	3,58%	81,00%	4,09%	87,00%	4,20%	82,75%	6,87%	86,00%	4,37%	82,00%	8,03%	83,00%	4,89%
	ANN	76,25%	6,06%	86,00%	3,47%	79,00%	3,47%	86,25%	5,38%	80,75%	4,38%	83,00%	3,60%	79,50%	3,38%	79,25%	4,56%
	CNN_1D	80,00%	2,93%	85,50%	3,14%	61,25%	10,72%	78,00%	7,48%	78,50%	2,40%	77,00%	3,26%	46,00%	9,37%	72,00%	3,26%
BDLib2	Forest	63,89%	5,36%	72,78%	3,47%	55,56%	7,88%	73,33%	4,41%	70,56%	6,74%	73,89%	0,96%	63,89%	5,36%	72,22%	11,71%
	GaussianNB	17,22%	3,85%	20,00%	0,00%	10,56%	0,96%	60,00%	6,67%	22,22%	2,55%	21,11%	1,92%	26,67%	3,33%	52,22%	3,47%
	KNN	66,67%	7,26%	68,33%	7,26%	66,67%	4,41%	66,67%	8,66%	62,22%	9,18%	63,33%	6,01%	61,11%	5,85%	55,00%	7,64%
	LogisticR	70,00%	8,82%	78,89%	2,55%	69,44%	9,18%	77,78%	2,55%	68,33%	5,00%	76,67%	1,67%	68,89%	5,36%	77,78%	2,55%
	SVC	67,22%	4,81%	75,00%	1,67%	67,78%	6,74%	72,78%	0,96%	75,00%	5,00%	74,44%	2,55%	70,56%	3,85%	71,67%	3,33%
	Voting	67,22%	3,85%	78,33%	0,00%	69,44%	6,94%	76,67%	5,00%	59,44%	8,22%	73,33%	2,89%	57,78%	8,39%	72,78%	4,19%
	ANN	61,67%	13,64%	77,78%	3,47%	64,44%	7,52%	76,11%	2,55%	62,22%	11,10%	66,11%	9,18%	55,56%	11,82%	73,89%	12,06%
	CNN_1D	63,89%	8,22%	73,33%	5,77%	30,56%	2,55%	64,44%	6,74%	64,44%	13,57%	64,44%	6,94%	34,44%	11,10%	45,56%	10,84%
US8K	Forest	58,15%	5,59%	57,07%	6,30%	55,77%	5,76%	65,36%	2,35%	60,88%	5,92%	62,08%	3,07%	55,06%	6,81%	65,13%	3,60%
	GaussianNB	20,33%	3,32%	37,64%	5,32%	7,87%	3,01%	37,36%	3,98%	26,31%	4,04%	37,33%	5,75%	12,80%	1,27%	38,82%	4,49%
	KNN	59,16%	7,30%	59,82%	4,35%	58,01%	7,05%	58,13%	5,28%	57,99%	6,37%	56,98%	4,68%	58,51%	6,21%	55,66%	4,89%
	LogisticR	62,16%	6,37%	68,91%	3,46%	61,78%	6,24%	68,67%	3,05%	65,74%	4,89%	68,08%	3,69%	64,79%	5,43%	68,19%	3,57%
	SVC	61,21%	6,16%	64,91%	4,53%	61,70%	5,55%	65,14%	3,89%	64,10%	6,34%	63,90%	3,07%	63,81%	6,46%	63,68%	3,58%
	Voting	63,48%	6,61%	68,08%	3,03%	63,87%	6,66%	66,87%	3,36%	66,09%	6,34%	66,98%	4,25%	66,55%	7,14%	66,48%	4,28%
	ANN	57,10%	8,14%	69,83%	4,70%	54,74%	8,07%	69,23%	4,93%	58,97%	8,78%	69,48%	4,31%	60,19%	6,52%	69,20%	3,90%
	CNN_1D	49,46%	6,18%	71,34%	4,61%	52,92%	6,02%	67,28%	5,51%	54,00%	7,06%	69,36%	5,34%	50,24%	5,06%	64,51%	3,48%

Source: Author

The feature extraction process was evaluated following two methods: the first employed a sliding window of 1 s, sampling rate of 22.050 Hz and 50% overlapping, while the second method utilized entire audio file as input also with sampling rate of 22.050 Hz. For the top-performing classifiers using the original audio data, the accuracy rates increased on average 2,0% in the ESC-10, 13,8% in the BDLib2, and 0,7% in the US8K. On the other hand, when using the augmented datasets, the accuracy rates decreased on average 2,7% in the ESC-10, 1,8% in the BDLib2, and 1,0% in the US8K. The slight decrease in accuracy rates noticeable with the windowed method is barely evident from the gradient colors of the top-performing classifiers showcased in Table 9.

Thus far, the results have shown that the utilization of PCA had minimal impact on the classifiers, with the exception of the CNN 1D. While the sliding window method did decrease

Table 9 – Accuracy rates overview using the benchmark datasets - The color gradient is focused on the classifiers utilized in the models augmented and windowed, dataset by dataset.

	Classifiers	Augmented								Windowed							
Model		norm		stand		norm + PCA		stand + PCA		norm		stand		norm + PCA		stand + PCA	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
ESC10	Forest	81,00%	3,79%	76,50%	6,21%	71,00%	8,17%	82,25%	5,11%	72,50%	3,64%	59,25%	3,26%	57,00%	7,58%	79,50%	8,69%
	GNB	39,25%	5,27%	39,75%	5,55%	10,00%		66,50%	7,68%	34,75%	4,63%	36,25%	3,42%	10,50%	0,68%	55,25%	4,45%
	KNN	77,75%	4,28%	77,25%	4,63%	75,50%	2,44%	76,50%	1,05%	72,00%	3,26%	69,50%	4,73%	69,50%	5,49%	72,00%	7,37%
	LR	80,25%	4,45%	85,00%	5,86%	80,75%	4,64%	84,75%	5,82%	75,50%	6,94%	83,50%	5,03%	75,00%	8,71%	83,25%	6,22%
	SVM	81,00%	1,63%	86,50%	5,48%	81,25%	1,53%	86,50%	5,26%	71,50%	3,58%	81,50%	4,28%	71,75%	4,73%	82,00%	6,03%
	Voting	84,00%	4,09%	85,25%	3,58%	81,00%	4,09%	87,00%	4,20%	79,75%	4,45%	85,00%	3,85%	76,00%	6,15%	84,50%	5,70%
	ANN	76,25%	6,06%	86,00%	3,47%	79,00%	3,47%	86,25%	5,38%	67,75%	5,41%	83,00%	5,56%	71,25%	6,56%	83,25%	8,51%
	CNN 1D	80,00%	2,93%	85,50%	3,14%	61,25%	10,72%	78,00%	7,48%	52,00%	5,70%	84,75%	5,41%	52,00%	8,18%	78,50%	6,52%
BDLib2	Forest	63,89%	5,36%	72,78%	3,47%	55,56%	7,88%	73,33%	4,41%	60,00%	7,26%	53,89%	1,92%	56,67%	3,33%	66,67%	3,33%
	GNB	17,22%	3,85%	20,00%	0,00%	10,56%	0,96%	60,00%	6,67%	23,33%	8,82%	19,44%	4,19%	10,00%	0,00%	60,56%	12,62%
	KNN	66,67%	7,26%	68,33%	7,26%	66,67%	4,41%	66,67%	8,66%	59,44%	1,92%	67,22%	6,74%	58,89%	3,47%	67,22%	6,94%
	LR	70,00%	8,82%	78,89%	2,55%	69,44%	9,18%	77,78%	2,55%	65,56%	10,58%	76,11%	1,92%	62,78%	11,10%	73,33%	2,89%
	SVM	67,22%	4,81%	75,00%	1,67%	67,78%	6,74%	72,78%	0,96%	69,44%	2,55%	73,89%	2,55%	67,22%	6,31%	73,89%	4,19%
	Voting	67,22%	3,85%	78,33%	0,00%	69,44%	6,94%	76,67%	5,00%	67,78%	5,36%	77,78%	1,92%	66,67%	7,64%	75,56%	1,92%
	ANN	61,67%	13,64%	77,78%	3,47%	64,44%	7,52%	76,11%	2,55%	60,56%	6,74%	76,67%	5,77%	56,11%	3,47%	76,11%	3,85%
	CNN 1D	63,89%	8,22%	73,33%	5,77%	30,56%	2,55%	64,44%	6,74%	51,67%	10,14%	75,56%	6,74%	49,44%	7,52%	73,89%	6,74%
US8K	Forest	58,15%	5,59%	57,07%	6,30%	55,77%	5,76%	65,36%	2,35%	54,20%	4,83%	49,86%	3,87%	45,53%	5,01%	64,55%	4,72%
	GNB	20,33%	3,32%	37,64%	5,32%	7,87%	3,01%	37,36%	3,98%	29,83%	5,33%	35,43%	5,46%	15,83%	1,75%	36,88%	5,35%
	KNN	59,16%	7,30%	59,82%	4,35%	58,01%	7,05%	58,13%	5,28%	51,41%	6,20%	56,58%	5,12%	51,38%	5,41%	54,97%	4,66%
	LR	62,16%	6,37%	68,91%	3,46%	61,78%	6,24%	68,67%	3,05%	53,18%	4,40%	68,21%	4,21%	52,50%	5,02%	68,19%	3,82%
	SVM	61,21%	6,16%	64,91%	4,53%	61,70%	5,55%	65,14%	3,89%	48,26%	7,18%	65,11%	3,55%	47,39%	7,52%	65,99%	4,56%
	Voting	63,48%	6,61%	68,08%	3,03%	63,87%	6,66%	66,87%	3,36%	55,33%	5,09%	67,69%	4,11%	50,87%	6,76%	66,16%	4,29%
	ANN	57,10%	8,14%	69,83%	4,70%	54,74%	8,07%	69,23%	4,93%	46,52%	7,77%	68,52%	4,47%	37,37%	8,74%	68,83%	5,08%
	CNN 1D	49,46%	6,18%	71,34%	4,61%	52,92%	6,02%	67,28%	5,51%	40,39%	8,37%	68,65%	4,01%	46,79%	7,60%	66,20%	5,09%

Source: Author

the accuracy rates of the top-performers, the reduction was mild, and the advantage to have an audio signal recognized in such a short term outweighs this reduction. Overall, it is important to note that neither the CNN 2D nor other metrics, such as memory allocation, processing memory, and response time, were assessed. Therefore, while these initial findings are promising, it is premature to definitively conclude that this approach (PCA) is heading in the correct direction.

To finish the feature selection evaluation, a few investigations were also performed using mutual information from scikit-learn. Mutual information is calculated between two variables and measures the reduction in uncertainty for one variable given a known value of the other variable. Given two random variables X and Y, it can be stated formally as follows:

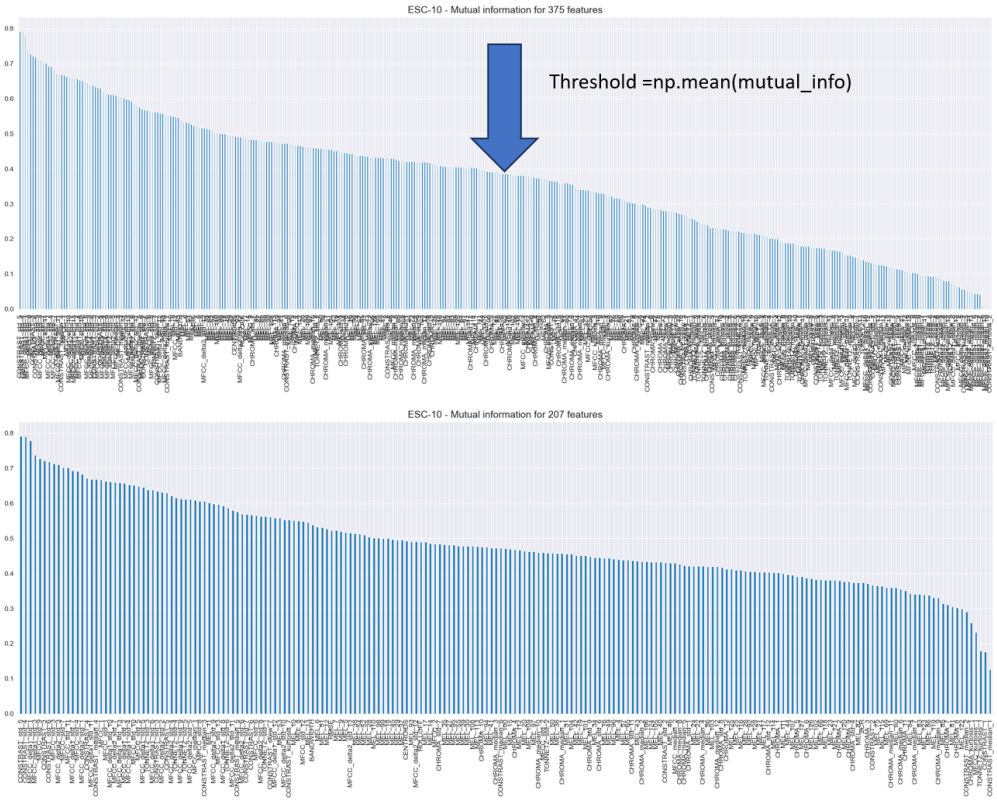
$$I(X;Y) = H(X) - H(X|Y) \quad \text{With entropy:} \quad H = -\sum_{i}^{C} p_i \log_2 p_i \tag{15}$$

Where $I(X;Y)$ is the mutual information for X and Y, $H(X)$ is the entropy for X, $H(X|Y)$ is the conditional entropy for X given Y, and p_i is the probability of randomly picking an element of class i (i.e. the proportion of the dataset made up of class i). Since mutual information is a measure of dependence or "mutual dependence" between two random variables, the result measure is symmetrical, meaning that $I(X;Y) = I(Y;X)$, measured as units of bits.

To establish a logical parameter, a threshold was calculated by taking the mean of the mutual information, considering only the features surpassing this threshold. Figure 46 illustrates

an example using the ESC-10 dataset, displaying 375 features in the upper image and 207 in the lower image after applying the $KBest$ feature selection based on mutual information. Remarkably, none of the native 12 features from Table 5 were completely discarded, and among the statistics features - mean, median, std, skewness and kurtosis - the latter two were the least frequently selected. It is noteworthy that the number of selected features is directly influenced by the dataset and the information within the k-fold. Following cross-validation, the number of features selected averaged 202 $\pm$ 7 for ESC-10, 168 $\pm$ 18 for BDLib2, 178 $\pm$ 3 for US8K, and 168 $\pm$ 5 for the tailored dataset US8k_AV.

Figure 46 – Original features x selected features using mutual information with threshold established above the mean of the mutual information.

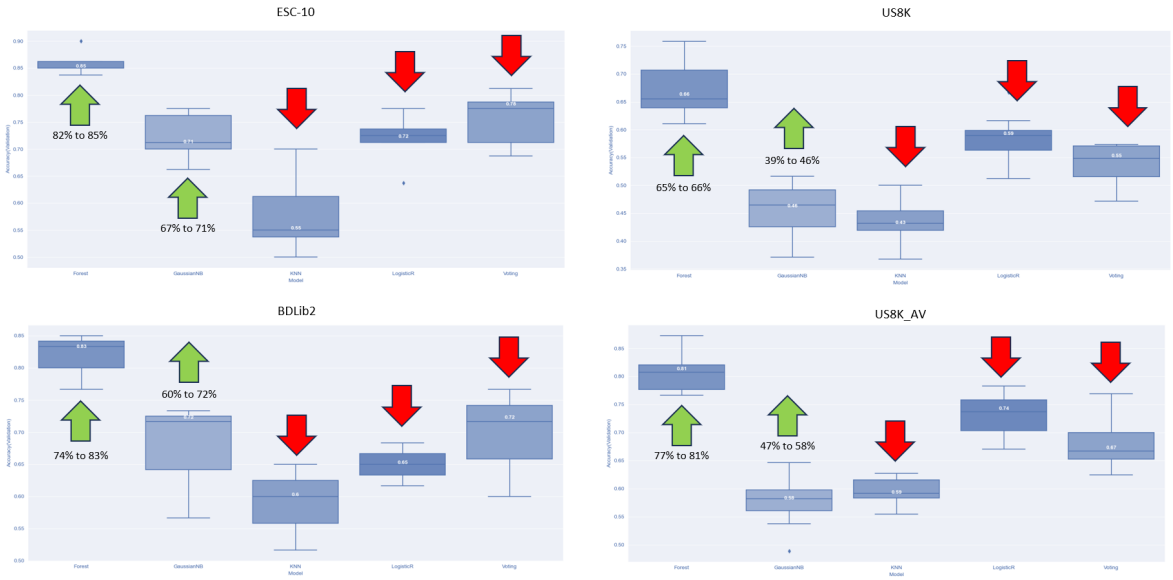


Source: Author

The Random Forest and GNB classifiers performed better when mutual information was utilized. Conversely, the accuracy rates of the remaining classifiers significantly worsened and no conclusion could be drawn on SVM because the algorithm didn't converge in the US8K dataset, however, SVM will most likely worsen too given that it worsened in the ESC-10 and BDLib2 datasets where it converged. Notably, Random Forest matched the top-performing classifiers in the US8K_AV dataset and therefore, it continues as a strong candidate to be further

evaluated considering other metrics such as processing memory, allocation memory and response time. Figure 47 illustrates the accuracy results of the datasets using box plots for the augmented non-windowed model, highlighting the improvements (in green) against the highest accuracy results among the classifiers and models.

Figure 47 – Box plot of the accuracy results after utilizing mutual information in the augmented non-windowed models.



Source: Author

The dataset created in subsection 4.2.4, referred to as US8K_AV, consists of 4.358 files distributed across 5 classes. These files collectively represent a total duration of 4,84 hours of recorded sounds. To maintain consistency and facilitate accuracy comparisons, the original 10-fold split for cross-validation from the source dataset (US8K) was retained.

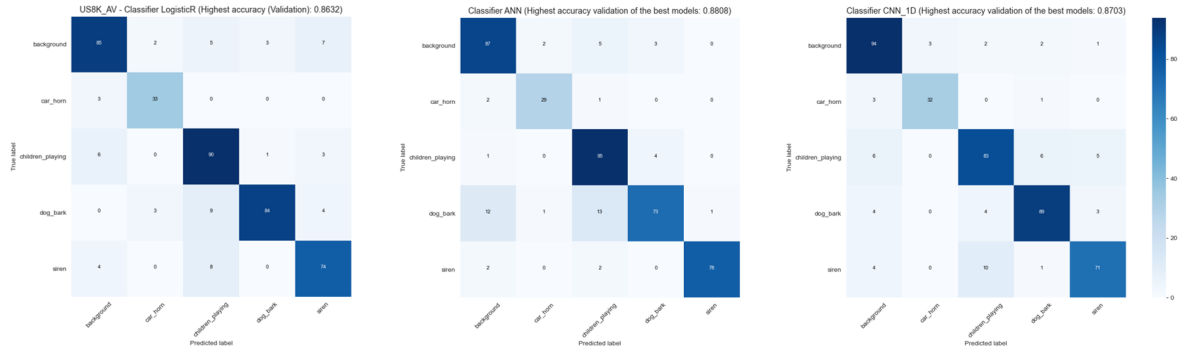
Similarly to the US8K dataset, the augmentation process also resulted in a slight improvement (1,1%) in accuracy rates for the top-performing classifiers. However, there were no significant differences observed when comparing the overall results within this dataset with those obtained using other datasets. Notably, the machine learning technique **LR** consistently outperformed other methods with averaged accuracy of 81%, while among the neural networks, **CNN 1D** emerged as the most effective model with 82% on the standardized models, dropping to 79% when PCA is utilized. Table 10 depicts the accuracy rates of the US8K_AV together with the confusion matrices for the best results, and although not explicitly illustrated in the table, the recall on the ANN model was more estable (82,10% $\pm$ 0,65%) than the CNN 1D model (80,64% $\pm$ 3,00%). Similarly to the findings presented in Table 9, the accuracy results of the

most effective classifiers on the US8K_AV dataset exhibited a marginal decline of 1% when employing the sliding process. This reduction was once again scarcely perceptible, as illustrated by the color gradient showcased in Table 11.

Table 10 – Accuracy rates overview and confusion matrices of the best models using the tailored dataset US8K_AV - The color gradient is focused on the classifiers utilized in the models augmented and original, dataset by dataset.

	Classifiers	Augmented								Original							
Model		norm		stand		norm + PCA		stand + PCA		norm		stand		norm + PCA		stand + PCA	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
US8K_AV	Forest	70,19%	5,05%	74,91%	3,67%	64,16%	6,89%	77,46%	3,93%	73,47%	5,00%	76,66%	4,13%	64,37%	8,14%	76,11%	3,87%
	GNB	26,90%	5,04%	44,42%	5,91%	9,88%	4,01%	46,09%	5,59%	35,64%	5,39%	46,92%	7,59%	13,34%	5,69%	45,32%	5,97%
	KNN	73,01%	3,92%	72,72%	4,92%	71,67%	4,76%	71,48%	5,04%	72,41%	3,32%	69,72%	5,18%	72,77%	3,43%	68,45%	5,87%
	LR	77,78%	3,01%	81,12%	2,51%	78,65%	2,67%	81,10%	2,89%	79,99%	2,55%	81,15%	2,72%	79,63%	2,42%	81,02%	2,70%
	SVM	77,15%	5,02%	77,42%	2,33%	78,80%	3,86%	78,21%	2,74%	78,55%	1,44%	75,56%	2,41%	78,86%	1,89%	76,68%	2,93%
	Voting	79,00%	3,24%	80,98%	3,65%	79,80%	2,69%	77,78%	3,57%	80,50%	1,07%	80,66%	3,07%	80,42%	2,18%	78,13%	2,68%
	ANN	71,94%	6,84%	82,02%	3,93%	69,39%	10,47%	81,95%	3,25%	75,25%	3,49%	82,18%	3,10%	73,84%	5,98%	81,36%	2,80%
	CNN 1D	62,57%	5,46%	82,64%	2,99%	67,70%	8,02%	79,92%	3,42%	68,79%	8,76%	81,26%	2,53%	67,24%	6,37%	78,05%	2,75%

US8K_AV – Confusion matrices of the top-performing classifiers



Source: Author

Table 11 – Accuracy rates overview using the tailored dataset US8K_AV - The color gradient is focused on the classifiers utilized in the models augmented and windowed, dataset by dataset.

	Classifiers	Augmented								Windowed							
Model		norm		stand		norm + PCA		stand + PCA		norm		stand		norm + PCA		stand + PCA	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
US8K_AV	Forest	70,19%	5,05%	74,91%	3,67%	64,16%	6,89%	77,46%	3,93%	67,33%	6,02%	65,95%	3,33%	53,04%	8,60%	75,80%	3,55%
	GNB	26,90%	5,04%	44,42%	5,91%	9,88%	4,01%	46,09%	5,59%	44,57%	8,47%	47,70%	7,25%	26,83%	1,78%	44,94%	5,56%
	KNN	73,01%	3,92%	72,72%	4,92%	71,67%	4,76%	71,48%	5,04%	67,77%	3,73%	69,71%	5,42%	67,07%	3,63%	68,78%	5,28%
	LR	77,78%	3,01%	81,12%	2,51%	78,65%	2,67%	81,10%	2,89%	62,87%	9,11%	80,43%	2,54%	61,67%	9,63%	80,57%	2,93%
	SVM	77,15%	5,02%	77,42%	2,33%	78,80%	3,86%	78,21%	2,74%	60,76%	9,48%	77,38%	2,88%	61,08%	9,34%	78,50%	2,93%
	Voting	79,00%	3,24%	80,98%	3,65%	79,80%	2,69%	77,78%	3,57%	66,97%	6,41%	79,69%	2,24%	61,20%	8,56%	76,52%	2,96%
	ANN	71,94%	6,84%	82,02%	3,93%	69,39%	10,47%	81,95%	3,25%	57,88%	8,45%	81,24%	2,96%	48,24%	9,52%	81,26%	2,74%
	CNN 1D	62,57%	5,46%	82,64%	2,99%	67,70%	8,02%	79,92%	3,42%	54,94%	12,17%	80,44%	3,58%	52,76%	12,10%	79,62%	2,73%

Source: Author

Now that benchmark and tailored datasets are defined, the next step is to implement a CNN 2D model utilizing aggregated features (log-mel spectrogram, Tonnetz, SCT and Chroma), compare it with a model that tries to capture the sound variation along time (log-mel spectrogram,

$\triangle$MFCC and $\triangle\triangle$MFCC), and assess the remaining evaluation metrics (processing memory, allocation memory and response time). In the evaluation flow, deploy the wining classifier on the RaspBerry Pi and finish by assessing the benchmark metrics from the training and classification flow.

6 CONCLUSION AND FUTURE WORK

The reported accuracy results on the datasets exhibit significant inconsistency in the body of literature, aligning with the initial concern highlighted by Ho, Wong, and Goh (2020) that "high accuracy does not imply reproducibility". The three benchmark datasets exhibit notable distinctions, despite originating from the same source repository (Freesound), they differ significantly in terms of sample duration and sound quality. ESC-10 and BDLib2 showcase cleaner signals predominantly focused on foreground sounds, whereas BDLib2 suffers from a lack of sample representativeness. Conversely, US8K presents a mixture of background and foreground sounds and boasts a much larger sample size compared to both ESC-10 and BDLib2. One could posit a hypothesis that ESC-10 would achieve the highest accuracy rates, followed by BDLib2 and US8K, in line with some findings in the literature, either using classical machine learning techniques such as (SILVA et al., 2019) and (BOUNTOURAKIS et al., 2019) or CNN 1D such as (VANDENDRIESSCHE et al., 2021), notably, these authors followed strictly the dataset specifications regarding k-fold cross-validation. Others like (LHOEST et al., 2021) and (LUZ et al., 2021) achieved remarkably high accuracy rates for US8K using Random Forest and k-NN, respectively 94,2% and 93,2%), however, they didn't comply with the dataset specifications. For comparison, these results surpass the official highest score forUS8K obtained with deep neural networks, which stood at 90% in 2022. The results using the proposed ESR algorithm either overcame or matched the literature in terms of accuracy rates.

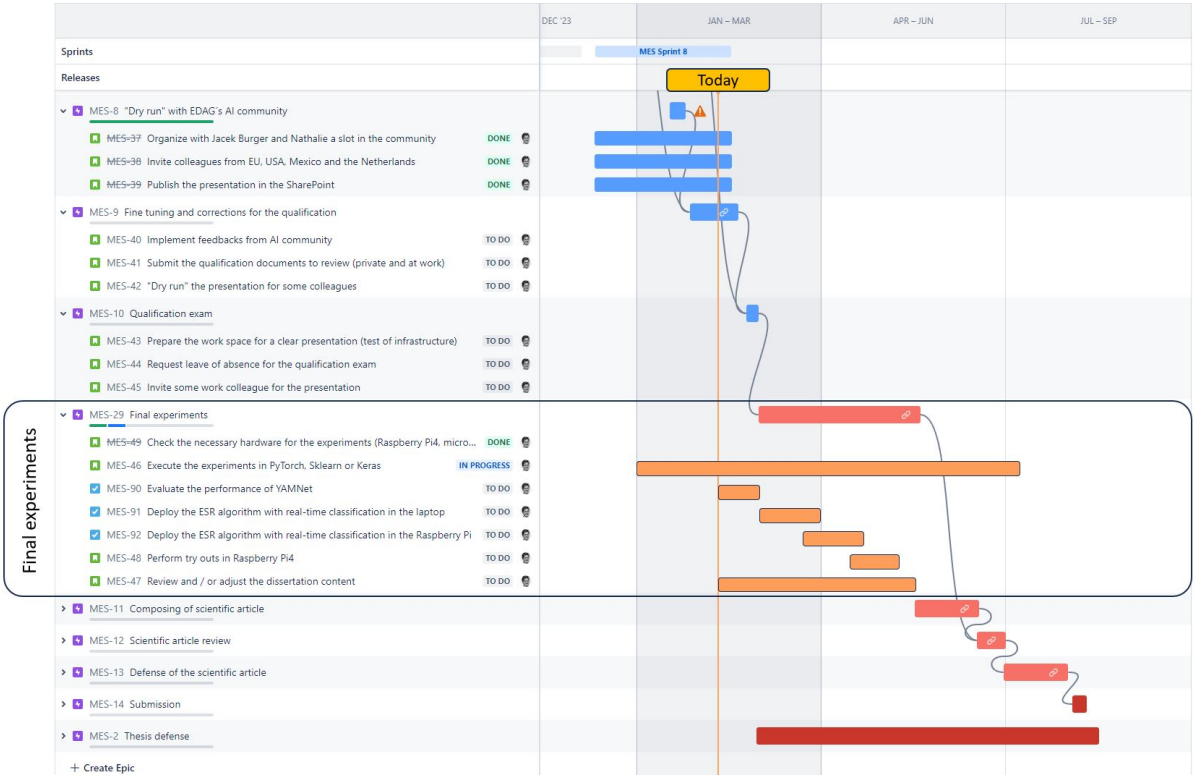
The accuracy rates achieved on the tailored dataset US8K_AV ($\sim$81%) remain lower than the average human accuracy of 95,7% attained in the ESC-10 dataset, despite the variation in class composition between the two datasets. Nevertheless, it is important to note that accuracy is just one of the metrics being assessed in this project. Additionally, the output of the ESR algorithm will be combined with data from other sensors, and collectively, they will generate specific actions in the autonomous vehicle.

Based on the experiments conducted thus far, we can state that the methodology proposed for the training/classification flow as well as the feature selection process are sound. The sliding window method for feature extraction had a similar performance to the entire audio length method and therefore, it will provide results to the EE architecture after 1 s + t ms response time that a sound source is detected.

6.1 ROADMAP - NEXT STEPS

This project was published on the Attlasian website (click here for aflorentino.atlassian) and have been developed under the agile approach, using backlog and sprints artifacts. Figure 48 highlights the main tasks within the epic "Final experiments", and Figure 49 presents the complete roadmap, including the current sprint and project backlog in the lower area.

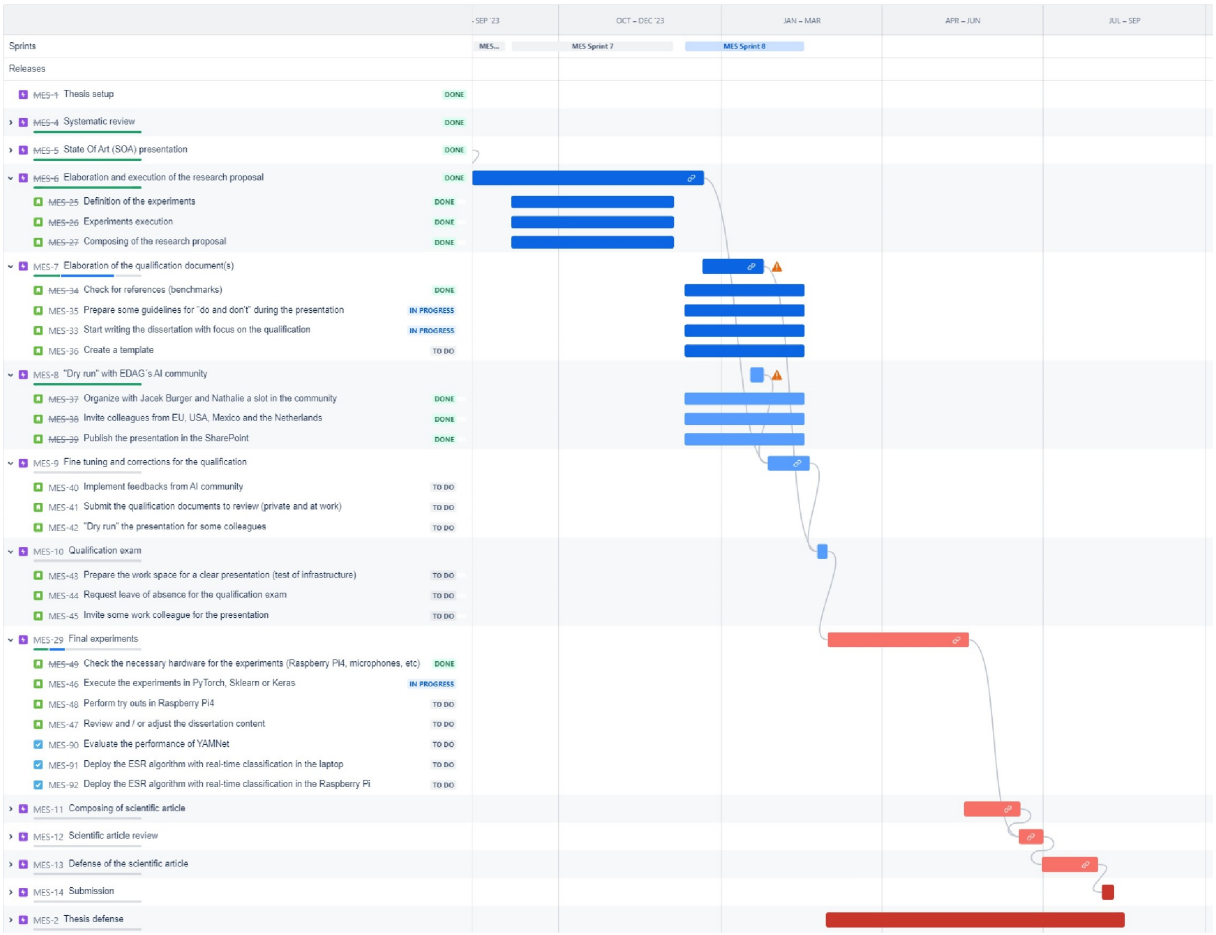
Figure 48 – Nexts steps on the epic "Final experiments".



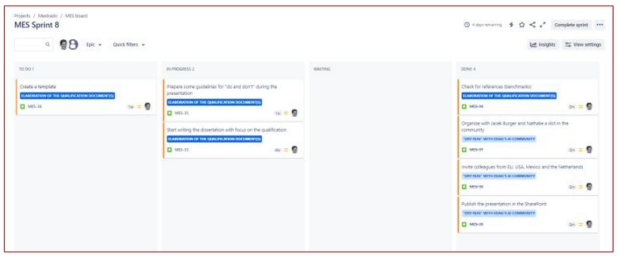
Source: Author

Figure 49 – Roadmap, kanban and backlog.

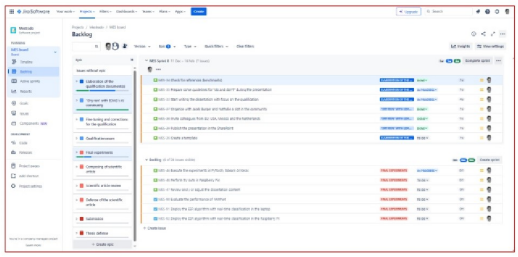
Roadmap



Current sprint (Kanban board)



Project backlog



Source: Author

REFERENCES

ABAYOMI-ALLI, Olusola O et al. Data Augmentation and Deep Learning Methods in Sound Classification: A Systematic Review. **Electronics**, v. 11, n. 22, 2022. ISSN 2079-9292. DOI: 10.3390/electronics11223795.

ABREHA, Gebremedhin Teklemariam. **An environmental audio-based context recognition system using smartphones**. Aug. 2014. PhD thesis. Available from: http://essay.utwente.nl/66444/.

ALÍAS, Francesc; SOCORÓ, Joan Claudi; SEVILLANO, Xavier. A review of physical and perceptual feature extraction techniques for speech, music and environmental sounds. **Applied Sciences**, v. 6, n. 5, 2016. DOI: 10.3390/app6050143.

ANACONDA. **Anaconda: The World's Most Popular Data Science Platform**. [S.l.: s.n.], 2023. Available from: https://www.anaconda.com/. Visited on: 23 Dec. 2023.

APTIV. **What Is an Electronic Control Unit?** [S.l.: s.n.], Feb. 2020. Available from: https://www.aptiv.com/en/insights/article/what-is-an-electronic-control-unit.

AUBAUER, Roland; LECKSCHAT, Dieter. Optimized second-order gradient microphone for hands-free speech recordings in cars. **Speech Communication**, v. 34, n. 1-2, p. 13–23, Apr. 2001. DOI: 10.1016/S0167-6393(00)00043-1. Available from: https://linkinghub.elsevier.com/retrieve/pii/S0167639300000431. Visited on: 6 May 2023.

BANSAL, A; GARG, N K. Environmental Sound Classification using Hybrid Ensemble Model. In: PROCEDIA Computer Science. [S.l.: s.n.], 2022. v. 218, p. 418–428. DOI: 10.1016/j.procs.2023.01.024.

BARBER, David. **Bayesian reasoning and machine learning**. Cambridge, England: Cambridge University Press, Feb. 2012. ISBN 978-0-521-51814-7.

BARTSCH, M A; WAKEFIELD, G H. Audio thumbnailing of popular music using chroma-based representations. **IEEE Transactions on Multimedia**, v. 7, n. 1, p. 96–104, 2005. DOI: 10.1109/TMM.2004.840597.

BBC. **Complete Sound Effects Library**. [S.l.: s.n.], 2023. Available from: https://sound-effects.bbcrewind.co.uk/. Visited on: 9 Dec. 2023.

BELLMAN, Richard E. **Adaptive Control Processes: A Guided Tour**. 1. ed. Princeton: Princeton University Press, 1961. ISBN 978-1-4008-7466-8. DOI: 10.1515/9781400874668.

BISHOP, Christopher M; BISHOP, Hugh. **Deep learning**. 1. ed. Cham, Switzerland: Springer International Publishing, Nov. 2023. ISBN 978-3-031-45467-7.

BLACKMAN, R B; TUKEY, J W. The measurement of power spectra from the point of view of communications engineering — Part I. **The Bell System Technical Journal**, v. 37, n. 1, p. 185–282, 1958. DOI: 10.1002/j.1538-7305.1958.tb03874.x.

BOSCH. **Embedded siren detection | Bosch Global**. [S.l.: s.n.], Jan. 2024. Available from: https://www.bosch.com/stories/embedded-siren-detection/.

BOUGUILA, Nizar; FAN, Wentao. **Mixture models and applications**. Ed. by Nizar Bouguila and Wentao Fan. 1. ed. Cham, Switzerland: Springer Nature, Aug. 2020. Series Title: Unsupervised and Semi-Supervised Learning. ISBN 978-3-030-23878-0.

BOUNTOURAKIS, Vasileios; VRYSIS, Lazaros; PAPANIKOLAOU, George. Machine Learning Algorithms for Environmental Sound Recognition: Towards Soundscape Semantics. In: PROCEEDINGS of the Audio Mostly 2015 on Interaction With Sound. New York, NY, USA: Association for Computing Machinery, 2015. Series Title: AM '15. ISBN 978-1-4503-3896-7. DOI: 10.1145/2814895.2814905.

BOUNTOURAKIS, Vasileios et al. An Enhanced Temporal Feature Integration Method for Environmental Sound Recognition. English. **Acoustics**, Volume 1, issue 2, p. 410–422, 2019. ISSN 2624-599X. DOI: 10.3390/acoustics1020023.

BRANDING, Jelto et al. Towards noise robust acoustic insect detection: from the lab to the greenhouse. **KI - Kunstliche Intelligenz**, 2023. DOI: 10.1007/s13218-023-00812-x.

BREIMAN, Leo. Random Forests. **Machine Learning**, v. 45, n. 1, p. 5–32, 2001. ISSN 1573-0565. DOI: 10.1023/A:1010933404324. Available from: https://doi.org/10.1023/A:1010933404324.

BUBASHAIT, Mohamed; HEWAHI, Nabil. Urban Sound Classification Using DNN, CNN LSTM a Comparative Approach. In: 2021 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies, 3ICT 2021. [S.l.]: Institute of Electrical and Electronics Engineers Inc., Sept. 2021. P. 46–50. ISBN 978-1-66544-032-5. DOI: 10.1109/3ICT53449.2021.9581339.

BURY. **BURY Automotive (OEM) acoustics microphones**. [S.l.: s.n.], Jan. 2024. Available from: https://www.bury.com/en/products/microphones/. Visited on: 6 Jan. 2024.

CHOROMAŃSKA, Anna et al. The Loss Surfaces of Multilayer Networks. In: INTERNATIONAL Conference on Artificial Intelligence and Statistics. [S.l.: s.n.], 2014. arXiv: 1412.0233v3.

CHU, H.-C.; ZHANG, Y.-L.; CHIANG, H.-C. A CNN Sound Classification Mechanism Using Data Augmentation. **Sensors**, v. 23, n. 15, 2023. DOI: 10.3390/s23156972.

DAVIS, S; MERMELSTEIN, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. **IEEE Transactions on Acoustics, Speech, and Signal Processing**, v. 28, n. 4, p. 357–366, 1980. ISSN 0096-3518. DOI: 10.1109/TASSP.1980.1163420.

DEBNATH, Lokenath; SHAH, Firdous Ahmad. **Wavelet transforms and their applications**. 2. ed. Secaucus, NJ: Birkhauser Boston, Nov. 2014. ISBN 978-0-8176-8417-4.

DEICHMANN, JOHANNES et al. **Autonomous driving's future: Convenient and connected**. [S.l.: s.n.], Jan. 2023. Available from: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/autonomous-drivings-future-convenient-and-connected. Visited on: 4 Jan. 2024.

DU, Yu; VARGA, Bálaz; DOBOS, Viktor. A Study on Speech Intelligibility Performance of Automotive Voice Microphones. In: PROCEDINGS [...] Dearborn, USA: Audio Engineering Society, 2022. 2022-June, p. (paper 2). tex.venue: Dearborn, USA tex.eventyear: 2022. Available from: http://www.aes.org/e-lib/browse.cfm?elib=21803. Visited on: 27 May 2023.

DU, Yu et al. Automotive microphone performance: From specification to user experience. In: PROCEDINGS [...] Neuburg an der Donau, Germany: Audio Engineering Society, 2019. 2019-September, p. (paper 5). tex.venue: Neuburg an der Donau, Germany tex.eventyear: 2019. Available from: http://www.aes.org/e-lib/browse.cfm?elib=20531. Visited on: 27 May 2023.

EYBEN, Florian; WÖLLMER, Martin; SCHULLER, Björn. Opensmile: The Munich Versatile and Fast Open-Source Audio Feature Extractor. In: PROCEEDINGS of the 18th ACM International Conference on Multimedia. New York, NY, USA: Association for Computing Machinery, 2010. P. 1459–1462. Series Title: MM '10. ISBN 978-1-60558-933-6. DOI: 10.1145/1873951.1874246. Available from: https://doi.org/10.1145/1873951.1874246.

FERGUSON, David I.; ZHU, Jiajun. **Controlling autonomous vehicle using audio data**. USA, Mar. 2014. US8676427B1. Publisher: United States Patent.

FONT, Frederic; ROMA, Gerard; SERRA, Xavier. Freesound Technical Demo. In: PROCEEDINGS of the 21st ACM International Conference on Multimedia. New York, NY, USA: Association for Computing Machinery, 2013. P. 411–412. Series Title: MM '13. ISBN 978-1-4503-2404-5. DOI: 10.1145/2502081.2502245.

FRIEDMAN, Nir; GEIGER, Dan; GOLDSZMIDT, Moises. Bayesian Network Classifiers. **Machine Learning**, v. 29, n. 2, p. 131–163, Nov. 1997. DOI: 10.1023/A:1007465528199.

FUKUYAMA, Keita et al. Identification of Respiratory Sounds Collected from Microphones Embedded in Mobile Phones. **Advanced Biomedical Engineering**, v. 11, p. 58–67, 2022. DOI: 10.14326/abe.11.58.

GENUER, Robin; POGGI, Jean-Michel. Random Forests. In: USE R! Cham: Springer International Publishing, 2020. P. 33–55. ISBN 978-3-030-56484-1.

GIANNAKOPOULOS, Theodoros; PIKRAKIS, Aggelos. **Introduction to audio analysis**. 1. ed. Oxford, UK: Academic Press, Apr. 2014. ISBN 978-0-08-099388-1.

GOLD, Ben; MORGAN, Nelson; ELLIS, Dan. **Speech and audio signal processing**. 2. ed. Hoboken, NJ: Wiley-Blackwell, Aug. 2011. ISBN 978-0-470-19536-9.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. 1. ed. London, England: MIT Press, Nov. 2016. Series Title: Adaptive Computation and Machine Learning series.

GOOGLE CLOUD. **Introduction to Cloud TPU**. [S.l.: s.n.], Feb. 2024. Available from: https://cloud.google.com/tpu/docs/intro-to-tpu.

HARTE, Christopher; SANDLER, Mark; GASSER, Martin. Detecting harmonic change in musical audio. In: PROCEEDINGS of the 1st ACM Workshop on Audio and Music Computing Multimedia. New York, NY, USA: Association for Computing Machinery, 2006. P. 21–26. Series Title: AMCMM '06. ISBN 1-59593-501-0. DOI: 10.1145/1178723.1178727.

HARTSHORN, Scott. **Machine Learning With Random Forests And Decision Trees: A Visual Guide For Beginners**. [S.l.]: Kindle Direct Publishing, 2016.

HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, Jerome. **The elements of Statistical Learning: Data Mining, Inference, and Prediction**. 2. ed. New York, NY: Springer New York, 2009. ISBN 978-0-387-84857-0.

HECHT-NIELSEN, Robert. Kolmogorov's Mapping Neural Network Existence Theorem. In: FIRST IEEE International Conference on Neural Networks. San Diego, CA: Piscataway, NJ: IEEE, 1987. v. 3, p. 11–14. Available from: https://api.semanticscholar.org/CorpusID:118526925.

HERMES, Dik J. **The perceptual structure of sound**. 1. ed. Cham, Switzerland: Springer International Publishing, June 2023. Series Title: Current Research in Systematic Musicology. ISBN 978-3-031-25565-6.

HINTON, Geoffrey E; OSINDERO, Simon; TEH, Yee-Whye. A Fast Learning Algorithm for Deep Belief Nets. **Neural Computation**, v. 18, n. 7, p. 1527–1554, Feb. 2006. ISSN 0899-7667. DOI: 10.1162/neco.2006.18.7.1527.

HO, Sung Yang; WONG, Limsoon; GOH, Wilson Wen Bin. Avoid Oversimplifications in Machine Learning: Going beyond the Class-Prediction Accuracy. **Patterns**, v. 1, n. 2, p. 100025, May 2020. Publisher: Cell Press. ISSN 26663899. DOI: 10.1016/j.patter.2020.100025.

HUANG, Jeremy Zhengqi; CHHABRIA, Hriday; JAIN, Dhruv. "Not There Yet": Feasibility and Challenges of Mobile Sound Recognition to Support Deaf and Hard-of-Hearing People. In: ASSETS 2023 - Proceedings of the 25th International ACM SIGACCESS Conference on Computers and Accessibility. [S.l.: s.n.], 2023. DOI: 10.1145/3597638.3608431.

HUSSAIN, Jamil et al. Model-based adaptive user interface based on context and user experience evaluation. **Journal on Multimodal User Interfaces**, v. 12, n. 1, p. 1–16, Mar. 2018. ISSN 1783-7677. DOI: 10.1007/s12193-018-0258-2. Available from: http://link.springer.com/10.1007/s12193-018-0258-2.

I-SPOT. **I-SPOT**. [S.l.: s.n.], Jan. 2020. Available from: https://i-spot-project.eu/. Visited on: 12 Jan. 2024.

INDOML. **Student Notes: Convolutional Neural Networks (CNN) Introduction – Belajar Pembelajaran Mesin Indonesia**. [S.l.: s.n.], Feb. 2018. Available from: https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/.

INTEL. **Intel® FPGAs and Programmable Devices-Intel® FPGA**. [S.l.: s.n.], Feb. 2024. Available from: https://www.intel.com/content/www/us/en/products/programmable.html.

JEANTET, Lorène; DUFOURQ, Emmanuel. Improving deep learning acoustic classifiers with contextual information for wildlife monitoring. **Ecological Informatics**, v. 77, 2023. DOI: 10.1016/j.ecoinf.2023.102256.

JEONG, Gilsu; AHN, Changbum R; PARK, Moonseo. Constructing an Audio Dataset of Construction Equipment from Online Sources for Audio-Based Recognition. In: PROCEEDINGS - Winter Simulation Conference. [S.l.: s.n.], 2022. 2022-December, p. 2354–2364. DOI: 10.1109/WSC57314.2022.10015388.

JIANG, Dan-Ning et al. Music type classification by spectral contrast feature. In: IEEE International Conference on Multimedia and Expo. [S.l.: s.n.], 2002. v. 1, p. 113–116. DOI: 10.1109/ICME.2002.1035731.

KERAS. **Keras: Deep Learning for Humans**. [S.l.: s.n.], 2023. Available from: https://keras.io/. Visited on: 23 Dec. 2023.

KLAPURI, Anssi; DAVY, Manuel. **Signal processing methods for music transcription**. Ed. by Anssi Klapuri and Manuel Davy. 1. ed. New York, NY: Springer, May 2006. ISBN 978-0-387-30667-4.

LAMRINI, M; CHKOURI, M Y; TOUHAFI, A. Evaluating the Performance of Pre-Trained Convolutional Neural Network for Audio Classification on Embedded Systems for Anomaly Detection in Smart Cities. **Sensors**, v. 23, n. 13, 2023. DOI: 10.3390/s23136227.

LECUN, Y et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, 1998. DOI: 10.1109/5.726791.

LHOEST, L. et al. MosAIc: A classical machine learning multi-classifier based approach against deep learning classifiers for embedded sound classification. **Applied Sciences (Switzerland)**, v. 11, n. 18, 2021. DOI: 10.3390/app11188394.

LI, Shaobo et al. An ensemble stacked convolutional neural network model for environmental event sound recognition. **Applied Sciences (Switzerland)**, v. 8, n. 7, 2018. DOI: 10.3390/app8071152.

LINDHOLM, Erik et al. NVIDIA Tesla: A Unified Graphics and Computing Architecture. **IEEE Micro**, v. 28, n. 2, p. 39–55, 2008. DOI: 10.1109/MM.2008.31.

LUZ, J.S. et al. Ensemble of handcrafted and deep features for urban sound classification. **Applied Acoustics**, v. 175, 2021. DOI: 10.1016/j.apacoust.2020.107819.

MARCHEGIANI, Letizia; NEWMAN, Paul. Listening for Sirens: Locating and Classifying Acoustic Alarms in City Scenes. **IEEE Transactions on Intelligent Transportation Systems**, v. 23, n. 10, p. 17087–17096, 2022. DOI: 10.1109/TITS.2022.3158076.

MCFEE, Brian et al. librosa: Audio and Music Signal Analysis in Python. In: HUFF, Kathryn; BERGSTRA, James (Eds.). **Proceedings of the 14th Python in Science Conference**. [S.l.: s.n.], 2015. P. 18–24. DOI: 10.25080/Majora-7b98e3ed-003. Available from: http://dx.doi.org/10.25080/Majora-7b98e3ed-003. Visited on: 17 Aug. 2023.

MCFEE, Brian et al. **librosa/librosa: 0.10.1**. [S.l.]: Zenodo, Aug. 2023. DOI: 10.5281/zenodo.8252662. Available from: https://doi.org/10.5281/zenodo.8252662.

MESAROS, Annamaria et al. Sound Event Detection in the DCASE 2017 Challenge. **IEEE/ACM Transactions on Audio, Speech, and Language Processing**, v. 27, n. 6, p. 992–1006, June 2019. ISSN 2329-9290. DOI: 10.1109/TASLP.2019.2907016. Available from: https://ieeexplore.ieee.org/document/8673582/.

MITCHELL, Thomas M. **Machine Learning**. 1. ed. New York, NY: McGraw-Hill Professional, Mar. 1997. Series Title: McGraw-Hill series in computer science. ISBN 978-0-07-042807-2.

MOORE, Brian. **An introduction to the psychology of hearing**. 1. ed. Leiden, Netherlands: Brill, Apr. 2013. ISBN 90-04-25242-8.

MUELLER, Meinard. **Fundamentals of music processing: audio, analysis, algorithms, applications.** Cham, Switzerland: Springer International Publishing, Oct. 2016. ISBN 978-3-319-35765-2.

MUELLER, Meinard. **Fundamentals of music processing: using Python and Jupyter notebooks.** [S.l.]: Springer Nature, Aug. 2021. ISBN 978-3-030-69808-9.

MUSHTAQ, Zohaib; SU, Shun Feng. Efficient classification of environmental sounds through multiple features aggregation and data enhancement techniques for spectrogram images. **Symmetry**, v. 12, n. 11, p. 1–34, Nov. 2020. Publisher: MDPI AG. ISSN 20738994. DOI: 10.3390/sym12111822.

MUSHTAQ, Zohaib; SU, Shun-Feng. Environmental sound classification using a regularized deep convolutional neural network with data augmentation. **Applied Acoustics**, v. 167, p. 107389, 2020. ISSN 0003-682X. DOI: 10.1016/j.apacoust.2020.107389.

NEWCOMB, Doug. **Car Audio For Dummies**. 1. ed. Chichester, England: John Wiley & Sons, Jan. 2008. ISBN 978-0-470-15158-7.

NORDBY, Jon Opedal. **Environmental sound classification on microcontrollers using Convolutional Neural Networks**. 2019. PhD thesis – Norwegian University of Life Sciences, Ås, Oslo, Norway. Available from: http://hdl.handle.net/11250/2611624. Visited on: 7 Oct. 2023.

NORTH, Tesla. **Tesla Model Y HW4 Infotainment System: Lower Specs vs HW3 - TeslaNorth.com**. [S.l.: s.n.], Feb. 2009. Available from: https://teslanorth.com/2023/08/07/tesla-model-y-hw4-infotainment-system/.

OPPENHEIM, A V; SCHAFER, R W. From frequency to quefrency: a history of the cepstrum. **IEEE Signal Processing Magazine**, v. 21, n. 5, p. 95–106, 2004. DOI: 10.1109/MSP.2004.1328092.

PANDYA, Sharnil; GHAYVAT, Hemant. Ambient acoustic event assistive framework for identification, detection, and recognition of unknown acoustic events of a residence. **Advanced Engineering Informatics**, v. 47, p. 101238, 2021. ISSN 1474-0346. DOI: https://doi.org/10.1016/j.aei.2020.101238. Available from: https://www.sciencedirect.com/science/article/pii/S147403462030207X.

PARK, Daniel S et al. Specaugment: A simple data augmentation method for automatic speech recognition. In: PROCEEDINGS of the Annual Conference of the International Speech Communication Association, INTERSPEECH. [S.l.: s.n.], 2019. 2019-September, p. 2613–2617. DOI: 10.21437/Interspeech.2019-2680.

PARK, Tae Hong. **Introduction to digital signal processing: Computer musically speaking**. Singapore, Singapore: World Scientific Publishing, May 2008. ISBN 978-981-279-027-9.

PEETERS, Geoffroy. A large set of audio features for sound description (similarity and classification) in the CUIDADO project. **CUIDADO Ist Project Report**, v. 54, n. 0, p. 1–25, 2004. Available from: http://recherche.ircam.fr/anasyn/peeters/ARTICLES/Peeters_2003_cuidadoaudiofeatures.pdf. Visited on: 28 Jan. 2024.

PELGROM, Marcel. **Analog-to-Digital Conversion**. 3. ed. Cham, Switzerland: Springer International Publishing, June 2018. ISBN 978-3-319-44970-8.

PICZAK, Karol J. Environmental sound classification with convolutional neural networks. **2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)**, p. 1–6, 2015. DOI: 10.1109/MLSP.2015.7324337.

PICZAK, Karol J. **ESC: Dataset for Environmental Sound Classification**. [S.l.]: Harvard Dataverse, 2015. DOI: doi/10.7910/DVN/YDEPUT.

PREECE, Stephen et al. Activity Identification Using Body-Mounted Sensors — A Review of Classification Techniques. **Physiological measurement**, v. 30, r1–33, Jan. 2009. DOI: 10.1088/0967-3334/30/4/R01.

RASPBERRY PI. **Raspberry Pi 4 Model B**. [S.l.: s.n.], Jan. 2023. Available from: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/. Visited on: 5 Jan. 2024.

RAYBURN, Ray A; EARGLE, John. **Eargle's the microphone book**. 2. ed. Oxford, England: Focal Press, Apr. 2004. ISBN 978-0-240-51961-6.

REDDI, Sashank J; KALE, Satyen; KUMAR, Surinder. On the Convergence of Adam and Beyond. **ArXiv**, abs/1904.09237, 2018. Available from: https://api.semanticscholar.org/CorpusID:3455897.

ROADSTAR. **Roadstar - The Power of Sound**. [S.l.: s.n.], Jan. 2021. Available from: http://www.roadstarbrasil.com.br/rs120mic.php. Visited on: 6 Jan. 2024.

ROSENBLATT, F. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. **Psychological Review**, v. 65, n. 6, p. 386–408, 1958. DOI: 10.1037/h0042519.

ROSSING, Thomas D; MOORE, Richard F; WHEELER, Paul A. **The science of sound: Pearson new international edition**. 3. ed. London, England: Pearson Education, Nov. 2013. ISBN 978-1-292-03957-2.

ROTHMUND, Felix. **A Deep Learning Approach to Speech, Music and Environmental Noise Classification**. Mar. 2018. PhD thesis – University of Music and Performing Arts Graz, Graz, Austria.

RUMELHART, David E; HINTON, Geoffrey E; WILLIAMS, Ronald J. Learning representations by back-propagating errors. **Nature**, v. 323, p. 533–536, 1986. DOI: 10.1038/323533a0.

RUMREICH, Mark. **Car Stereo Cookbook**. 2. ed. New York, NY: McGraw-Hill Professional, Apr. 2005. ISBN 978-0-07-144847-5.

RUSSELL, Stuart; NORVIG, Peter. **Artificial intelligence: A Modern Approach**. 3. ed. Upper Saddle River, NJ: Pearson, Dec. 2010. ISBN 978-0-13-604259-4.

SAE. **Levels of Driving Automation Refined for Clarity and International Audience**. [S.l.: s.n.], Jan. 2021. Available from: https://www.sae.org/blog/sae-j3016-update. Visited on: 4 Jan. 2024.

SALAMON, Justin; BELLO, Juan Pablo. Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification. **IEEE Signal Processing Letters**, v. 24, n. 3, p. 279–283, 2017. DOI: 10.1109/LSP.2017.2657381.

SALAMON, Justin; JACOBY, Christopher; BELLO, Juan Pablo. A Dataset and Taxonomy for Urban Sound Research. In: PROCEEDINGS of the 22nd ACM international conference on Multimedia. New York, NY, USA: ACM, Nov. 2014. P. 1041–1044. ISBN 978-1-4503-3063-3. DOI: 10.1145/2647868.2655045.

SARAUBON, Kobkiat; ANURUGSA, Keattisuk; KONGSAKPAIBUL, Adichart. A Smart System for Elderly Care using IoT and Mobile Technologies. In: ICSEB 18. [S.l.: s.n.], 2018. DOI: 10.1145/3301761.3301769.

SARKAR, Dipayan; NATARAJAN, Vijayalakshmi. **Ensemble Machine Learning Cookbook**. 1. ed. Birmingham, England: Packt Publishing, Jan. 2019. ISBN 978-1-78913-660-9.

SCIKIT-LEARN. **scikit-learn: Machine Learning in Python**. [S.l.: s.n.], 2023. Available from: https://scikit-learn.org/stable/. Visited on: 23 Dec. 2023.

SEEED STUDIO. **ReSpeaker Mic Array v2.0 Seeed Studio Wiki**. [S.l.: s.n.], Jan. 2021. Available from: https://wiki.seeedstudio.com/ReSpeaker_Mic_Array_v2.0/. Visited on: 5 Jan. 2024.

SHABTAI, Noam R; TZIRKEL, Eli. Detecting the direction of emergency vehicle sirens with microphones. In: EAA Spatial Audio Signal Processing Symposium. Paris, France: [s.n.], Sept. 2019. P. 137–142. DOI: 10.25836/sasp.2019.22.

SHARMA, Jivitesh; GRANMO, Ole-Christoffer; GOODWIN, Morten. Emergency detection with environment sound using deep convolutional neural networks. In: PROCEEDINGS of Fifth International Congress on Information and Communication Technology. Singapore: Springer Singapore, 2021. P. 144–154. ISBN 978-981-15-5859-7. DOI: DOI:10.1007/978-981-15-5859-7_14.

SHREYAS, N et al. Chapter 7 - Trends of Sound Event Recognition in Audio Surveillance: A Recent Review and Study. In: PETER, Dinesh et al. (Eds.). **The Cognitive Approach in Cloud Computing and Internet of Things Technologies for Surveillance Tracking Systems**. [S.l.]: Academic Press, 2020. Series Title: Intelligent Data-Centric Systems. P. 95–106. ISBN 978-0-12-816385-6. DOI: https://doi.org/10.1016/B978-0-12-816385-6.00007-6. Available from: https://www.sciencedirect.com/science/article/pii/B9780128163856000076.

SILVA, Bruno da et al. Evaluation of classical Machine Learning techniques towards urban sound recognition on embedded systems. **Applied Sciences (Switzerland)**, v. 9, n. 18, p. 3885, Sept. 2019. DOI: 10.3390/app9183885. Visited on: 17 Oct. 2022.

SMITH, Steven. **Digital signal processing: A practical guide for engineers and scientists**. [S.l.]: Elsevier, Oct. 2013. ISBN 978-0-08-047732-9.

SU, Y. et al. Environment sound classification using a two-stream CNN based on decision-level fusion. **Sensors (Switzerland)**, v. 19, n. 7, 2019. DOI: 10.3390/s19071733.

SU, Yu et al. Performance analysis of multiple aggregated acoustic features for environment sound classification. **Applied Acoustics**, v. 158, 2020. DOI: 10.1016/j.apacoust.2019.107050.

SUN, Hongyi et al. Emergency Vehicles Audio Detection and Localization in Autonomous Driving. **ArXiv**, abs/2109.14797, 2021. arXiv: 238226804.

TANG, Guichen et al. Improved Convolutional Neural Networks for Acoustic Event Classification. **Multimedia Tools and Applications**, v. 78, p. 15801–15816, 2018. DOI: 10.1007/s11042-018-6991-4.

TEAM AUDACITY. **Audacity® (Version 3.1.2)**. [S.l.: s.n.], Jan. 2024. Available from: https://www.audacityteam.org/.

TENSORFLOW. **TensorFlow**. [S.l.: s.n.], 2023. Available from: https://www.tensorflow.org/. Visited on: 23 Dec. 2023.

TRAN, Van-Thuan; TSAI, Wei-Ho. Acoustic-Based Emergency Vehicle Detection Using Convolutional Neural Networks. **IEEE Access**, v. 8, p. 75702–75713, 2020. DOI: 10.1109/ACCESS.2020.2988986.

TRIPATHI, Achyut Mani; MISHRA, Aakansha. Self-supervised learning for Environmental Sound Classification. **Applied Acoustics**, v. 182, p. 108183, 2021. ISSN 0003-682X. DOI: https://doi.org/10.1016/j.apacoust.2021.108183.

VANDENDRIESSCHE, Jurgen et al. Environmental Sound Recognition on Embedded Systems: From FPGAs to TPUs. **Electronics (Switzerland)**, v. 10, n. 21, p. 1–31, 2021. DOI: 10.3390/electronics.

VEERARAGHAVAN, A. K.; RANGA CHARAN, S. SoC Based Acoustic Controlled Semi Autonomous Driving System. In: PANDIAN, A. Pasumpon et al. (Eds.). **Proceeding of the International Conference on Computer Networks, Big Data and IoT (ICCBI - 2018)**. Cham: Springer International Publishing, 2020. P. 591–599. ISBN 978-3-030-24642-6. DOI: 10.1007/978-3-030-24643-3_71. Available from: http://link.springer.com/10.1007/978-3-030-24643-3_71. Visited on: 23 Mar. 2023.

VIDAÑA-VILA, Ester et al. Low-Cost Distributed Acoustic Sensor Network for Real-Time Urban Sound Monitoring. **Electronics**, v. 9, n. 12, 2020. ISSN 2079-9292. DOI: 10.3390/electronics9122119. Visited on: 6 Jan. 2024.

WANG, Deliang; BROWN, Guy J. **Computational auditory scene analysis: principles, algorithms, and applications**. Ed. by Deliang Wang and Guy J Brown. 1. ed. Nashville, TN: John Wiley & Sons, Sept. 2006. ISBN 978-0-471-74109-1.

WAYMO. **The Waymo Driver's Rapid Learning Curve**. [S.l.: s.n.], Jan. 2023. Available from: https://waymo.com/blog/2023/08/the-waymo-drivers-rapid-learning-curve/. Visited on: 21 Jan. 2024.

WICKRAMASINGHE, Indika; KALUTARAGE, Harsha. Naive Bayes: applications, variations and vulnerabilities: a review of literature with code snippets for implementation. **Soft Computing**, v. 25, n. 3, p. 2277–2293, Feb. 2021. DOI: 10.1007/s00500-020-05297-6.

YIN, J et al. Real-Time Acoustic Perception for Automotive Applications. In: 2023 Design, Automation & Test in Europe Conference & Exhibition (DATE). [S.l.: s.n.], 2023. P. 1–6. DOI: 10.23919/DATE56975.2023.10137209.

ZHANG, Tong; C. JAY KUO, C. **Content-based audio classification and retrieval for audiovisual data parsing**. 1. ed. New York, NY: Springer, Dec. 2010. Series Title: The Springer International Series in Engineering and Computer Science. ISBN 978-1-4419-4878-6.

ZHU, Hailong et al. Requirements-Driven Automotive Electrical/Electronic Architecture: A Survey and Prospective Trends. **IEEE Access**, v. 9, p. 100096–100112, 2021. DOI: 10.1109/ACCESS.2021.3093077.

ZOELZER, Udo. **Digital Audio Signal Processing**. Chichester, UK: Wiley, June 2008. ISBN 978-0-470-99785-7.

APPENDIX A – SYSTEMATIC REVIEW METHODOLOGY

A systematic review following a rigorous and well-structured approach to summarizing and synthesizing the existing research was performed according the methodology below. The first step was a search considering the following parameters:

a) **Search query**: “sound AND recognition AND embedded”, “sound AND classification AND embedded”, “audio AND scene AND classification”, “audio AND scene AND recognition”, “auditory AND scene AND classification”, and “auditory AND scene AND recognition”;
b) **Knowledge databases**: *sciencedirect.com*, *scopus.com*, *webofscience.com*, *ieeexplore.ieee.org*, and *semanticscholar.org*;
c) **Exclusion criterion**: none.

In total, 2.629 documents were selected from the search query:

a) 384 papers at IEEE Xplore;
b) 341 papers at Semantics Scholar;
c) 450 papers at Web of Science;
d) 863 papers at ScienceDirect;
e) 591 papers at Scopus.

To start the screening and selection process, the bibliographic information of these documents was processed using a specialized software named VOSviewer for the purpose of establishing relationships through bibliographic coupling, co-citation, co-authorship, and co-occurrence of keywords thus creating a dynamic and interactive bibliometric networks with multiple layers (Figure 50) that allowed to analyze the strength between the nodes connecting each other as well the relationship between the keywords, authors and co-authors. In the end of this process, 143 documents were selected.

For each identified document, a corresponding entry was registered in the software Zotero, enabling the analysis of the subsequent metrics:

a) Number of explicit citations;
b) Number of supportive citations;
c) Number of of contrasting citation.

When the metrics were favorable and the main objective of the selected document was related to the main objective of this study, the data extraction process began using the software Obsidian (Figure 51) to save the following information in a comprehensive report for the systematic review, adhering to the preferred reporting guidelines established in Preferred Reporting Items for Systematic Reviews (PRISMA):

a) Document metadata;

b) Keywords, citation and references;

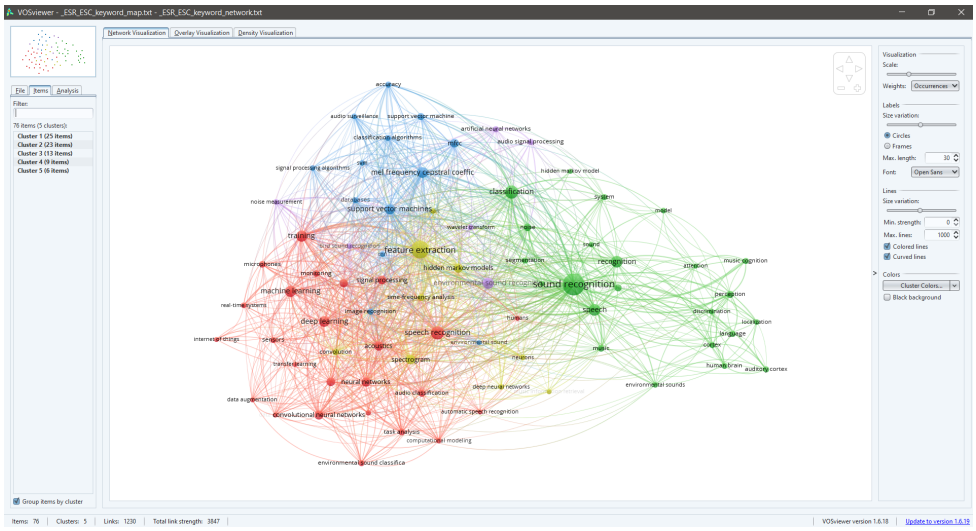
c) The need for the research;

d) The presented methodology;

e) The final result and future works;

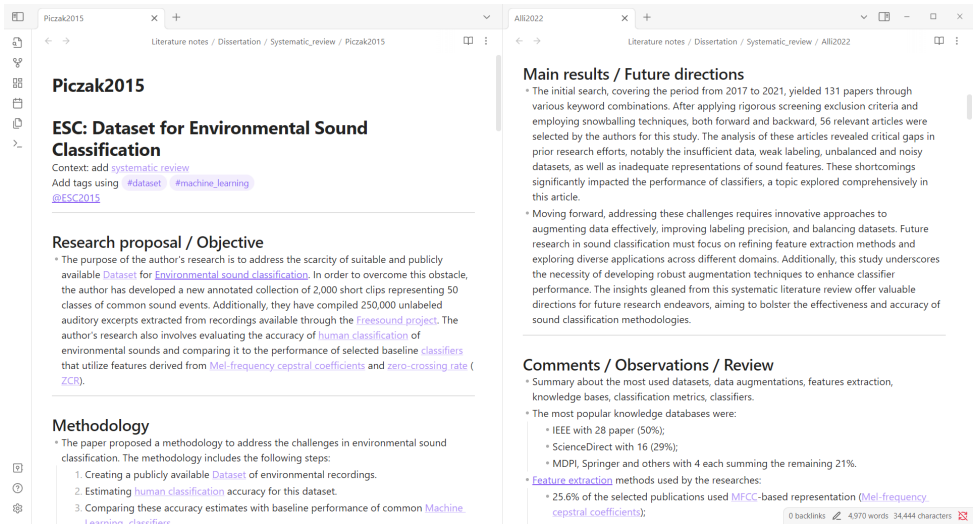
f) Personal comments and remarks.

Figure 50 – Visualization of bibliometric networks.



Source: Author

Figure 51 – File structure in Obsidian for document data extraction.



Source: Author

At the end of this stage, the Obsidian database contained 64 documents. After analyzing the information, 42 documents were selected as relevant. A detailed study of these documents was conducted, and the snowballing process was employed to identify additional relevant documents to complement the references.

Furthermore, references commonly used in the field of acoustics, machine learning and neural networks, particularly those with a high number of citations, were not included in the aforementioned process.