Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell’Ingegneria
Curriculum Ingegneria Biomedica, Elettronica e delle Telecomunicazioni

Deep Learning for Audio Signal Processing in the Automotive Field

Ph.D. Dissertation of:
Michela Cantarini

Advisor:
Prof. Stefano Squartini

Curriculum Supervisor:
Prof. Franco Chiaraluce

XXXV edition - new series

Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell’Ingegneria
Curriculum Ingegneria Biomedica, Elettronica e delle Telecomunicazioni

Deep Learning for Audio Signal Processing in the Automotive Field

Ph.D. Dissertation of:
Michela Cantarini

Advisor:
Prof. Stefano Squartini

Curriculum Supervisor:
Prof. Franco Chiaraluce

XXXV edition - new series

Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Facoltà di Ingegneria
Via Brecce Bianche – 60131 Ancona (AN), Italy

To the A3Lab Group

Abstract

Over the last few years, the automotive industry has directed its research toward intelligent vehicles that come with sophisticated driver-assistance technologies to enhance the safety of drivers and passengers. These systems are designed to match the human ability to perceive and react to the surrounding environment in order to help drivers make better decisions, all the way to the ultimate goal of autonomous driving. A central factor in human perception is hearing, and deep learning applied to audio signal processing has developed computational models that can detect and identify sounds in the environment around the car. This research explores the potential of intelligent monitoring systems for advanced human-vehicle interaction solutions, specifically focusing on emergency vehicle detection systems for smart cars. As a first approach, an algorithm is proposed for generating synthetic audio files to reproduce siren sounds in multiple noise contexts, which, balanced with urban traffic noises, are used to train a convolutional neural network for siren/noise classification. Several acoustic features, source separation techniques, and strategies to reduce the computational load of the algorithm are studied to identify siren sounds even in loud background noise. This research also presents a workflow based on few-shot metric learning for emergency siren detection, which uses prototypical networks to recognize ambulance sirens without requiring extensive real-world data collection or domain adaptation strategies. A novel prototype of driver-assistant for emergency-vehicles detection is also proposed. This device uses audio-based deep learning algorithms to detect an emergency vehicle approaching through the sound of its siren and computer vision techniques to monitor the driver's attention through gaze movements. The innovation lies in the alerting based on the driver's awareness, which limits warnings only to situations of actual need. Finally, an audio-visual dataset for driving scene understanding is presented. Data are representative of different types of roads, urbanization contexts, weather and lighting conditions. This dataset is a valuable instrument for developing driver-assistance technologies that rely on audio and video data in single-modality or multimodality and for improving the performance of systems currently in use. This research shows that the topic of emergency siren detection and, in general, ambient intelligence in the automotive field still has great potential for innovation in terms of reliable, customizable, and cost-effective solutions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, automotive research has shown a growing interest in safety technologies for drivers and passengers, developing intelligent vehicles with increasing levels of automation through advanced driver-assistance systems (ADASs) [1, 2]. These solutions form the basis of autonomous vehicles, in which different types of sensors replace the human ability to sense the environment and react in real time [3–5]. Hearing is a primary factor in human driving, and deep learning applied to audio signal processing has enabled technologies to "listen" to sounds, understand them and respond accordingly. The process of automatically detecting and identifying sound events occurring in the surroundings of the car alerts drivers in case of distractions, allowing them to make better decisions and thus preventing traffic accidents.

Recent advances in deep learning have enabled the development of powerful models that can leverage information from audio data to help recognize potentially dangerous situations, as illustrated in several case studies. The identification and localization of static or moving sound sources like automobiles, bicyclists, or pedestrians in environments with narrow, confined, or diffuse obstacles, such as alleys in historic villages or densely built-up or vegetation-rich areas, can be carried out by directional acoustic detection techniques. These methodologies are particularly useful in the case of approaching vehicles around blind corners that can be detected with acoustic sensors before they enter the driver's line of sight [6]. Some weather conditions are more accurately represented by audio than visual data, especially when the distinction between atmospheric events is subtle (e.g., diffuse moisture or light rain, medium or high-intensity rain) and as daylight becomes weak. For this reason, the intensity of rainfall on different materials can be estimated through audio data [7] to activate the windshield wipers and control their speed automatically. Similarly, audio-based road wetness detection systems could be employed to alert the driver or change the tire setup [8, 9]. Classification of roughness and degree of deterioration of road pavement [10, 11], estimation of speed and density of vehicular traffic [12, 13] are other topics related to safe driving scenarios that audio research has successfully developed.

A key challenge is the reliable and accurate detection of emergency sirens in traffic noise to alert drivers of the approach of an emergency vehicle and enable them to prioritize it in rescue operations. For this reason, emergency vehicle detection systems installed in automobiles are becoming indispensable safety devices, and research in this area continues to be an active field of study.

1.1 Emergency Vehicle Detection Systems

Emergency-vehicles detection has been an ongoing research topic since the 1950s [14], resulting in the development of several models of emergency vehicle detection (EVD) systems that have evolved along with sensing technologies. EVD systems have upgraded from basic electronic devices to advanced digital systems, with the main goal of helping emergency vehicles reach their destination more quickly and safely. The proliferation of daily automobile use has led to increased urban traffic levels and travel times, causing drivers to engage in activities that divert their attention from driving, such as listening to music or making business calls. The modern car, equipped with various amenities, soundproof cabins and high-performance sound systems, also creates an isolated environment where drivers may not be aware of an incoming emergency vehicle with due timeliness. For this reason, EVD systems demonstrate their usefulness in several installation contexts. When integrated within the car, these devices alert drivers of the approach of an emergency vehicle through warning signals, benefiting people with hearing impairments. When installed at traffic lights or intersections, they activate reserved lanes.

In the literature, the identification of emergency vehicles has been performed with several technologies, as evidenced by the wide variety of related patents. For example, radio frequency-based detectors have been developed using transmitter units installed in emergency vehicles and receiver units placed in cars or intersections [15, 16]. Other systems employ sensors to detect electromagnetic data emitted by flashing lights and siren sounds, then transmitted to computer systems to recognize specific siren patterns [17]. In other devices, GPS is used to track the location of emergency vehicles and provide the driver with their real-time position [18]. In addition, image analysis technologies based on computer vision have been implemented [19, 20], also integrated with sound processing systems to detect the presence of emergency vehicles out of sight [21].

Audio data processing and analysis have always played an important role in detecting emergency vehicles, recognizable by the siren sound emitted through embedded electronic devices. For this reason, emergency siren detection (ESD) is the main technique behind EVD systems. In the 1960s and 1970s, early audio-based EVD systems used electrical circuits equipped with analog filters to select and amplify the sound recorded with external microphones in the range of siren frequencies [22]. Similar devices, in combination with frequency-voltage converters, have been designed to detect slow and continuous variations in the siren signal [23]. Since the 1980s and 1990s, more advanced emergency siren detectors using digital signal processing techniques have been developed. These types of equipment convert audio signals into discrete time-frequency representations. Then, the match with the frequencies of the alarm signal or the number of peaks detected in a certain period of time after a band-pass filtering process determines the presence of the siren sound [24, 25]. Other devices include a sound generator installed on the emergency vehicle and a detection unit on common vehicles. In these systems, the acoustic signal is transduced into an electric current that is compared with pre-programmed patterns, so any match is notified to the driver through a display [26]. Emergency-vehicles detectors are becoming increasingly sophisticated today, using technologies such as machine and deep learning to detect and classify different types of emergency vehicles. Fully audio-based EVD systems include sound acquisition using microphones, audio signal segmentation, and computation of spectrograms that are given as input into pre-trained neural networks [27]. More complex models employ data fusion techniques to compute and concatenate audio-visual features into a single feature vector containing information about the presence of an emergency vehicle [28]. Some systems also perform the localization of the emergency vehicle using laser imaging detection and ranging (LiDAR), RADAR or ultrasonic data [29].

Methods for detecting emergency sirens can be classified according to algorithmic techniques correlated to the type and amount of data required for their implementation. The most commonly used detection strategies include digital signal processing, machine learning and deep learning algorithms. Artificial neural network-based methodologies that exploit a large amount of audio or multimodal data to compute a model under fully supervised conditions are currently the most widely used approaches. Most studies have addressed the problem of the massive volume of data needed to train a neural model by using datasets computed from synthetic audio collections or siren recordings from publicly available web resources. However, data generated via algorithm may not accurately simulate real-world situations, and data acquired with different devices require standardization procedures.

In addition, the quality and quantity of data are crucial for the implementation of ESD strategies working in real time. Synthetic data must accurately mimic the environmental and operational noises of the vehicle where the recording sensors are installed. Alternatively, real-world audio files collected in the same context as the siren detection device should have a duration of several hours to ensure the reliability and generalization capability of the neural model. The ideal situation in terms of data availability is a combination of the above.

A significant dataset in terms of quantity and quality of data should be used to train the deep learning model, which is then adapted with fewer examples to recognize siren sounds and noises in real-world environments. In this sort of context, data augmentation strategies applied directly to the raw signal (e.g., noise addition, distortion, or velocity scaling) or to the time-frequency representation (e.g., pitch or time-shifting) [30, 31] of on-board recordings have the disadvantage of altering the target signal or background, making the use of synthetic data preferable. Given these premises, the main goals of research on emergency siren detection are the generation procedures and recording techniques of audio data, which, together with the acoustic features investigation, enable the development of accurate and customizable detection strategies in real-world driving scenarios.

1.2 State-Of-The-Art

Many researchers have devoted considerable attention to developing algorithms for emergency siren recognition through digital signal processing techniques. These approaches, which involve the manipulation and analysis of discrete digital signals to extract useful information, have been used alone or in combination with other advanced approaches such as machine and deep learning. In particular, deep neural networks have improved the performance of siren recognition algorithms by providing models that can learn the task, adapt to new data and prove robust to variations in the real-world environment. The following is a summary of the most significant recent work in the emergency siren detection field, distinguished by algorithmic methodologies, with emphasis on the datasets employed and the findings achieved.

1.2.1 Digital Signal Processing Approaches

The detection and recognition of emergency sirens using digital signal processing techniques have been studied by several researchers. Different approaches such as pitch detection, two-times Fast Fourier Transform, longest common subsequence, peak detection and minimum mean square error methods have been proposed and implemented on low-power microprocessors and microcontrollers, with varying results in terms of detection time, false alarm rate and missing siren signal.

Meucci *et al.* [32] developed a pitch detection algorithm based on the module difference function and peak searching to extract periodic siren signals from aperiodic ones. The algorithm was implemented on a low-power microprocessor, and performance was evaluated both on real signals recorded in city streets and digitally synthesized signals at different signal-to-noise ratios (SNRs). The performance of the detector was analyzed by varying several parameters, obtaining probability rates from approximately 46% to 98% in the range of SNRs between -15 dB and 10 dB. Miyazaki *et al.* [33] used a two-times Fast Fourier Transform algorithm for siren detection and programmed it on a microcontroller. The authors tested the system on pure siren sound, noise, and siren mixed with noise, also considering the Doppler effect, obtaining an average detection time of about 8 seconds at SNR equal to 0 dB. Liaw *et al.* [34] proposed the longest common subsequence method to recognize the sound of an ambulance siren in Taiwan. The approach was applied to compare the sequence of the input sound, consisting of background noise and music partially overlapped with siren sounds and the sequence of the ambulance siren, achieving a true positive rate of 85%. Kiran *et al.* [35] used a peak detection algorithm coupled with the minimum mean square error method to detect acoustic siren signals. The workflow of the system consisted of real-time audio capturing, segmentation into sequential frames, application of band-pass filtering, spectral analysis, and peak searching in the frequency domain. Automatic detection of the emergency vehicle siren was performed by measuring the number of peaks at the siren frequencies present in the audio frames, using in testing a mix of pure siren signals, sirens immersed in different background noises, and only noise. The resulting achievement was an autocorrelation time in the siren detection process of about 7.9 seconds. Other recent studies about siren recognition employed digital signal processing techniques based on frequency and chronological data [36] or statistical methods [37].

1.2.2 Machine and Deep Learning Approaches

Recent studies have developed algorithms to identify emergency siren sounds through machine and deep learning approaches. These works used a variety of features and classifiers, including hidden Markov models, part-based models, and support vector machines. In particular, artificial neural networks have achieved the highest accuracy rates in detecting emergency sirens in different recording configurations and noise conditions. Some studies proposed frameworks for classifying and localizing alerting sound events, and promising results have been obtained using a combination of audio and video-based detection systems.

Beritelli *et al.* [38] developed an algorithm for identifying emergency sirens through speech recognition techniques. The authors used a multi-layer artificial neural network to classify Mel-Frequency Cepstral Coefficients (MFCCs) extracted from siren sounds of Italian emergency service vehicles under different recording configurations and additive noise at increasing signal-to-noise ratios (SNRs). The algorithm achieved an accuracy rate greater than 99% with a response time of fewer than 400 milliseconds. Schroder *et al.* [39] proposed a hidden Markov model and a part-based model for detecting police siren sound in clean and noisy environments. The authors used MFCCs and Mel-spectrograms as acoustic features and tested the classifiers at different SNRs. The part-based model classifier was the most accurate, achieving an accuracy rate of 86% at an SNR of -10 dB. Carmel *et al.* [40] presented a technique for detecting alarm sounds in noisy environments using support vector machine classifiers. The authors employed several time-domain and frequency-domain features, such as Pitch, Short-Time Energy, Zero-Crossing Rate, MFCCs, Spectral Flux, Discrete Wavelet Transform, and Wavelet Packet Transform, achieving an accuracy rate equal to 98% per 100 milliseconds audio frame. Marchegiani *et al.* [41] proposed a framework for classifying and localizing alerting sound events using a U-Net [42] architecture. Semantic segmentation was applied to gammatone spectrograms in a multi-task learning scheme for acoustic event classification. The experiments concerning the classification task reported an average accuracy rate of 94% at SNRs between -40 dB and 10 dB. Other studies based on artificial neural networks compared several acoustic features [43], implemented the siren detection algorithm in mobile apps [44, 45] and developed hybrid audio-video detection systems [46]. Fatimah *et al.* [47] extracted two sets of features from the ambulance siren sound. The ensemble bagged trees classifier with the Fourier decomposition method obtained the best performance with an accuracy of 98.49%.

Tran *et al.* [48] illustrated a study for the classification of siren sounds, vehicle horns, and noise. Three models based on convolutional neural networks (CNNs) were developed: the first combined MFCCs and log-Mel features in a 2D-CNN (MLNet), the second implemented a 1D-CNN (WaveNet) which automatically learned the features for classification from raw waveform, and at last, a CNN-based ensemble model (SirenNet) was designed combining the previously described networks. The experiments showed that SirenNet achieved an accuracy of 98.24% in the siren sound detection with frames of 1.5 seconds. Tran *et al.* [49] recently presented another study based on audio and image data. The authors devised a modified YOLO [50] model called YOLO-EVD tailored to the problem of emergency vehicle detection. This model, trained with a novel dataset for vision-based EVD, obtained a mean average precision of 95.5%. Additionally, a convolutional neural network called WaveResNet was implemented for audio-based EVD, which reached an accuracy of 98% in traffic conditions. The integration of the two models formed an audio-visual EVD system (AV-EVD) with a siren misdetection rate of 1.54%.

1.3 Motivations and Contributions

This research endeavors to broaden the potential of intelligent monitoring systems for advanced human-machine interaction technologies in the living environment of the automobile. The primary objective is to investigate the field of emergency siren recognition, which falls under the wider scope of sound-event detection methodologies embedded within emergency vehicle detection systems. The literature on emergency siren detection has presented several challenges, which have been addressed through digital signal processing, machine learning, and deep learning algorithms. Among these, convolutional neural networks that learn time-frequency representations of audio signals segmented in short-time frames have been shown to be the most effective approach for accurate siren detection in noisy scenarios and in the presence of the Doppler effect. This study takes as its starting point the best state-of-the-art findings, then improves upon them by investigating several datasets, neural approaches, and architectures to accurately identify emergency sirens in real-world environments. This research culminates in an advanced driver-assistance prototype for emergency-vehicles detection that relies on multimodal data analyzed with deep learning algorithms.

The main contribution of this work is the implementation of strategies for emergency siren detection that are reliable, easily customizable in different vehicles and cost-effective, demonstrating that the research area still holds great scope for innovations. The first objective is to create a neural model capable of identifying the siren signal in different environmental driving contexts and generalizing it to ever-changing and unpredictable background noises. In this regard, an algorithm is proposed to generate synthetic audio files that reproduce the sound of sirens in multiple traffic contexts. The siren audio segments equally balanced with traffic noises and transformed from the amplitude to the time-frequency domain constitute a dataset to train a convolutional neural network with a supervised approach for siren/noise classification. The selection and design of acoustic features suitable for the task require considerable expertise on the problem and constitute a significant engineering effort. Therefore, the accuracy in identifying sirens with several acoustic features is investigated and compared, highlighting their capability to effectively represent the siren signal in high background noise contexts and generalize to scenarios unseen during the training.

The second objective aims at the reduction of the computational load of the algorithm. For this purpose, the investigation with synthetic data has been extended to short-time Fourier transform spectrograms as features. A harmonic-percussive source separation technique has also been applied to improve siren detection accuracy. The decreased number of network hyperparameters, also performed by slicing operations on the time-frequency representations, reduces the computational load, making it suitable for real-time embedded systems.

In the third phase, the goal of developing an EVD system to be installed in vehicles leads to testing the previously computed neural models on real data. At this point, the problem of emergency siren detection is approached with a different perspective: to find a technique able to accurately identify siren sounds without the need for adaptation between source and target domains and without requiring a large collection of real-world data for training a supervised deep learning model. A workflow based on few-shot metric learning for emergency siren detection is proposed, in which prototypical networks [51] are trained on publicly available sources or synthetic data in multiple combinations. At inference time, the best knowledge learned in associating a sound with its class representation is transferred to identify ambulance sirens, given only a few instances for its prototype computation. The encouraging results confirm the robustness of meta-learning approaches for real-world applications.

Finally, algorithmic investigations are put into practice through the design of an advanced multimodal driving-assistance prototype for emergency-vehicles detection. This system leverages audio and video deep learning algorithms to detect the approaching of an emergency vehicle through the sound of its siren and, subsequently, to monitor the driver's gazer to check his/her awareness. If the driver demonstrates to be unaware of the situation, the system alerts attention with an audio-visual warning.

This research also laid the foundation for undertaking a study on driving scene understanding. For this purpose, a collection of audio and video data acquired in diverse driving scenarios on board a sensor-equipped research vehicle is presented. This carefully recorded, processed, and labeled multimodal dataset represents the first fundamental step in the development and improvement of ADASs designed to understand the environment surrounding the car.

The outline of this dissertation is the following. In Chapter 1, the emergency siren detection topic is introduced, followed by a state-of-the-art mainly focused on the audio-based approaches and a summary of motivations and contributions of this work. Chapter 2 gives an overview of the theoretical background of the data-driven techniques and the performance metrics used to develop the presented systems. Chapter 3 describes the methodologies to collect the datasets used in the experiments. Chapter 4 discusses the approaches for emergency siren detection based on convolutional neural networks in combination with synthetic data. Meta-learning approaches are described in Chapter 5, where real-world siren data have been employed to test the algorithms. Chapter 6 outlines an advanced multimodal driver-assistance prototype for emergency-vehicles detection, and Chapter 7 presents an audio-visual dataset for driving scene understanding.

Chapter 2

Background

The emergence of the Internet of Things (IoT) [52] has led to the spread of advanced devices that connect to cloud computing systems or perform complex computations in vehicles, homes and cities. These technologies have been developed through data-driven algorithms able to make decisions without human intervention and replicate the human thought process, enabling computers to perform previously unimaginable tasks. Through artificial neural networks, deep learning algorithms learn multiple levels of representation and abstraction to analyze images, sounds and text. In audio signal processing, deep learning techniques are used to detect and recognize patterns of real-world sounds. This chapter provides the historical background, the theoretical description, and the metrics to evaluate the performance of the main deep neural network architectures employed in this research.

2.1 Deep Learning: Main Historical Events

In 1943, McCulloch and Pitts [53] developed a computational model for neural networks that employed mathematical principles and algorithms. They referred to this model as threshold logic, which laid the foundation for dividing neural network research into two distinct paths. One path focused on the biological workings of the brain, while the other concentrated on the use of neural networks in artificial intelligence. In the late 1940s, psychologist Donald Hebb proposed a theory of learning based on the concept of neural plasticity, now known as Hebbian learning [54]. This theory is commonly regarded as a form of unsupervised learning, and its subsequent developments were early models for long-term potentiation. In 1950, these concepts were implemented in computational models through Turing's B-type machines [55].

In 1958, Rosenblatt [56] developed the perceptron, an algorithm for identifying patterns using a two-layer learning computer network that employed basic mathematical operations of addition and subtraction. Additionally, Rosenblatt used mathematical notation to describe circuitry that was not present in the fundamental perceptron, such as the exclusive-or circuit. At about the same time, Widrow and Hoff developed a single-layer linear network and associated learning rule called ADALINE (Adaptive Linear Neuron) [57]. This network was used to implement adaptive filters, which are still actively used today. After the release of a study on machine learning by Minsky and Papert in 1969 [58], progress in the field of neural networks slowed down. The authors identified two major challenges with the technology at the time. One of these was that single-layer neural networks were unable to process the exclusive-or circuit, and the other was that computers were not advanced enough to handle the extensive computational demands of large neural networks. As a result, research in this area slowed until computers improved in terms of processing power. An important development that helped spur progress was the back-propagation algorithm introduced by Werbos in 1975 [59], effectively resolving the exclusive-or problem.

In 1980, several events caused renewed interest. Kunihiko Fukushima, working on computer vision, started developing the Neocognitron [60], a hierarchical and multilayered neural network. This design was the first deep learning model using a convolutional neural network. Fukushima's design helped the computers learn to recognize and identify visual patterns and also allowed for the fine-tuning of significant features by manually adjusting the weight of the desired connections. Kohonen made many contributions to the field of artificial neural networks, also called Self-Organizing Maps [61]. In 1982, Hopfield described the recurrent artificial neural network as a content-addressable memory system [62]. His works persuaded hundreds of highly qualified scientists, mathematicians, and technologists to join the emerging field of neural networks.

In the mid-1980s, a method of processing information simultaneously across multiple systems gained recognition as connectionism. Rumelhart and McClelland [63] published a comprehensive study on utilizing this technique in computer systems to replicate neural functions. The application of back-propagation was first realized in a practical sense through the work of Yann LeCun in 1989 at Bell Labs. He used convolutional networks in conjunction with back-propagation to classify handwritten digits [64, 65], and this system was later employed to process an abundant amount of handwritten checks in the United States. Despite initial excitement, the use of artificial neural networks faced challenges due to the limited resources available with early computer processors. Some important advances, such as long short-term memory [66] and bidirectional recurrent neural networks [67], went mostly unnoticed until later years.

In the mid-2000s, advancements in technology, such as graphics processing units (GPUs) and distributed computing, allowed for the widespread deployment of neural networks, particularly in areas such as image and visual recognition. This technological progress led to the development of the field known as

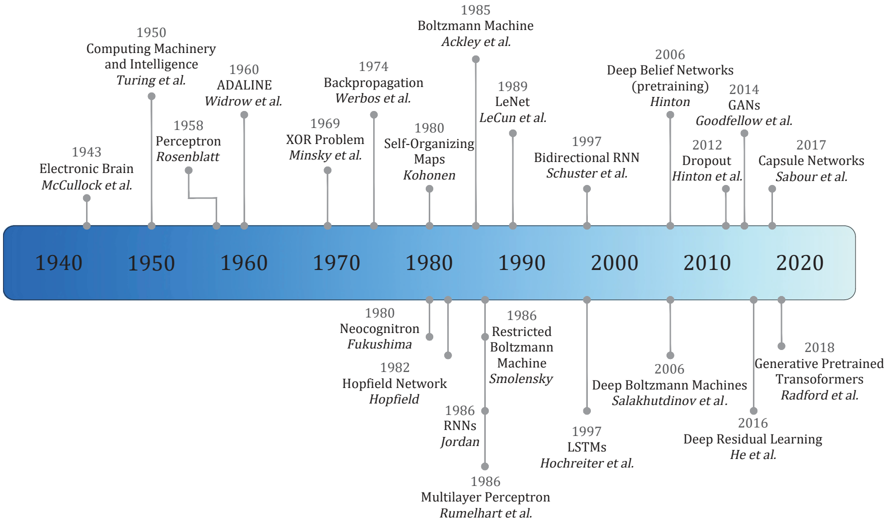


Figure 2.1: Historical timeline of deep learning evolution.

deep learning. Since then, advances in deep neural networks have not stopped evolving. Recent progress in statistical models, applications, and algorithms has resolved issues related to the performance of neural models with layer-by-layer pre-training methodologies [68] and later with deep residual learning [69]. Novel methods for capacity control, such as dropout [70], have been developed to mitigate overfitting, or attention mechanisms [71] solve the issue of increasing the memory and complexity of a system without increasing the number of learnable parameters. Built solely on attention mechanisms, the transformer architecture [72] has demonstrated compelling success in many areas. Another key development was the ability to generate realistic data through the invention of generative adversarial networks [73].

The timeline in Figure 2.1 summarizes the main historical events that have contributed to the evolution of deep learning.

2.2 Artificial Neural Networks

The human brain is considered the most advanced system of its kind because of its ability to interpret and analyze information. It is composed of specialized cells called neurons. Neurons are the structural and functional units of the nervous system, responsible for storing and processing information and being able to control various bodily functions. Artificial neural networks, on the other hand, are engineered systems inspired by the biological brain or "Massively parallel distributed processors made up of simple processing units having a natural propensity for storing experiential knowledge and making it available for use" [74].

The basic principles of operation and parallels between biological and artificial neurons with corresponding mathematical formulations are illustrated as follows.

2.2.1 The Human Nervous System

The biological neuron is composed of four main parts:

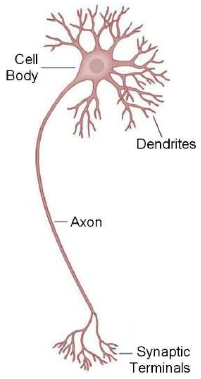


Figure 2.2: The biological neuron.

- Dendrites: input terminals that receive electric pulses from adjoining neurons. Pulses are weighted by the synaptic input connections.
- Cell Body or Nucleus: processes the incoming electric pulses and generates output spikes based on a threshold criterion.
- Axon: carries the output pulse towards the synaptic terminals, which are connected to subsequent neurons.
- Synapses: output terminals that assign a weight to a particular input.

The neuron properties can be described in:

- Local simplicity: the neuron receives stimuli (excitation or inhibition) from dendrites and produces an impulse to the axon, which is proportional to the weighted sum of the inputs.
- Global complexity: the human brain possesses $\mathcal{O}(10^{10})$ neurons, with more than $10k$ connections each.
- Learning: even though the network topology is relatively fixed, the strength of connections (synaptic weights) can change when the network is exposed to external stimuli.
- Distributed control: no centralized control; each neuron reacts only to its own stimuli.
- Tolerance to failures: performance slowly decreases with the increase of failures.

Biological neural networks can quickly solve complex tasks, such as memorization, recognition, and association.

2.2.2 Fundamentals of Artificial Neural Networks

An artificial neural network (ANN) is a mathematical model based on calculations inspired by biological neural networks. This model resembles the brain in two aspects: knowledge is acquired by the network from its environment through a learning process, and synaptic weights are used to store the acquired knowledge. Artificial neural networks are constituted by groups of interconnected information consisting of artificial neurons and processes using a connectionist approach to computation. In most cases, an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. In practical terms, artificial neural networks are non-linear statistical data structures organized as modeling tools. They can be used to simulate the complex relationships between inputs and outputs that other analytic functions fail to represent. An ANN receives external signals on an input layer of nodes (or processing units), each connected with some internal nodes organized in several levels. Each node processes the received signals performing a very simple task and transmits the result to subsequent nodes.

Artificial neurons are individual information-processing units representing the building blocks of artificial neural networks. Specifically, the model of a neuron is composed of four basic elements, as shown in Figure 2.3:

- A set of synapses, each characterized by a weight or strength of its own.
- The neural model also includes an externally applied bias.
- An adder for summing the input signals, weighted by the respective synaptic strengths of the neuron plus the bias; the operations described here constitute a linear combiner.
- An activation function for limiting the amplitude of the output of a neuron. Typically, the normalized amplitude range of the output of a neuron is written as the closed unit interval [0,1] or, alternatively, [-1,1].

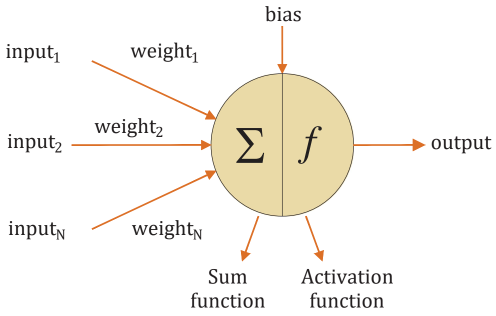


Figure 2.3: Model of an artificial neuron.

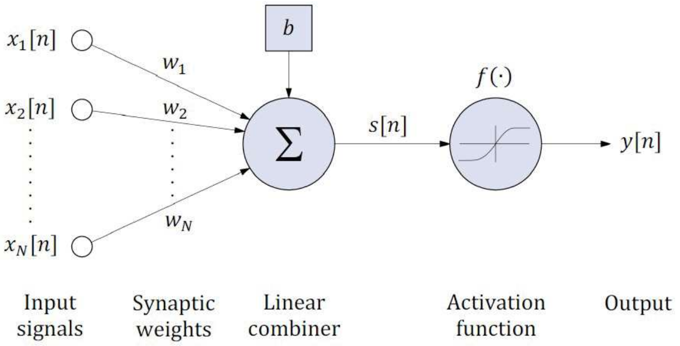


Figure 2.4: Mathematical description of an artificial neuron.

The mathematical description of the neuron activity, illustrated in Figure 2.4, can be defined as:

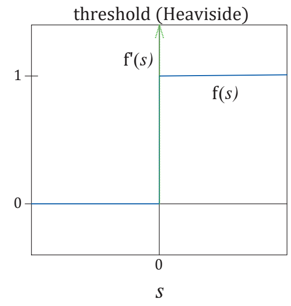
$$\begin{aligned} s[n] &= \sum_{j=1}^{N} w_j x_j[n] + b \\ y[n] &= f(s[n]) \end{aligned} \tag{2.1}$$

where

- $x_1[n], x_2[n], \cdots, x_N[n]$ are the input signals,
- $w_1, w_2, \cdots, w_N$ are the respective synaptic weights of neuron k,
- b (or w_0) is the bias,
- $s[n]$ is the output of the linear combination of input signals,
- $f(\cdot)$ is the activation function,
- $y[n]$ is the output signal of the neuron.

The activation function is a non-linear function applied to introduce non-linear properties in a neural network. Some widely used options are described as follows.

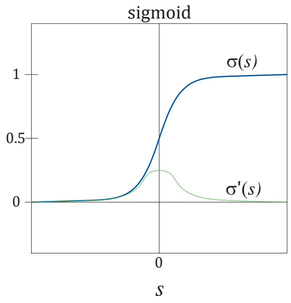
- The threshold function is commonly referred to as the Heaviside step function, and its derivative is the Dirac delta function.



$$
\begin{aligned}
&f\,(s) = 1 \quad \text{if } s \geq 0\\
&f\,(s) = 0 \quad \text{if } s < 0\\
\\
&f'\,(s) = 0 \quad \text{if } s < 0\\
&f'\,(s) = \infty \;\; \text{if } s = 0\\
&f'\,(s) = 0 \quad \text{if } s > 0
\end{aligned}
\tag{2.2}
$$

Figure 2.5: The threshold function.

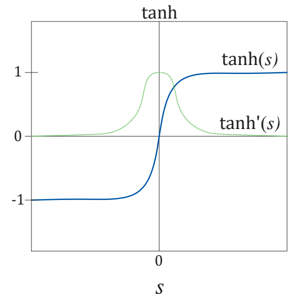
- The sigmoid function, whose graph is “S”-shaped, is the most common activation function used in neural networks. It is defined as a strictly increasing function that exhibits a graceful balance between linear and non-linear behavior. An example of the sigmoid function is the logistic function.



$$
\begin{aligned}
\sigma(s) &= \frac{1}{1+e^{-s}}\\
\sigma'(s) &= \sigma(s)(1-\sigma(s))
\end{aligned}
\tag{2.3}
$$

Figure 2.6: The sigmoid function.

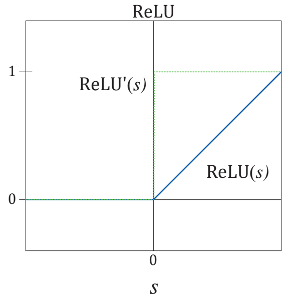
- The hyperbolic tangent (tanh) function is a scaled and shifted version of the sigmoid function. Together with the sigmoid function, it has the pros of simple derivatives and the output bounded between finite values. The cons relate to info loss due to the short derivative range for high and small s, where gradients can be close to zero (vanishing gradients).



$$
\begin{aligned}
\tanh(s) &= \frac{e^{s}-e^{-s}}{e^{s}+e^{-s}}\\
\tanh'(s) &= 1-\tanh(s)^2
\end{aligned}
\tag{2.4}
$$

Figure 2.7: The tanh function.

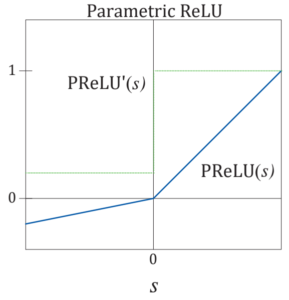
- The Rectified Linear Unit (ReLU) function shows the advantages related to the linear part that speeds up the computation, especially of gradients. However, the ReLU function does not allow for negative values, so certain patterns may not be captured, and output values can be very large (exploding gradients).



$$
\begin{aligned}
\text{ReLU}(s) &= \max(0, s) \\
\text{ReLU'}(s) &= \max(0, 1)
\end{aligned}
\tag{2.5}
$$

Figure 2.8: The ReLU function.

- The Parametric Rectifier Linear Unit (PReLU) is an activation function similar to ReLU, with the advantage of also assuming negative values.



$$
\begin{aligned}
\text{PReLU}(s) &= as \;\text{ for }\; s < 0 \\
\text{PReLU}(s) &= s \quad\text{ for }\; s \geq 0 \\
\\
\text{PReLU'}(s) &= a \quad\text{ for }\; s < 0 \\
\text{PReLU'}(s) &= 1 \quad\text{ for }\; s \geq 0
\end{aligned}
\tag{2.6}
$$

Figure 2.9: The PReLU function.

- The Exponential Linear Unit (ELU) is an activation function similar to PReLU, introduced to solve some limitations of other activation functions such as ReLU and its variants. As for the PReLU, the advantage of the ELU function is that it has a non-zero gradient for positive and negative values of s, which can help speed up convergence during training. In addition, the ELU function has been shown to perform better than ReLU and its variants in some scenarios, especially when dealing with noisy data [75].

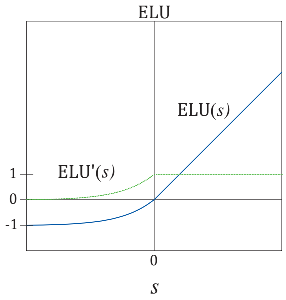


Figure 2.10: The ELU function ($a = 1$).

$$\begin{aligned}
&\text{ELU}(s) = a(\exp^s - 1) && \text{for } s < 0\\
&\text{ELU}(s) = s && \text{for } s \geq 0\\
\\
&\text{ELU'}(s) = \text{ELU}(s) + a && \text{for } s < 0\\
&\text{ELU'}(s) = 1 && \text{for } s \geq 0
\end{aligned} \tag{2.7}$$

- The softmax is an activation function that maps a vector $\mathbf{s}$ of K real values into a vector of K real values that sum to 1 so that these values can be interpreted as probabilities.

$$\text{softmax}(s_j) = \frac{\exp(s_j)}{\sum_{k=1}^{K} \exp(s_k)} \qquad \forall j = 1, \cdots, K \tag{2.8}$$

  where

  - s_j is the j-th element of the input vector,
  - $\exp(s_j)$ is the standard exponential function applied to s_j,
  - $\sum_{k=1}^{K} \exp(s_k)$ is a normalization term to ensure that the output vector values sum to 1,
  - K corresponds to the number of model outputs if softmax is the activation function of the final layer.

The connection between input and output, known as the transfer function, is learned from data instead of being programmed. Training starts by randomly assigning weights w_j, which are refined as learning progresses.

2.3 Deep Neural Network Architectures

The organization of neurons in an artificial neural network is strongly related to its purpose. This section provides a concise overview of the neural network architectures used in this research.

2.3.1 Multi-Layer Perceptron

The multi-layer perceptron (MLP) or multi-layer feed-forward network is an artificial neural network characterized by one or more hidden layers whose computation nodes are correspondingly called hidden neurons. Artificial neural

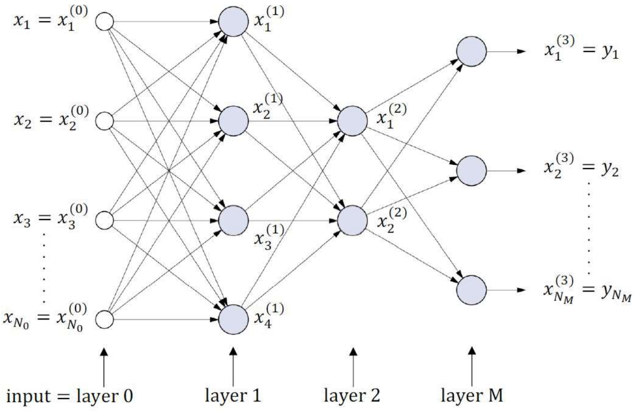


Figure 2.11: Multi-layer perceptron.

networks are often referred to as "deep" when they have more than one or two hidden layers. An MLP with one or more hidden layers and a sufficient number of non-linear units can approximate any continuous function on a compact input domain with arbitrary precision.

Multi-Layer Perceptron Architecture

The MLP architecture consists of multiple layers of neurons, each fully connected to those in the previous layer (from which they receive input) and those in the subsequent layer (which they, in turn, influence).

In a multi-layer neural network composed of M layers of neurons, each neuron of the k-th layer is connected only to all the neurons of the $(k+1)$-th layer. No feedbacks are present, and no connections between neurons of the same layer are allowed. Thus, the network has no memory and acts instantaneously (feed-forward). Omitting the n variable for the sake of conciseness, the notation of a generic MLP architecture, shown in Figure 2.11, is defined in the following:

- M is the number of layers (l index),
- N_l is the number of neurons of the l-th layer,
- $s_k^{(l)}$ is the induced local field of the k-th neuron at the l-th layer,
- $x_k^{(l)}$ is the output of the k-th neuron at the l-th layer,
- $w_{kj}^{(l)}$ is the weight connecting the j-th neuron at $(l-1)$-th layer to the k-th neuron at the l-th layer; $w_{k0}^{(l)}$ is the bias weight of the k-th neuron at the l-th layer.

Multi-Layer Perceptron Operations

The mathematical description of the neural activity of the M-hidden-layered MLP is summarized into two main computational steps.
Given $x_k^{(0)}$ with $k = (1, \dots, N_0)$ the input at layer 0, for all layers $l = 1, \dots, M$, do:

- $s_k^{(l)} = \sum\limits_{j=0}^{N_{l-1}} w_{kj}^{(l)} x_j^{(l-1)}$ with $k = 1, \dots, N_l$ and $w_{k0}^{(l)}$ the bias weight,
- $x_k^{(l)} = f(s_k^{(l)})$ where $f(\cdot)$ is the activation function,

and the output at layer M is $y_k = x_k^{(M)}$ with $k = (1, \dots, N_M)$.

The behavior of an MLP architecture is parameterized by the connection weights, which are adapted during an iterative process called network training, composed of two computational phases, a forward and a backward phase. In the forward processing or propagation, input examples are fed to the input layer, and the resulting output is propagated via the hidden layers toward the output layer. During the backward processing or propagation, the error signal originating at the output neurons is sent back through the layers, and the network parameters (i.e., weights and biases) are tuned.

Examining in detail the individual computational steps of the algorithm:

1. Forward phase: for each pattern n, the inputs $x_k[n]$ and the outputs $y_k[n]$ are evaluated. Specifically, given a set of inputs $x_k[n]$ with $k = 1, \dots, N_0$ and $n = 1, \dots, Q$ (Q is the number of input patterns), the aim is to determine the set of weights $w_{kj}^{(l)}$ $\forall k, j, l$ to yield the corresponding outputs $y_k[n]$ with $k = 1, \dots, N_M$.

2. Error computation: $\epsilon_n = \sum_{k=1}^{N_M} E_k^{(M)}$. The network has to approximate the desired outputs, also called targets, defined as $d_k[n]$ with $k = 1, \dots, N_M$ and $n = 1, \dots, Q$ (Q also denotes the number of output patterns). The result is achieved by means of the weights adaptation, which consists in minimizing a suitable Cost function, used to measure the accuracy of this approximation. The standard Mean Square Error (MSE) is often employed, calculated over all output neurons and all Q input/target pairs, and defined as:

$$\epsilon = \frac{1}{Q} \sum_{n=1}^{Q} \epsilon_n \qquad \epsilon_n = \frac{1}{2} \sum_{k=1}^{N_M} (d_k[n] - y_k[n])^2 \tag{2.9}$$

   in which ϵ_n is the Loss function.

3. Backward phase: gradient-based techniques are widely used to minimize the Cost function through the local gradient computation $\delta_k^{(l)}$ for $k = 1, \dots, N_l$ and $E_k^{(l-1)}$ for $k = 1, \dots, N_{l-1}$.

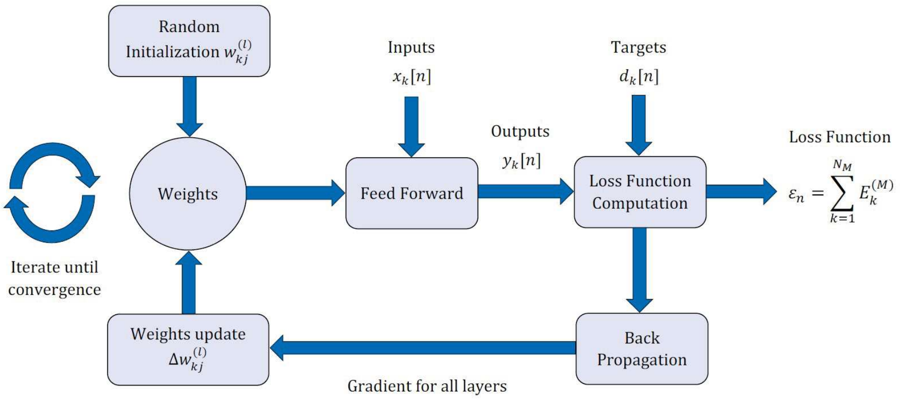

Random Initialization $w_{kj}^{(l)}$
Inputs $x_k[n]$
Targets $d_k[n]$
Outputs $y_k[n]$
Loss Function
$\varepsilon_n = \sum_{k=1}^{N_M} E_k^{(M)}$
Weights
Feed Forward
Loss Function Computation
Iterate until convergence
Weights update $\Delta w_{kj}^{(l)}$
Back Propagation
Gradient for all layers

Figure 2.12: Block diagram of the iterative forward and backward processing.

4. Updating weights $w_{kj}^{(l)}$ for $k = 1, \ldots, N_l$ and $j = 0, \ldots, N_{l-1}$.
5. Iterative process: repeat for all layers $l = (M-1), (M-2), \ldots, 1$.

Several algorithms can be used to minimize the Cost function ϵ, and the Stochastic Gradient Descent (SGD) is the standard choice, where weights are updated pattern-by-pattern by computing partial derivatives of the Loss function. The learning rule for each weight of the network is defined as:

$$\begin{aligned} w'^{(l)}_{kj} &= w_{kj}^{(l)} - \alpha \frac{\partial \epsilon_n}{\partial w_{kj}^{(l)}} \text{ for } k = 1, \ldots, N_l \text{ and } j = 0, \ldots, N_{l-1} \\ &= w_{kj}^{(l)} + \alpha \delta_k^{(l)} x_j^{(l-1)} \\ &= w_{kj}^{(l)} + \triangle w_{kj}^{(l)}, \end{aligned} \tag{2.10}$$

where $w'^{(l)}_{kj}$ is the updated weight, $w_{kj}^{(l)}$ is the previous weight, and α is the learning rate. $\triangle w_{kj}^{(l)}$ denotes the variation of weight $w_{kj}^{(l)}$ at the l-th layer.

Figure 2.12 illustrates the block diagram of the iterative forward and backward processing phases.

2.3.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are specialized feed-forward neural networks that mimic the functioning of the human visual cortex. The main advantage of this network is the robust pattern recognition system characterized by a strong immunity to pattern shifts. In standard applications, the input of a CNN architecture is an image, i.e., a 2D signal with no temporal context. Convolutional neural networks employ the convolution operation in at least one of their layers, and their architectures are generally composed of one or

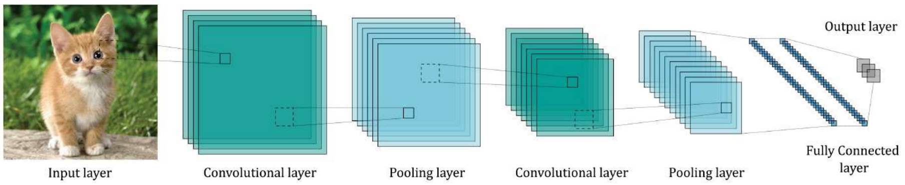


Figure 2.13: Convolutional neural network.

more convolutional, pooling and feed-forward (also called dense or fully connected) layers. Figure 2.13 illustrates a generic convolutional neural network architecture.

The convolutional layer performs a convolution between matrices and adds a scalar bias to produce an output. Then, a non-linearity is applied element-wise. The convolution operation occurs between the input matrix $x_{i,j}$ and the kernel, i.e., the 2D filter $w_{i,j}$ ($m \times m$ matrix $\mathbf{W}$). Kernels are generally smaller than the input, allowing CNNs to process large inputs with few trainable parameters. The 2D convolution operation can be expressed as:

$$y_{i,j} = w_{i,j} * x_{i,j} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} w_{(a,b)} x_{(i-a)(j-b)} \tag{2.11}$$

Convolutional kernels process the input data matrix by dividing it into local receptive fields, a region of the same size as the kernel, and sliding the local receptive field across the entire output. Each hidden neuron is thus connected to a local receptive field, and all the neurons form a feature map matrix. The weights in each feature map are shared: all hidden neurons aim to detect the same pattern, just at different locations in the input image. The shape of the input, the shape of the kernel and the sliding step of the kernel determine the output shape of the convolutional layer. In particular, the two fundamental processing paradigms of the convolution operation are padding, which consists of adding zero pixels around the boundary of the input image to prevent loss of information, and stride, which is the process of moving the kernel window more than one element at a time to reduce the output dimensions.

After the convolutional layer, a pooling layer is usually applied to reduce the feature map dimensions and speed up the computation. The pooling layer reduces the dimension of the matrix by a rule: a sub-matrix of the input is selected, and the output is the maximum (max-pooling) or the average (average-pooling) value of this sub-matrix. The pooling process introduces tolerance against shifts in the input patterns. Together with the convolutional layer, it allows the CNN to detect if a particular event occurs, regardless of its deformation or position. The pooling operation aims to introduce robustness against

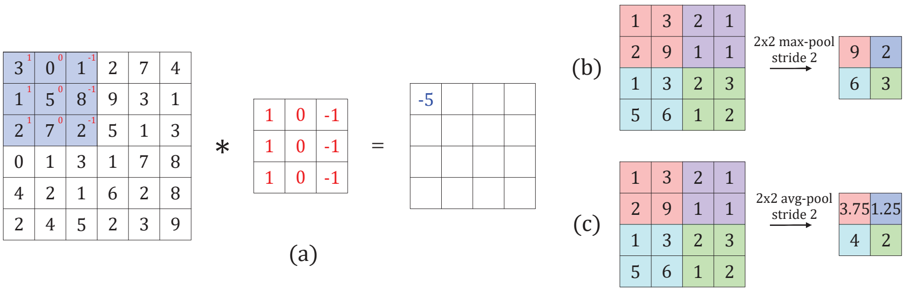


Figure 2.14: Convolution (a), max-pooling (b), and average-pooling (c) operations.

translations of the input patterns.

Finally, at the top of the network, a layer of neurons is applied. This layer does not differ from the multi-layer perceptron, being composed by a set of neurons and being fully connected with the previous layer. In Figure 2.14, examples of convolution, max-pooling and average pooling operations are shown.

The forward phase of a convolutional neural network can be summarized into the operations occurring in each type of layer:

- In convolutional layers, the input matrix dimension can be assumed with a $N \times N$ size, the kernel matrix dimension is $m \times m$, the convolutional layer has the size $(N-m+1) \times (N-m+1)$, and its ij entry is equal to

$$x_{ij}^{(l)} = f\left(b^{(l)} + \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} w_{ab}^{(l)} x_{(i-a)(j-b)}^{(l-1)}\right) = f(s_{ij}^{(l)}) \tag{2.12}$$

- In pooling layers, the operation consists of taking some $p \times p$ region, with $p \in \mathbb{N}$ and yielding a single value, which is the maximum (max-pooling) or the average (average-pooling) of that region.

- In feed-forward layers, standard forward operations are computed.

Similarly, backward propagation in the different types of layers is computed in this way:

- In convolutional layers, the partial of the error function with respect to each output of neuron activities is computed and back-propagated from the output to the convolutional layer. For each entry $w_{ab}^{(l)}$ of the kernel $\mathbf{W}^{(l)}$ at layer l, the chain rule is applied, and the gradient component is obtained as:

$$\frac{\partial \epsilon}{\partial w_{ab}^{(l)}} = \sum_{i=0}^{N-m} \sum_{j=0}^{N-m} \frac{\partial \epsilon}{\partial s_{ij}^{(l)}} x_{(i-a)(j-b)}^{(l-1)} \tag{2.13}$$

To compute the weight updates for the convolutional layer, the derivatives are calculated by applying the chain-rule as follows:

$$\frac{\partial \epsilon}{\partial s_{ij}^{(l)}} = \frac{\partial \epsilon}{\partial x_{ij}^{(l)}} f'(s_{ij}^{(l)}) \tag{2.14}$$

Finally, the errors are back-propagated to the previous layer:

$$\frac{\partial \epsilon}{\partial x_{ij}^{(l-1)}} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \frac{\partial \epsilon}{\partial s_{(i+a)(j+b)}^{(l)}} w_{ab}^{(l)} \tag{2.15}$$

This means applying a convolution to the derivatives by using the kernel matrix at layer l, denoted as $\mathbf{W}^{(l)}$ flipped along both axes (with zero-padding).

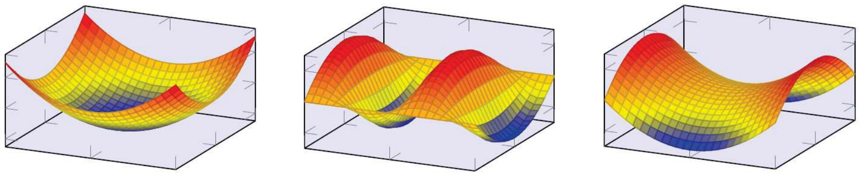
- In pooling layers, each value yielded in the forward phase corresponds to an error coming through the back-propagation. This error is forwarded to the previous layer with upsampling.
- In feed-forward layers, standard back-propagation operations are performed.

2.4 Optimization Algorithms

Most deep learning algorithms involve optimization in the training phase. The most widely used is gradient-based optimization, which belongs to the first-order iterative optimization algorithms. Specifically, optimization is the task of minimizing some function $g(x)$ by altering x: $g(x)$ is called the objective function that, in the deep learning field, is also called the Cost, Loss, or error function. The aim of optimization techniques is reached by doing a small change ε in the input x to obtain the corresponding change in the output $g(x)$:

$$g(x + \varepsilon) \approx g(x) + \varepsilon g'(x) \tag{2.16}$$

This formulation is based on the calculation of the derivative $g'(x)$. The gradient descent technique is based on reducing $g(x)$ by moving x in small steps with the opposite sign of the derivative. The aim is to find the minimum of the Cost function: when $g'(x) = 0$, the derivative provides no information about which direction to move, and this point is defined as stationary point. A



(a) Convex function. (b) Non-convex function. (c) “Saddle” function.

Figure 2.15: Examples of Cost functions.

local minimum is a point where $g(x)$ is lower than at all neighboring, and it is no longer possible to decrease $g(x)$ by making infinitesimal steps. The absolute lowest value of $g(x)$ is a global minimum. The gradient descent algorithm shows problems with non-convex functions, as the learning process often gets stuck in a local optimum rather than finding the right way to the global optimum. Also, most points of zero gradients are not local optima but saddle points located in large space regions where gradients are close to zero, so the learning process is slowed. Figure 2.15 illustrates a convex function and examples of functions in which the problems of local optima and zero gradients emerge. The learning rule can be specifically modified, resulting in diverse optimization algorithms, which are outlined below and are the most commonly used.

2.4.1 Stochastic, Batch, and Mini-Batch Gradient Descent

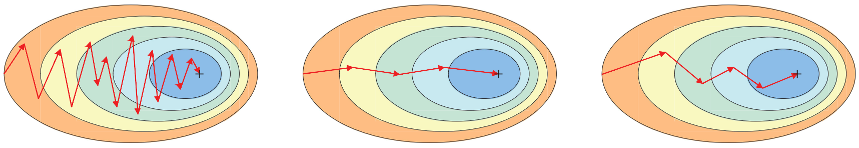
Defined the training set as the set of available input/target Q model pairs, an epoch consists of the complete presentation of the training set during the learning process. Parameters, i.e., weights and biases $w_{kj}^{(l)}$ with $k = 1, \dots, N_l$, $j = 0, \dots, N_{l-1}$, and $l = 1, \dots, M$, are updated at the end of each epoch. On the other hand, hyperparameters, representing the parameters that describe the network architecture and the training characteristics, are chosen and set by the user before the training phase.

One of the hyperparameters that have a key role in optimizing the algorithm is the number of instances processed at a time during the gradient descent that results in the model update and determines different convergence trajectories to the global minimum, as illustrated in Figure 2.16. Different gradient descent algorithms are defined based on the number of training instances processed.

- Stochastic Gradient Descent (SGD): it represents the case previously exposed, also called online or sequential learning. The gradient is computed considering a single input pattern $(x_k[n], d_k[n])$ with $n \in \{1, \dots, Q\}$ chosen randomly from the training set at each iteration. It yields exactly the learning rule seen before.

- Batch Gradient Descent (BGD): the gradient descent algorithm processes the entire training set simultaneously, calculating all gradients for each pattern index n and then averaging them across all available Q patterns.

- Mini-Batch Gradient Descent (mBGD): on each step of the algorithm, a mini-batch of examples $(x_k[p], d_k[p])$ with $p \in \{1, \ldots, P\}$ and $P < Q$ is sampled uniformly from the training set. The mini-batch size is typically chosen to be composed of a relatively small number of examples.



(a) Stochastic Gradient Descent.

(b) Batch Gradient Descent.

(c) Mini-Batch Gradient Descent.

Figure 2.16: Comparison between the convergence trajectories of SGD, BGD, and mBGD.

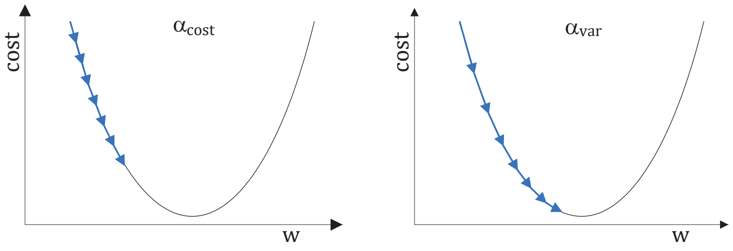
2.4.2 Learning Rate Decay

Learning Rate Decay is a family of strategies that slowly reduces the learning rate over iterations, to be used in optimization algorithms in order to avoid exceeding a good minimum. The learning rate determines the size of the steps taken by the optimizer in adjusting model parameters; a lower learning rate results in smaller steps, which leads to slower convergence and a reduced risk of exceeding a minimum point.

A constant learning rate (α) might negatively influence the algorithm performance: if it is set too high, the algorithm can oscillate and become unstable; if it is too small, the algorithm takes too long to converge. With a variable learning rate, the complexity of the local error surface is responded to, and the entire optimization process benefits. Figure 2.17 is explanatory of constant and variable learning rates in the Cost function minimization process.

One of the most common strategies is to apply to the initial learning rate α_0 a reduction factor proportional to the number of epochs achieved, according to the equation:

$$\alpha = \frac{k}{\text{epoch\_number}} \alpha_0 \tag{2.17}$$



(a) Constant α.

(b) Variable α.

Figure 2.17: Comparison between constant and variable learning rates.

2.4.3 ADAM Algorithm

ADAM (Adaptive Moment Estimation) [76] is a stochastic gradient descent optimization algorithm that uses moving averages of the parameters to provide a running estimate of the second raw moments of the gradients, the mean and variance. In this way, the algorithm adapts the learning rate for each parameter based on the historical gradient information, resulting in faster convergence compared to standard stochastic gradient descent.

ADAM algorithm keeps stored an exponentially decaying average of past squared gradients $v[n]$ and of past gradients $m[n]$:

$$m[n] = \beta_1 m[n-1] + (1-\beta_1)\frac{\partial \epsilon_n}{\partial w_{kj}^{(l)}} \tag{2.18}$$

$$v[n] = \beta_2 v[n-1] + (1-\beta_2)\left(\frac{\partial \epsilon_n}{\partial w_{kj}^{(l)}}\right)^2 \tag{2.19}$$

where $m[n]$ and $v[n]$ are estimates of the first moment (the mean) and the second moment (the variance) of the gradients respectively, hence the name of the method. As $m[n]$ and $v[n]$ are initialized as vectors of 0 values, it can be observed that they are biased towards zero, especially during the initial time steps, when the decay rates are small (i.e., β_1 and β_2 are close to 1).

Bias-corrected first and second-moment estimates can be computed to solve the issue:

$$\hat{m}[n] = \frac{m[n]}{1-\beta_1^n} \tag{2.20}$$

$$\hat{v}[n] = \frac{v[n]}{1-\beta_2^n} \tag{2.21}$$

Then the ADAM update rule is:

$$w_{kj}^{(l)}[n+1] = w_{kj}^{(l)}[n] - \frac{\alpha}{\sqrt{\hat{v}[n]} + \varepsilon}\hat{m}[n] \tag{2.22}$$

Default values for parameters are: 0.9 for β_1, 0.999 for β_2, and 10^{-8} for ε. ADAM works well in practice and compares favorably to other adaptive learning-method algorithms.

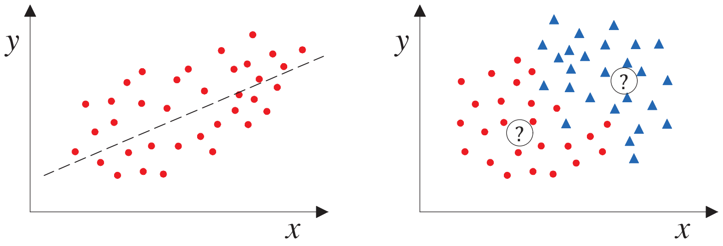
2.5 Supervised Learning

Any learning process aims to improve the subject's experience for a certain task. A neural network is a parametric system that increases task-oriented experience by adapting its parameters (weights) using the available environmental knowledge (data). The learning process can be performed according to two main supervision paradigms: supervised and unsupervised, each tackling a different type of learning problem with a specific type of network architecture.

- In supervised learning, the algorithm makes use of a dataset containing examples associated with a label or target: it is comparable to learning with a teacher.
- In unsupervised learning, no labels are attached to the data. There is no external teacher to oversee the learning process, so the challenge is to self-discover useful patterns in available data.

Supervised learning is the paradigm assumed in this research. It is a type of learning where a network is trained using a set of data that includes inputs and their corresponding outputs. The goal of supervised learning is to have the network recognize the relationship between the input variables and the output and then make predictions for unknown outputs. During the training phase, an algorithm such as back-propagation is used to modify the weights and other parameters of the network, with the goal of minimizing the overall error in the training data. This objective is achieved by providing the network with a significant number of examples to learn from, and the network must also be able to generalize to new cases it has not seen before.

Given as input a finite sequence $S = (x[n], d[n])$ with $n = 1, \dots, Q$ of pairs from $\mathcal{X} \times \mathcal{D} = \mathcal{T}$ (training data), where $d[n]$ is the label or target corresponding to $x[n]$, the output of the learning algorithm is a mapping function $F : \mathcal{X} \to \mathcal{D}$ that has the goal to predict $d[n] \in \mathcal{D}$ given $x[n] \in \mathcal{X}$. The purpose of the learning algorithm is to find a good mapping F between $x[n]$ and $d[n]$ for pairs from $\mathcal{X}_u \times \mathcal{D}_u = \mathcal{T}_u$ unseen during the training (testing data). The Loss function $\mathcal{L} : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ measures the accuracy of the approximation of $d[n]$ by means of $y[n] = F(x[n])$.



(a) Regression.

(b) Classification.

Figure 2.18: Regression and classification learning problems.

Depending on whether $\mathcal{D}$ is continuous or discrete, two types of learning problems are distinguished: regression or classification, which require different Loss and, consequently, Cost functions.

- Regression: the algorithm is asked to predict a continuous value given a certain input. For instance, it outputs a function $F : \mathbb{R}^{N_0} \rightarrow \mathbb{R}$ where N_0 is the input dimensionality. So, regression models are used to predict a continuous value.
- Classification: the algorithm is asked to specify to which of C categories a given input belongs using a function $F : \mathbb{R}^{N_0} \rightarrow \{1, \ldots, C\}$. Therefore, the model has to predict which category (class), among some discrete set of options, an example belongs. The type of classification task depends on the number of classes and their concurrency (binary, multi-class, and multi-label).

In Figure 2.18, regression and classification problems are schematized.

2.6 Generalization

The generalization capability of a neural network is its ability to perform well when it is fed with data unseen during training. Several factors affect the input-output mapping learned by the neural network, including the size of the training set, the size of the network, and the complexity of the problem.

The training set is used to train a deep learning model, and the error measure computed on the training set is the training error. In a regression task, the model is trained by minimizing the training error, which is computed as follows:

$$\frac{1}{Q^{(train)}} \sum_{n=1}^{Q^{(train)}} \sum_{k=1}^{N_M} \left(d_k[n]^{(train)} - y_k[n]^{(train)}\right)^2 \tag{2.23}$$

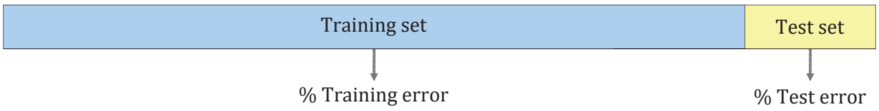


Figure 2.19: Partitioning of the dataset in training and test sets.

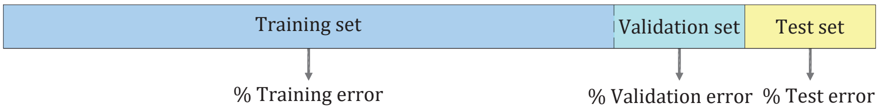


Figure 2.20: Partitioning of the dataset in training, validation, and test sets.

At the same time, a deep learning model aims to minimize the generalization error, i.e., the expected value of the error on data belonging to the test set, also called the testing error (Figure 2.19):

$$\frac{1}{Q^{(test)}} \sum_{n=1}^{Q^{(test)}} \sum_{k=1}^{N_M} \left(d_k[n]^{(test)} - y_k[n]^{(test)}\right)^2 \tag{2.24}$$

The objective of a well-designed deep learning model is to minimize both the training error and the difference between the training and testing errors. In computing a neural model, two situations to avoid are underfitting, when the model cannot achieve a sufficiently low training error, and overfitting, when the gap between the training and the testing error is too large.

During the training phase, it is common practice to use a subset of the training set called the validation set (or development set) to monitor the performance of the model during the training phase, generally at the end of a bunch of q epochs. The validation set is the sample of data used to provide an unbiased evaluation of a model fit on the training set while tuning the model hyperparameters (Figure 2.20).

Several strategies for optimizing the generalization capability of a neural network have been proposed in the literature. One of the most commonly applied techniques is to use the training duration to find the optimal number of epochs for the best results.

For this purpose, the early-stopping strategy has been designed, as depicted in Figure 2.21. For each iteration step i, it consists of performing the network training for a fixed number of epochs q, evaluating the network performance on the validation set, and comparing the validation performance at epoch $i \times q$ with the performance at epoch $(i-1) \times q$. If the error increases, the training is stopped (a patience factor can be used on the scope).

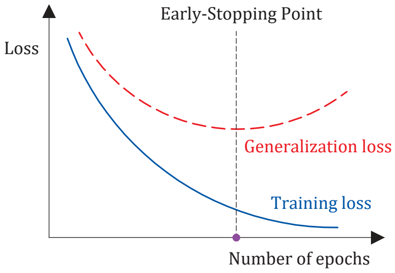


Figure 2.21: The early-stopping generalization method.

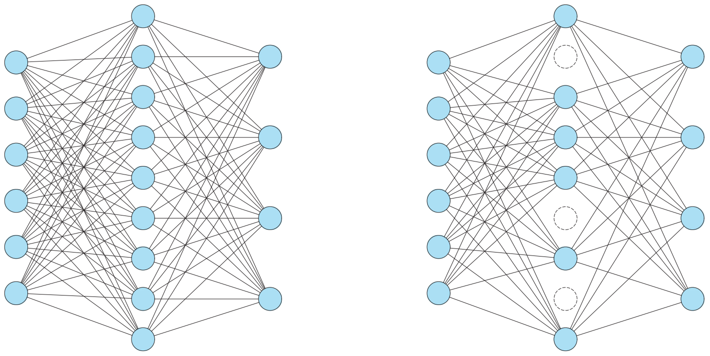
2.7 Regularization Techniques

In order to obtain more robust models, different techniques have been proposed to regularize the weight update during neural network training. They aim to improve the generalization properties of the model, i.e., the ability to perform on newly unseen data as well as (in a reasonable manner) on the training set. A brief description of the most common techniques is given in the following paragraphs.

2.7.1 Dropout

Dropout [70, 77] provides a computationally inexpensive but powerful method of regularizing a broad family of models. It allows for the reduction of overfitting by preventing complex co-adaptations of neural layers and efficiently evaluating various network layouts. The term dropout refers to randomly dropping out units (both hidden and visible) in a neural network, as shown in Figure 2.22. During the training, units are randomly frozen: a different network layout is evaluated at each training batch (or mini-batch). Each is a thinned version of the network, composed of all the units that survived dropout. Theoretically, if n is the number of units, dropout allows training a collection of 2^n thinned networks with extensive weight sharing. Each thinned network gets trained rarely, thus preventing overfitting.

In the training phase, different sets of neurons are activated for each batch of examples, and a thinned network is sampled. During each batch of examples, forward and backward processing is performed only on the thinned network. Gradients for each parameter are then averaged over the batch. During the testing phase, a single neural network without dropout is used, and the network weights are scaled-down versions of the trained weights.



(a) ANN without Dropout.

(b) ANN with Dropout.

Figure 2.22: Dropout regularization method.

2.7.2 Batch Normalization

Batch normalization (BN) [78] is a method to reduce the internal covariate shift, i.e., the change in the distribution of network activations due to the change of network parameters during the training phase. The variables in neural layers may take values with varying magnitudes, possibly hampering the convergence (learning rate compensation effect) even more critically in deep neural networks. Batch normalization is based on batch statistics and is applied to the individual layers (optionally, to all of them). The benefits of this method are the possibility to use higher learning rates without the risk of divergence, with the consequent acceleration of training speed, and a reduced sensitivity to weights initialization.

The batch normalization algorithm applies to all vector values of a batch (or mini-batch) of the training procedure. It takes in input the vector values $\mathbf{x} \in \mathscr{B}$ of a certain neural layer, where $\mathscr{B}$ is the batch, and transforms it as:

$$BN(\mathbf{x}) = \gamma \frac{\mathbf{x} - \hat{\mu}_{\mathscr{B}}}{\sqrt{\hat{\sigma}^2_{\mathscr{B}} - \varepsilon}} + \beta \tag{2.25}$$

where:

- $\hat{\mu}_{\mathscr{B}}$ is the mean of $\mathbf{x}$ values over the batch $\mathscr{B}$,
- $\hat{\sigma}_{\mathscr{B}}$ is the standard deviation of $\mathbf{x}$ values over the batch $\mathscr{B}$,
- γ, β are learnable parameters,
- ε is a small positive constant that prevents the division by 0.

Batch normalization behaves differently in the training and testing phases: in training, the mean and variance are calculated over a single batch; in testing, the model makes predictions on a sample at a time. Thus, the entire dataset is used to compute stable estimates of the variable statistics and fix them at prediction time.

2.8 Evaluation Metrics

The process of assessing the performance of a system involves estimating its behavior when exposed to new data. The evaluation is considered unbiased when the system is tested on data unseen during the training phase and with available reference annotations. The generated output is then compared with the reference, and metrics are computed to quantify the performance of the algorithm. The definition of performance and the measurement method may vary depending on the objectives and specifications of the system. For example, the accuracy rate may be used to determine the capability of the algorithm to correctly classify or identify a sound. In contrast, the mean absolute error may be used to assess the error made by regression models. It is important to note that no single metric is universally applicable to all algorithms, as each provides a distinct perspective on the performance of the system. The evaluation metrics used in the experiments illustrated in this dissertation are explained in the following. They are distinguished into classification-related metrics employed in the studies concerning emergency siren detection and regression-related metrics for the work mentioned in the other contributions.

2.8.1 Classification-Related Metrics

Concerning a classification task, performance evaluation is done by assessing the predictions made by the system under review against the corresponding annotations or ground truth. The computation of classification metrics is based on the count of correct predictions and several types of errors the system makes. These counts are referred to as intermediate statistics and are established based on the evaluation protocol. The following definitions apply to the intermediate statistics for a target sound event:

- True Positive (TP): a correct prediction that indicates the presence of the event, as denoted by both the system output and the reference.
- True Negative (TN): a correct prediction that indicates the absence of the event, as denoted by both the system output and the reference.
- False Positive (FP): an incorrect prediction, as the system output indicates the presence of the event, while the reference denotes its absence.
- False Negative (FN): an incorrect prediction, as the system output indicates the absence of the event, while the reference denotes its presence.

In this research, the emergency siren detection task has been framed as a single-label binary problem. The intermediate metrics thus reflect the correct recognition of the single true class without generating false alarms. The evaluations were carried out using segment-based metrics, demonstrating the capability of the system to detect fixed-time instances correctly. The performance was measured by comparing the ground truth and system output at the instance level. However, it is paramount to consider the specific requirements of a problem and the characteristics of the dataset when choosing the right metric for performance evaluation. For these reasons, in the several experiments conducted for emergency siren detection, the metrics have been selected according to the dataset composition. In the following, definitions of accuracy, precision, recall, F-score, and area under the precision-recall curve (AUPRC) are reported.

Accuracy

Accuracy is a commonly used metric that measures how often the classifier makes the correct decision. It is the ratio of the correct outputs from the system to the total number of outputs:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.26}$$

Accuracy is a representative metric when the number of positive and negative examples in the dataset is approximately balanced and when the prediction concerns a single class. In cases where the dataset is imbalanced, with a large number of instances of one class and a small number of instances of another, accuracy may not be a reliable measure of performance. In these cases, precision, recall, F-score and AUPRC are more suitable performance metrics.

Precision and Recall

Precision measures the accuracy of the model in classifying a sample as positive. It is the ratio of positive samples correctly classified to the total samples classified as positive, including incorrect classifications:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.27}$$

When the model makes many incorrect positive or few positive correct classifications, this increases the denominator and makes the precision small. On the other hand, the precision is high when the model makes many correct positive classifications (maximize TP) and fewer incorrect positive classifications (minimize FP). Precision reflects how reliable the model is in classifying samples as positive.

Recall measures the ability of the model to detect positive samples. It is the ratio of positive samples correctly classified as positive to the total number of positive samples:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.28}$$

The higher the recall, the more positive samples are detected, which means the model can correctly classify all the positive samples as positive. A model with high recall but low precision classifies most positive samples correctly but has many false positives (i.e., classifies many negative samples as positive). On the other hand, when a model has high precision but low recall, it is accurate when it classifies a sample as positive but can only classify a few positive samples.

F-Score

F-score is a good representation of the overall performance of a classifier when the data is imbalanced and for multi-class classification. It is the harmonic mean of precision and recall:

$$\text{F} - \text{score} = \frac{2\text{TP}}{2\text{TP} + \text{FN} + \text{FP}} = 2 \cdot \frac{(\text{Precision} \cdot \text{Recall})}{(\text{Precision} + \text{Recall})} \tag{2.29}$$

When working with imbalanced datasets, accuracy alone can be misleading because a model can achieve high accuracy by predicting the majority class most of the time. F-score, with its emphasis on both precision and recall, provides a more comprehensive evaluation of the capability of the model to handle imbalanced datasets.

Area Under Precision-Recall Curve

The precision-recall curve consists of multiple pairs of precision and recall values evaluated at different thresholds, such that the tradeoff between the two values can be seen. This representation is typically used for binary classification in situations where classes are heavily imbalanced. AUPRC and average precision (AP) are similar ways of summarizing the precision-recall curve into a single metric. Specifically, AUPRC is defined as the trapezoidal area under the curve, while AP is the weighted mean of the precision achieved at each threshold value n:

$$\text{AP} = \sum_n (\text{Recall}_n - \text{Recall}_{n-1}) \cdot \text{Precision}_n \tag{2.30}$$

Figure 2.23 illustrates an example of the precision-recall curve that plots the precision (the fraction of true positive predictions among all positive predic-

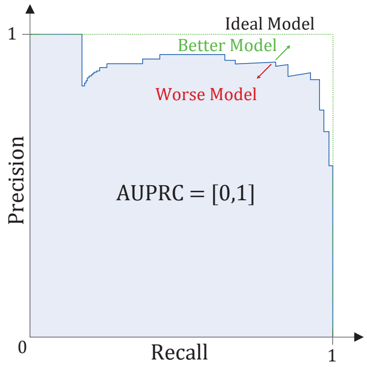


Figure 2.23: Example of Precision-Recall curve.

tions) against recall (the fraction of true positive predictions among all actual positive instances). AUPRC and AP scores range from 0 to 1.

2.8.2 Regression-Related Metrics

Regression metrics are quantitative measures used to evaluate the quality and effectiveness of regression models. These metrics are able to work on a set of continuous values and provide a standardized way to assess how well the predictions of the model align with the actual values. Several common regression metrics serve various purposes in evaluating the performance of a regression model. Each metric focuses on different aspects of the predictive capabilities of the model, such as the magnitude of errors, the variability of predictions, or the proportion of variance explained. Three of the most commonly used regression metrics, mean squared error (MSE), mean absolute error (MAE) and mean absolute percentage error (MAPE), are presented as follows.

Mean Squared Error

MSE measures the average of the squared differences between the target values y_i and the values $\hat{y}_i$ predicted by the regression model:

$$\text{MSE} = \frac{1}{n}\sum_{1}^{n}(y_i - \hat{y}_i)^2 \tag{2.31}$$

Due to the squaring of differences, MSE assigns greater importance to larger errors, resulting in a penalty for even small errors. This characteristic can lead to an overestimation of the inadequacy of the model. However, MSE is commonly favored over other metrics due to its differentiability, enabling more effective optimization during the model-building process.

Mean Absolute Error

MAE measures the absolute differences between the target values y_i and the values $\hat{y_i}$ predicted by the regression model:

$$\text{MAE} = \frac{1}{n}\sum_{1}^{n} |y_i - \hat{y_i}| \tag{2.32}$$

MAE exhibits greater robustness to outliers compared to MSE and imposes less severe penalties on errors. It assigns equal weight to each individual difference, resulting in a linear score. However, it is not well-suited for applications that require emphasis on outlier observations.

Mean Absolute Percentage Error

MAPE, also known as mean absolute percentage deviation (MAPD), is a regression metric that measures the average of the absolute percentage errors between the target values y_i and the values $\hat{y_i}$ predicted by the regression model:

$$\text{MAPE} = \frac{1}{n}\sum_{1}^{n} |\frac{y_i - \hat{y_i}}{y_i}| \tag{2.33}$$

MAPE is commonly used to evaluate the accuracy of predictions in terms of percentage error. The idea of this metric is to be sensitive to relative errors. It is scale-independent, meaning it can be used to compare the performance of models across different datasets or target variables, regardless of their absolute values. This aspect allows for assessing the performance of the model in a more interpretable and intuitive manner.

Chapter 3

Datasets

Any problem solved by machine learning, particularly by deep learning, requires an adequate amount of data for the parameterization of the algorithms. The ability to access public databases makes it possible to test approaches to assess their actual benefit in real-world applications and to compare the performance of existing algorithms on a common basis of comparison. Although several public datasets containing emergency sirens are available [79–81], each country has its own regulations on the characteristics of warning sounds associated with the different categories of emergency vehicles.

The present study focuses on the detection of ambulance sirens according to Italian law. This choice required the *ad hoc* creation of audio data collections to address the challenge of detecting a specific category of siren sounds. The following chapter describes both the methodologies for creating synthetic audio data of Italian ambulance sirens and the equipment and procedures used to record ambulance sirens in the real world. Finally, the acoustic features involved in this research are presented.

3.1 Siren Audio Data

Machine and deep learning techniques exhibit accurate modeling and generalization capabilities because of the data that support them. Specifically, labeled data have a crucial influence on the development and evaluation of supervised algorithms in research fields dealing with sound event detection and classification.

Well-established and easily accessible reference databases attract the research community's interest as readily available study supports, thus accelerating the progress of related fields. There are many popular databases in research areas related to audio, such as the ESC50 [82] and the UrbanSound8K [83], for the classification of environmental and urban sounds, respectively. Considering the quantity, quality and labeling accuracy of the data contained in the most well-known public databases such as those mentioned, the process of creating a dataset for specific system development is naturally very delicate. The audio data content must have a good diversity of characteristics to broaden the learning scope, a significant duration for robust modeling, and careful labeling to represent the aspects of interest. Unfortunately, there is no rule dictating a minimum amount of data, as it usually depends on the intended use and type of algorithm. In addition, there are no specific rules regarding the co-occurrence of environmental sounds.

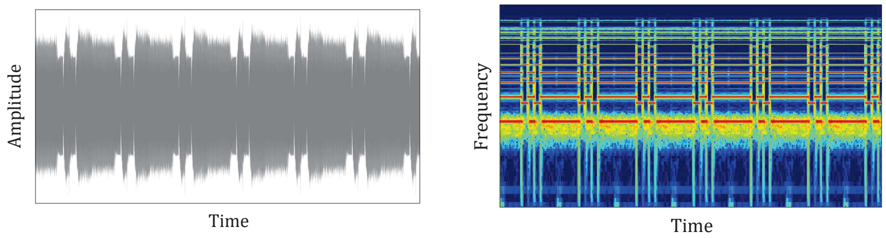
Therefore, creating a database of emergency siren sounds is a complicated task, whether it involves collections of audio files generated via algorithms or recordings in real-world environments. The challenge in synthetic data creation is related to contextualizing the target signal in scenarios of ever-changing and unpredictable vehicular traffic and weather conditions, as well as the faithful reproduction of attenuation phenomena and the Doppler effect associated with the relative velocity between source and observer. On the other hand, the issues involved in the recording procedure lie in the effort required in terms of acquisition times, labeling and processing of audio data. These aspects of the research conducted are discussed in the following sections.

3.1.1 Siren Audio Data Generated via Algorithm

The implementation of an algorithm for the automated generation of siren audio files in background noise contexts has the advantage of producing an audio data collection with controlled quality and content.

In this work, the development of the algorithm began with the selection of a certain type of siren, specifically the acoustic alarm of an Italian ambulance. Its characteristics of duration, periodicity, and tone are regulated by the Ministerial Decree issued by the Ministry of Transport on October 17th, 1980 [84]. A clean audio file of the Italian ambulance siren, consisting of alternating two distinct tones at 392 Hz and 660 Hz, was downloaded from web resources. The siren alarm is composed of two consecutive repetitions of this sequence: the 392 Hz tone for a duration of 1/3 period, followed by the 660 Hz tone for 1/18 period, then the 392 Hz tone for 1/18 period, and finally, the 660 Hz tone for 1/18 period. The total length is (3 ± 0.5) seconds, and the pause between two sound sequences is expected not to exceed 0.2 seconds. The ambulance alarm generates a two-tone siren from a square wave, resulting in a signal comprised of the fundamental frequency and other higher-order harmonics. A single period of the siren sound was isolated and repeated to obtain an audio file of 10 seconds. After that, the Doppler effect and attenuation by distance were applied to the previously generated 10-second ambulance siren.

For the implementation of the Doppler effect, inspiration was taken from the work done in [85], which is based on the interpolation and de-interpolation of



(a) (b)

Figure 3.1: Italian ambulance siren waveform (a) and spectrogram (b).

delay lines. A digital delay line is an elementary processing unit that introduces a time delay between its input and output. Assuming that $x[n]$ is a discrete-time signal of the causal type (i.e., it takes non-zero values only for $n > 0$, while it is equal to 0 for $n < 0$) and M is the length of the delay in samples, the output $y[n]$ is equivalent to the input sequence $x[n]$ delayed by a quantity equal to M:

$$y[n] = x[n - M] \tag{3.1}$$

If the parameter M changes over time, the resulting unit is called a variable-length delay line and can be used to model and simulate moving acoustic sources. A software implementation of the delay line, proposed in [86], is based on the use of a circular buffer of length N. The input signal is written in the buffer sample by sample, at the position of a write-pointer. This position is increased by one at every time step. The output signal is read sample by sample at the position of a read-pointer, delayed by M samples with respect to the write-pointer. The delay M (i.e., the distance between the read and the write-pointer) is allowed to vary over time as long as the condition $M \leq N$ is met.

It is widely acknowledged that a time-varying delay line can lead to a significant frequency shift [87]. For this reason, the time-varying delay is commonly used in creating vibrato and chorus effects [88]. Based on this principle, it is reasonable to assume that a time-varying delay line could accurately simulate the Doppler shift. When considering the Doppler shift from a physical perspective, it is useful to view the air as a metaphor for a magnetic tape, which travels from the source to the listener at the speed of sound. In a scenario where the source and listener remain stationary, the listener hears what has been recorded by the source. On the other hand, the listener perceives the Doppler shift when either the source or listener is in movement. This analogy also works for a computational model based on time-varying delay lines of digital signals. If the delay to be implemented is an integer quantity, the sample of the Doppler-affected signal is given by a sample of the delayed input signal. Otherwise, when the delay to be implemented is a fractional quantity represented by the interpolation time between samples n and $n+1$, expressed by α, the output sample is generated from the linear interpolation between two adjacent samples of the original input signal. In this case, the expression is derived by truncating the first-order Taylor series expansion and fitting an approximation of the first-order derivative[1]:

$$y[n+\alpha] = \alpha x[n+1] + (1-\alpha)x[n] \tag{3.2}$$

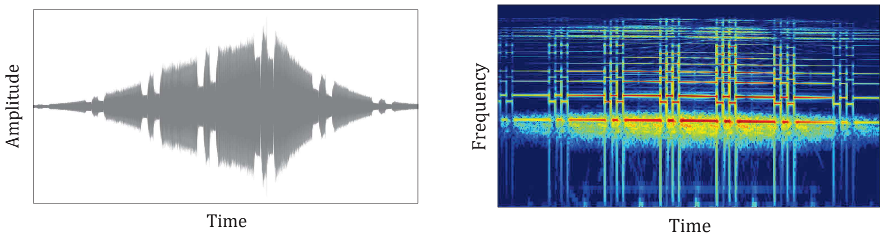
The implementation of the Doppler effect simulation method to be applied to the siren audio file was focused on specific configurations of approaching or moving the source away from the receiver according to the principles of kinematics. Several initial distances and eight directions of motion according to 45-degree angles of a circumference were treated as case studies. Regarding the attenuation of the sound wave due to distance, the far-field propagation was considered, according to which the energy of the spherical wavefront emitted by a source decreases with the square of the distance from the source [89]. Figures 3.1 and 3.2 show the waveform and spectrogram of a clean siren signal before and after applying the Doppler effect.

The last step was to create realistic audio files of sirens immersed in traffic noise contexts. Noise audio files were downloaded from a collaborative database of recordings [90]. Using pre-processing techniques, they were standardized in lengths, sampling rates, channels, and bit depth. All audio data were resampled to 16 kHz with 32-bit depth encoding and made monophonic, then normalized and split into 10-second files. Siren audio files with attenuation for distance and Doppler effect were added to urban traffic noises at decreasing SNRs to simulate rising levels of traffic noise and, thus, more challenging situations for the emergency siren detection task.

3.1.2 Siren Audio Data Recorded

In May 2021, a campaign of siren recordings was planned and conducted using a car equipped with eight condenser microphones model Behringer ECM8000. The installation setup included four microphones inside the passenger compartment, with two at the sides of the front seats and two at the rear seats at seatback height, two in the trunk at the floorboard height, and two behind the license plate on opposite sides. The microphones were connected via XLR connectors to an eight-channel Roland Octa-Capture soundboard, which in turn was interfaced via USB to a laptop controlled by an operator

[1] https://ccrma.stanford.edu/~jos/Interpolation/Interpolation_4up.pdf



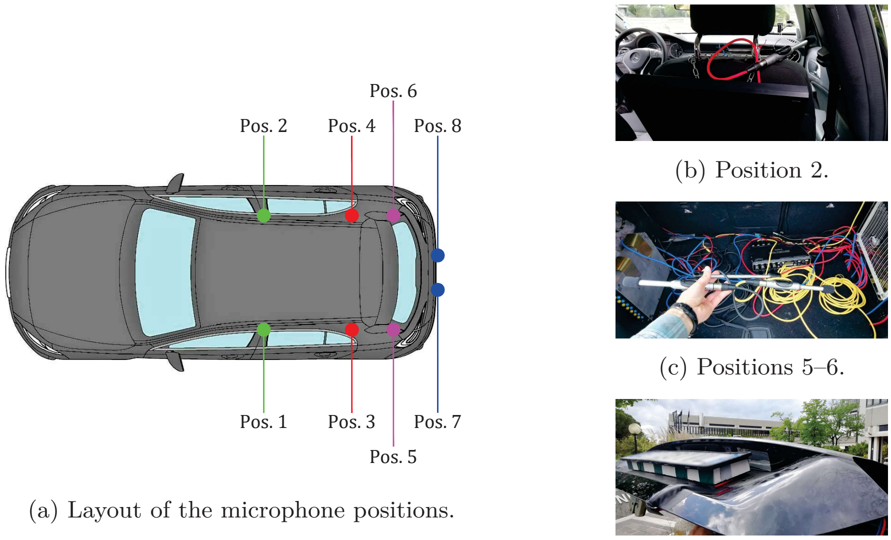
(a) (b)

Figure 3.2: Dopplered Italian ambulance siren waveform (a) and spectrogram (b).

inside the car. The positions of recording sensors were assigned to all relevant places of the vehicle: at the front and rear of the passenger compartment, in the trunk, and externally. The positions inside the passenger compartment were evenly distributed within the cabin and did not interfere with the view, the air conditioning vents, or the audio system. The locations of recording sensors were also carefully planned concerning different utilization. Internal microphones could be used for the audio equalization system [91]; the trunk represents a weather-protected environment scarcely affected by the sounds inside the passenger compartment and offers other applications, such as asphalt wetness detection [9]; the installation behind the license plate is a location in the outdoor setting that combines rapid responsiveness to external signals with a moderately sheltered condition from wind and weather. Figure 3.3 shows the microphone setup of the equipped car.

Recordings were performed for seven days, with the car moving in traffic and stopping at parking areas, always with the engine running. Itineraries were carefully planned to cover the busiest roads where the transit of ambulances requires the activation of the siren alarm, focusing on the most populated city of the Marche Region in Italy. High-traffic suburban areas near the main hospital and central urban areas were explored. Sirens were recorded in several contexts: adjacent to a construction site, along a coastal road, in a suburban area near a shopping center and a residential neighborhood, in the city center, and on a high-speed road. Different driving settings were considered: stationary with the engine running, at moderate speed with frequent stops in urban areas, and at high speed in suburban locations.

Recordings were carried out separately in eight channels, corresponding to the eight microphones, with 44.1 kHz sampling rate and 32 bit-depth, and saved in wav format for a total of 18 audio tracks of approximately 10 hours and 30 minutes. The content of each track was analyzed, and the portion of the audio file in which the siren sound was audible, even weakly, with reference



(a) Layout of the microphone positions.

(b) Position 2.

(c) Positions 5–6.

(d) Position 8.

Figure 3.3: Setup of the car equipped with audio recording devices.

to the channels corresponding to the external microphones, was selected and labeled as siren. Spectrogram visualization helped identify the presence of the fundamental frequencies and the upper harmonics of the two siren tones for correctly labeling the audio files.

3.1.3 Acoustic Features

The time-domain representation of a sound signal, or waveform, is not straightforward to interpret directly. For this reason, frequency and time-frequency domain representations that provide features of sound signals more closely with human perception have been used for years. In this section, a brief description of the representations employed in the present research is provided.

STFT Spectrograms

The short-time Fourier transform (STFT) is a popular method for analyzing speech and audio signals because it is simple to use and computationally efficient. The STFT breaks down an audio signal into smaller overlapping segments, called windows, and calculates the frequency spectrum for each window using the Fast Fourier Transform. This results in a representation of the signal that shows how its frequency content changes over time. The formula for the STFT is given by:

$$STFT[f,t] = \sum_{n=0}^{L-1} s[n] \cdot w[t] e^{-j2\pi fn}, \tag{3.3}$$

where $STFT[f,t]$ is a function that indicates how the spectral content of the signal evolves over time, with time represented by the row index t and frequency represented by the column index f. $s[n]$ is the audio signal, L is the window length, and $w[t]$ is the Hanning window function. The choice of window function and window length determines the trade-off between time and frequency resolution, with a larger window length resulting in better frequency resolution at the expense of poorer time resolution. The STFT allows for adjustable time-frequency sampling to control the resolution of the final representation, called the STFT spectrogram.

Log-Mel Spectrograms

Log-Mel spectrograms are widely used acoustic features for sound event detection and classification tasks. The feature computation begins with a set of short-time Fourier transform spectra. They are calculated from an input signal divided into frames lasting between 20 and 40 milliseconds since the signal is not subject to significant changes on a short-term scale. The Mel filter bank, composed of a set of triangular filters in the Mel scale, simulates the overall frequency selectivity of the human auditory system using the frequency warping:

$$Mel(f) = 2595 \log_{10}(1 + \frac{f}{700}) \tag{3.4}$$

The filter bank is then applied to the power spectra to generate a Mel spectrogram. Finally, logarithmic scaling is applied to obtain the log-Mel spectrogram.

Gammatone Spectrograms

Gammatone spectrograms or gammatonegrams [92] are the results of the application of a set of bandpass filters in the equivalent rectangular bandwidth (ERB) scale after STFT computation on input signals. This type of filter bank, characterized by a band whose amplitude increases with the central frequency f_c, was introduced to describe the impulsive response of the human auditory system and represents the auditory perception, emphasizing audible frequencies. The impulse response centered in a given frequency f_c takes the expression:

$$g(t, f_c) = \begin{cases} at^{n-1} e^{-2\pi bt} \cos(2\pi f_c t + \phi) & \text{if } t \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

where a controls the gain, n is the filter order, b is filter bandwidth, ϕ is the phase of the carrier, and f_c is central frequency.

3.1.4 Harmonic Filtering

For many applications, it is necessary to consider only the harmonic or percussive part of an audio signal, but sometimes sounds are neither fully harmonic nor percussive, such as clapping, rain, and vehicle engine noise. The harmonic content of an audio signal often plays a relevant role in its identification. To improve the performance of a classification algorithm, a source separation technique that reduces the percussive and residual components and enhances the harmonic ones, as described in [93], can be applied to tonal sounds occurring in the presence of background noise, such as emergency sirens. The principle of this technique is based on the spectrogram decomposition method inspired by the sines+transients+noise (STN) audio model [94]:

$$S = H + P + R \tag{3.6}$$

where H contains the harmonic, P the percussive, and R the residual components not included in H or P.

This decomposition technique was first developed in [95]. The audio signal is transformed from the amplitude to the frequency domain through the STFT computation. Then, a median filter is applied horizontally and vertically to obtain one spectrogram with the accentuated harmonic components and the other with the percussive ones. Two separation factors $(\beta_h, \beta_p) \geq 1$ are defined to increase the harmonic-to-percussive ratio and vice versa in the spectrogram. The choice of a harmonic $\beta_h > 1$ allows clearly harmonic components to be retained and percussive and residual components to be eliminated.

The librosa [96] library includes a median-filtering harmonic percussive source separation function, and the *margin* parameter defines both the separation factors. If *margin*=1, a spectrogram is decomposed in $S = H + P$; if *margin*> 1, a spectrogram is decomposed in $S = H + P + R$, with a greater harmonic or percussive separation according to the values assigned to (β_h, β_p).

Chapter 4

Convolutional Neural Networks for Emergency Siren Recognition

Convolutional neural networks (CNNs), which have become popular in computer vision for visual recognition and image classification [97, 98], have also been successfully applied in the audio fields of speech [99] and music analysis [100, 101]. Their scope has been extended to the detection and classification of environmental sounds characterized by varied and chaotic structures, demonstrating their effectiveness in capturing energy modulation patterns across time and frequency [102]. They also have the ability to learn and identify spectro-temporal features representative of different classes of sounds, even if part of the sound is masked by noise [103–105]. However, the limited initial exploration of CNNs in classifying environmental sounds can be attributed to the paucity of labeled data. This problem is crucial for deep neural networks, which need a significant amount of training data to learn from input to output a non-linear function that can generalize with good performance on data unseen during the training phase. For this purpose, research has devised strategies to overcome the problems encountered and obtain a wide availability of labeled data, thereby improving the potential of CNNs in the fields of sound event detection and classification [106, 107].

The generation of synthetic datasets is a strategy for obtaining data of the desired numerosity with controlled characteristics and accurate labeling. This section discusses the experiments for emergency siren recognition with convolutional neural networks by employing synthetic audio data to derive datasets for training neural models. Again with synthetic datasets, these models have been tested to assess their capability to generalize to increasingly variegated and challenging scenarios. Also, different acoustic features and optimization strategies to reduce the network hyperparameters have been investigated.

4.1 CNN-based Approach with Synthetic Dataset

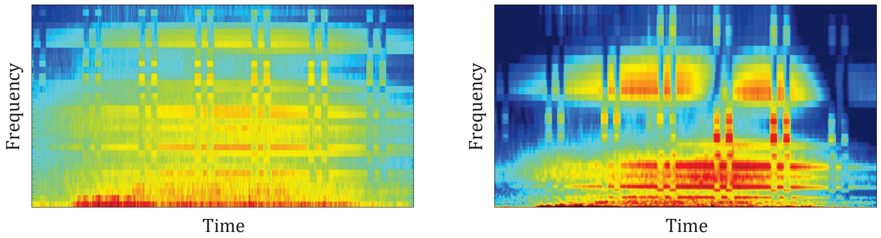
In this first phase of the research, the main objective focused on developing a reliable emergency siren recognition system based on convolutional neural networks. The robustness of the solution has been related to the identification of siren tone components in time-frequency representations computed from fixed-length audio files in contexts with vehicular traffic noise and adverse weather conditions. The work concentrated on generating synthetic audio data from which acoustic features have been computed and defining the neural architecture to realize such a system. The comparison of the performance obtained with three datasets and two different acoustic features provided important insights into the accurate identification of siren sounds.

4.1.1 Proposed System

The proposed system is an approach based on the deep neural architecture presented in [41], where the detection and localization of acoustic alerting events in urban scenarios have been performed in a multitasking learning scheme, along with signal denoising. In this study, the encoding path of such architecture has been taken up and adapted to siren/noise classification. The whole system consists of an acoustic feature computation phase and a classification phase. The acoustic feature computation phase transforms the time-varying audio signal into acoustic spectral features. Then, the classification phase takes the feature vectors as input and maps them into a binary classification of ambulance siren presence or absence. The network parameters have been computed in a supervised manner, using the annotations of fixed-length audio segments as one hot target vector.

Acoustic Feature Computation

The acoustic feature computation procedure operated on 0.5-second mono audio signals sampled at 16 kHz. Two spectrogram-like representations have been used and compared: log-Mel and gammatone spectrograms. The choice of these acoustic features is related to their effectiveness demonstrated in similar work. Log-Mel spectrograms have shown good performance in audio tagging [108] and sound event detection tasks [109]. Similarly, gammatonegrams have been proven to be robust in identifying sound events in contexts where noise is significant [110, 111]. Log-Mel spectrograms have been computed by filtering the magnitude spectrum of the STFT with a filter bank consisting of 40 filters uniformly spaced in the Mel frequency scale, while the gammatone filter bank comprised 64 filters in the ERB frequency scale. For each segment, the STFT has been calculated on frames with a Hanning window of 25 ms, a hop size of



(a) Log-Mel spectrogram.

(b) Gammatone spectrogram.

Figure 4.1: Comparison of acoustic feature representations of a siren sound in background noise.

10 ms, and a fast Fourier transform of 1024 points. The resulting spectrograms have been converted to a logarithmic scale to accommodate human perception of volume. All audio files have a duration of 0.5 seconds, so the resulting feature matrix $x \in \mathbb{R}^{D1 \times D2}$ has a shape of 51 × 40 for log-Mels and 51 × 64 for gammatonegrams. Both representations have been saved as grayscale images at a resolution of 496 × 368 pixels. The librosa toolbox [96] has been used to extract log-Mel features, and gammatone spectrograms have been computed according to the Python adaptation of the algorithm described in [92]. Figure 4.1 compares log-Mel and gammatone spectrograms of a siren audio file in background noise.

CNN Architecture

The architecture of the convolutional neural network employed for emergency siren recognition is shown in Figure 4.2. The first stages of the model are CNN blocks, where convolutional layers act as feature map extractors on the input representations. At the end of each block, max-pooling is used to halve the dimensions of the feature maps output from the convolutional layers. Finally, the feature maps are flattened and passed to densely connected layers that deal with the classification task.

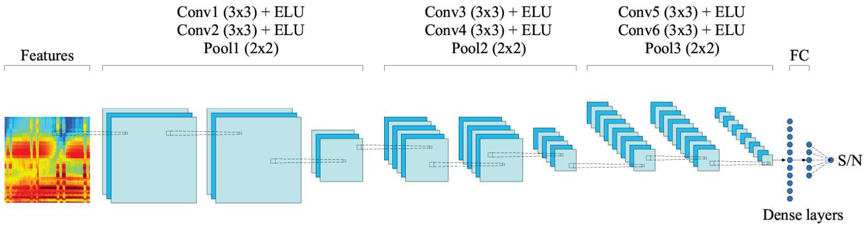


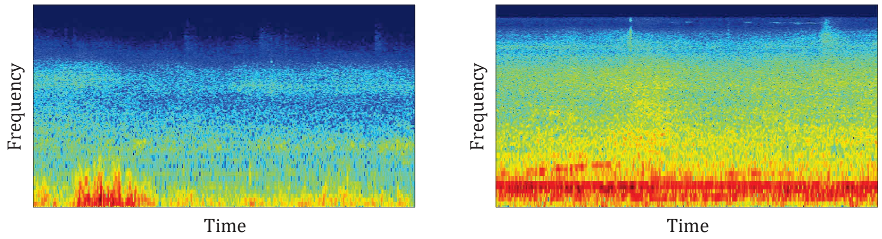
Figure 4.2: CNN architecture for noise/siren classification.

Layer	Kernel size	Stride	Nums of filters
Input	-	-	-
Conv1	(3,3)	(1,1)	4
Conv2	(3,3)	(1,1)	4
Pool1	(2,2)	(2,2)	-
Conv3	(3,3)	(1,1)	8
Conv4	(3,3)	(1,1)	8
Pool2	(2,2)	(2,2)	-
Conv5	(3,3)	(1,1)	16
Conv6	(3,3)	(1,1)	16
Pool3	(2,2)	(2,2)	-
FC1	-	-	10
FC2	-	nums of classes	nums of classes

Table 4.1: Configuration of the CNN.

The CNN consists of six convolutional layers and two fully connected layers. The convolutional part is organized into three blocks with the same structure but a different number of filters. The first convolutional block comprises two convolutional layers with a 3×3 kernel and 4 filters, and an exponential linear unit (ELU) activation function is applied after each of them. The final layer of each block performs a 2×2 max-pooling with a stride equal to 2. The number of feature channels doubles in the subsequent two convolutional blocks, from 4 to 8 to 16. Then, the feature maps are flattened and given as input to a fully connected layer with 10 units. Finally, a softmax activation function is applied in the last layer, returning an output vector representing probabilities that an input feature vector x belongs to the noise or siren class. Details of the CNN architecture are presented in Table 4.1.

To confirm its effectiveness, the reference neural architecture [41] was analyzed through a grid search [112] of the network hyperparameters aimed at its optimization in relation to the characteristics of the input dataset. In particular, the investigation focused on the convolutional layers and activation functions on the intermediate layers. Concerning the convolutional layers, the numerosity of the dataset and the reduced complexity of the input representations did not require the inclusion of additional convolutional blocks. This aspect made it possible to contain the number of hyperparameters of the network. Adding dropout or batch normalization layers also did not make significant improvements. In terms of activation functions, investigations with both the ReLU and ELU activation functions yielded better results with the latter in terms of speed and classification performance. The explanation lies in the characteristics of the ELU activation function. Like the ReLU, the ELU has a linear part for positive values. However, it also admits negative values in a reduced range that allows the mean values to be closer to zero, similar to the batch normalization process. For this reason, the reduced variation in forward-propagated information enables rapid convergence, speeding up the computation. In addition, the saturation plateau in its negative regime allows



(a) Spectrogram of noise type (A).

(b) Spectrogram of noise type (B).

Figure 4.3: Examples of traffic noise spectrograms.

for learning a more robust and stable representation, providing better performance in classification [75].

4.1.2 Dataset

Three sets of audio segments equally balanced between sirens and noises have been collected to compute the acoustic features. Siren audio files with attenuation for distance and Doppler effect have been generated using the procedure described in Section 3.1.1 and added to urban traffic noises at decreasing SNRs. Noise audio files used for both siren and noise classes have been downloaded from web resources [90]. All the audio files have been pre-processed to standardize their characteristics. They have been resampled to 16 kHz, encoded to 32-bit depth, made monophonic through the amplitude averaging of each audio channel, and normalized. Finally, they have been split into 0.5-second chunks with an overlap of 10 ms.

The datasets created for training have been called training sets (A), (B), and (A+B). The training set (A) comprises 64 000 audio segments, of which 32 000 are urban traffic noises, and the remaining 32 000 are siren sounds mixed with traffic noise with spectral content mostly below 2500 Hz. Also, the training set (B) consists of 64 000 audio segments with the siren/noise distribution equal to the training set (A); the only difference is that the traffic noise spectral content added to siren audio files is mostly below 5500 Hz. The training set (A+B) is the sum of the previously described datasets. Siren audio files that compose the training sets have been generated with SNRs of 0 dB, -5 dB, -10 dB, and -15 dB. Figure 4.3 presents examples of spectrograms computed on background noises used to generate the synthetic siren audio files for each dataset.

Also, three datasets have been created for the testing phase, called test sets (a), (b), and (a+b). The test set (a) includes seven collections of 12 000 audio segments, of which 6000 are noises, and 6000 are sirens mixed with noise with

Dataset	Class (label)	Samples	SNR (dB)
Training (A)	Noise (0) Siren (1)	32 000 32 000	- 0,-5,-10,-15
Training (B)	Noise (0) Siren (1)	32 000 32 000	- 0,-5,-10,-15
Training (A+B)	Noise (0) Siren (1)	64 000 64 000	- 0,-5,-10,-15
Test (a)	Noise (0) Siren (1)	6000 6000 × 7 SNRs	- 0,-5,-10,-15,-20,-25,-30
Test (b)	Noise (0) Siren (1)	6000 6000 × 7 SNRs	- 0,-5,-10,-15,-20,-25,-30
Test (a+b)	Noise (0) Siren (1)	12 000 12 000 × 7 SNRs	- 0,-5,-10,-15,-20,-25,-30

Table 4.2: Dataset composition.

spectral content similar to the training set (A). The siren audio files have been generated at several SNR values, equal to 0 dB, -5 dB, -10 dB, and -15 dB as for the training set, with the addition of -20 dB, -25 dB, and -30 dB to evaluate the capability of the neural model to generalize the siren recognition at SNRs unseen during the training. The same subdivision has been assigned to test set (b), whose siren files present a background noise with similar characteristics to the training set (B); the test set (a+b) is the sum of the previous ones.

Detailed information on the dataset composition is shown in Table 4.2.

4.1.3 Experimental Setup

Training Settings

The CNN has been trained on the three datasets (training (A), training (B), and training (A+B)) and tested for each test set distinguished by SNRs. Training cases (A) and (B) have been tested with both test sets (a) and (b); training case (A+B) has been tested with test set (a+b). Figure 4.4 schematizes the structure of the dataset and the combinations of training and test sets employed in the experiments.

The training configurations include the use of "same"[1] padding for the input of each convolution, a "He uniform"[2] kernel initializer, a learning rate of 0.0001, ADAM [76] optimization, binary crossentropy loss function, and a split of 80% for the training phase and 20% for the validation phase. The algorithm has been implemented in the Python programming language, and the experiments have been performed using Keras [113] and Tensorflow [114] as the backend.

[1] `https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D` (accessed on 28 February 2023)

[2] `https://www.tensorflow.org/api_docs/python/tf/keras/initializers` (accessed on 28 February 2023)

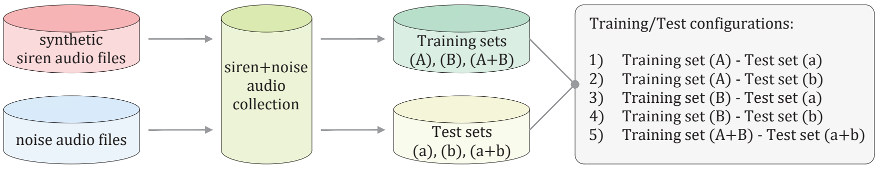

synthetic siren audio files
noise audio files
siren+noise audio collection
Training sets (A), (B), (A+B)
Test sets (a), (b), (a+b)
Training/Test configurations:
1) Training set (A) - Test set (a)
2) Training set (A) - Test set (b)
3) Training set (B) - Test set (a)
4) Training set (B) - Test set (b)
5) Training set (A+B) - Test set (a+b)

Figure 4.4: Diagram of the training/test sets combinations used in the experiments.

Performance Metrics

Performance in testing has been assessed by comparing the results in the classification task in terms of accuracy rate and F-score. Specifically, the accuracy can be considered a representative metric because the datasets are balanced between only two classes. At the same time, the F-score, representing the balancing precision and recall on the positive class, emphasizes the capability of correctly identifying examples of the positive siren class.

4.1.4 Results

The following tables summarize the results of the experiments performed under mono- and multi-scenario conditions.

Results in Mono-Scenarios

Table 4.3 shows the outcomes of the neural model trained with the training set (A) and tested on test sets (a) and (b) at the same SNRs used in training (from -15 dB to 0 dB). The performance of gammatonegrams (GTs) and log-Mel spectrograms (log-Mels) are compared. The best results are provided by the test set (a) in combination with GTs, achieving an accuracy rate equal to 95.00% and an F-score of 97.74% at -15 dB, which gradually increases along with signal-to-noise ratios. On the other hand, performance with the test set (b) and GTs decreases significantly for SNRs less than -5 dB. In all experiments, the performance of log-Mel spectrograms follows the trend of GTs, but they yield lower scores than gammatonegrams.

Table 4.4 shows the results obtained by the neural model trained with the training set (B) and tested on test sets (a) and (b) at the same SNRs used in training, comparing the GTs and log-Mel findings. Again, the model trained and tested on datasets with similar background noise frequency ranges, together with gammatone spectrograms, performs best. Considering the test set (b) and GTs, both the accuracy rate and F-score at -15 dB are equal to 99.87%

	Training (A)							
SNR (dB)	Test (a)				Test (b)			
	GTs		log-Mels		GTs		log-Mels	
	Acc (%)	F (%)	Acc (%)	F (%)	Acc (%)	F (%)	Acc (%)	F (%)
0	**100**	**100**	93.94	93.93	96.75	96.64	95.72	95.80
-5	**98.77**	**98.75**	92.69	92.58	93.63	93.19	79.04	75.38
-10	**95.01**	**94.75**	91.94	91.79	66.67	50.02	62.47	45.35
-15	**95.00**	**94.74**	91.92	91.76	50.11	0.50	57.56	33.42

Table 4.3: Results of the neural model trained with the training set (A) and tested on test sets (a) and (b) at the same SNRs used in training.

and reach 100% for the highest SNRs. As in the previous experiments, the performance of the model computed with log-Mels is lower than that of GTs.

	Training (B)							
SNR (dB)	Test (a)				Test (b)			
	GTs		log-Mels		GTs		log-Mels	
	Acc (%)	F (%)	Acc (%)	F (%)	Acc (%)	F (%)	Acc (%)	F (%)
0	86.55	84.46	67.37	57.26	**100**	**100**	98.79	98.79
-5	55.76	20.65	56.01	24.72	**100**	**100**	98.30	98.30
-10	50.00	0.00	49.19	1.45	**100**	**100**	89.41	88.39
-15	50.00	0.00	48.79	0.00	**99.87**	**99.87**	83.98	81.46

Table 4.4: Results of the neural model trained with the training set (B) and tested on test sets (a) and (b) at the same SNRs used in training.

Results in Multi-Scenarios

Table 4.5 shows the outcomes of the experiments performed with the ensemble datasets at the same SNRs used during the training. The best results are achieved by GTs, with accuracy rates between 94.97% and 99.78% and F-scores between 94.72% and 99.78% for SNRs ranging from -15 dB to 0 dB. Again, in all the tests GTs outperform log-Mels.

	Training (A+B)			
SNR (dB)	Test (a+b)			
	GTs		log-Mels	
	Acc (%)	F (%)	Acc (%)	F (%)
0	**99.78**	**99.78**	95.63	95.63
-5	**99.34**	**99.34**	94.70	94.65
-10	**98.10**	**98.07**	92.71	92.52
-15	**94.97**	**94.72**	92.00	91.76

Table 4.5: Results of the neural model trained with the training set (A+B) and tested on test set (a+b) at the same SNRs used in training.

Results in Conditions Unseen in Training

Finally, Table 4.6 presents the experimental results for emergency siren recognition in the best-performing configurations of the previous cases for SNRs unseen in training. It is noteworthy that, for all three training and testing setups, the models can generalize emergency siren recognition even under very high noise conditions, up to an SNR of -30 dB. The best outcomes can be observed with datasets (A,a) and ensemble datasets ((A+B),(a+b)).

	Training (A)		Training (B)		Training (A+B)	
SNR (dB)	Test (a)		Test (b)		Test (a+b)	
	GTs	log-Mels	GTs	log-Mels	GTs	log-Mels
	Acc (%)	Acc (%)	Acc (%)	Acc (%)	Acc (%)	Acc (%)
-20	**95.00**	91.92	93.91	83.79	**94.62**	91.63
-25	**95.00**	91.92	90.00	83.79	**94.37**	91.44
-30	**95.00**	91.92	90.00	83.79	**94.37**	91.41

Table 4.6: Results obtained with the corresponding training and test sets at SNRs not used in training.

4.1.5 Remarks

The following conclusions can be drawn based on the results of the analysis:

- In single-scenario experiments with affine background noise (Training (A)–Test (a), Training (B)–Test (b)), the models performed exceptionally well, even at low signal-to-noise ratios.
- In single-scenario experiments with non-affine background noise (Training (A)–Test (b), Training (B)–Test (a)), the models showed a loss of performance as SNR decreased.
- In multi-scenario experiments, the model demonstrated good performance even at low SNRs.
- The experiments that test the previously computed best models on datasets comprising SNRs unseen during the training show good generalization capability, especially in mono-scenario with datasets (A,a) and in multi-scenario.
- In all simulations, gammatone spectrograms have been found to be an effective acoustic representation, yielding better results than those obtained with log-Mel spectrograms.

The considerations that can be deduced from the results concern the learning mechanisms of the neural network. In particular, the high accuracy in the emergency siren recognition task in the case of affine training and test sets suggests that during the training, the model has learned the characteristics of both the siren sound and the background noise. Confirming this, in the experiments with non-affine background noise, F-score values near or equal to zero at low SNRs indicate the complete inability of the network to distinguish sirens in contexts unseen in the training phase when the traffic noise is significant. Therefore, strategies have been required to be developed to avoid model overfitting on training data, e.g., increasing their variability in terms of background noise typologies. In fact, the purpose of the research includes as a primary goal that siren sound recognition occurs in a generalized manner in several road and urbanization contexts.

Furthermore, the siren spectrograms with low-spectral background noise (A-type noise) in both mono and multi-scenario conditions showed improved results at signal-to-noise ratios unseen during the training. This aspect gives a better understanding of the patterns that make siren sounds recognizable in a time-frequency representation. Spectrograms with high harmonic components not obscured by background noise improve the pattern recognition capability of siren frames. So, strategies for noise reduction and enhancement of the fundamental and harmonic frequencies of siren tones can provide significant improvements in the emergency siren recognition task.

Finally, considerations are drawn about the effectiveness of signal representation by different acoustic features. The experiments have shown better results with gammatonegrams than with log-Mel spectrograms, but further experiments elucidated how to improve the performance of this latter. Specifically, increasing the number of frequency bins provides greater resolution of the time-frequency representation and thus allows for higher accuracy rates. At the same time, computing acoustic features not as images but as feature vectors eliminates the performance loss associated with the conversion of the grayscale image to an array. So, this initial study on emergency siren recognition also provided insight into the best ways to process the source signal. Given these assumptions, log-Mel spectrograms with an appropriate number of frequency bins have been preferred to gammatonegrams in subsequent investigations, as the triangular filter bank is computationally less onerous and more suitable for implementations in embedded devices.

4.2 CNN-based Approach with Improved Synthetic Dataset and Harmonic Filtering

Research on emergency siren recognition with convolutional neural networks has been extended based on the results of the experiments described in Section 4.1.4.

In particular, the model accuracy in recognizing siren sounds in background noise contexts similar to those used in training led to the design of strategies for reducing the overfitting of the model to the training data. To this end, the research involved creating a synthetic dataset of sirens in variegated multi-scenario conditions with different noise frequency distributions between training and test sets. The benefit of a neural model trained on multiple background noises is that it can identify and learn siren alarm patterns regardless of the characteristics of the noise in which it is immersed and thus recognize them in ever-changing and unpredictable scenarios.

In addition, the network capability to correctly identify siren frames in which the upper harmonic components of the signal are unmasked led to experimentation with the harmonic-percussive source separation technique introduced in Section 3.1.4. The role of the harmonic filter is to enhance the fundamental and harmonic frequencies of the two siren tones and reduce the background noise, producing time-frequency representations that highlight the frequency bands of the target signal.

Finally, the computational load associated with training a neural model with gammatone spectrograms in the form of images suggested exploring strategies to reduce the number of network hyperparameters, thus facilitating the implementation of the algorithm in real-time embedded systems.

In the following paragraphs, the composition of the proposed multi-scenario synthetic dataset, computation of acoustic features, details of filtering techniques applied to the audio file collection, and strategies to reduce the computational load of the algorithm without loss of performance are explained.

4.2.1 Materials and Methods

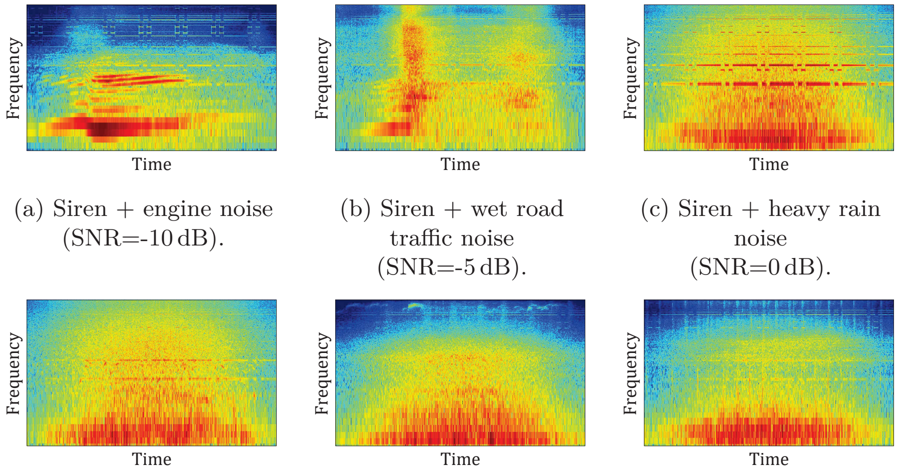
The adopted methodology begins with the creation of a synthetic dataset that consists of siren sounds mixed with multiple types of background noise. The convolutional neural network used in the previous experiments, adapted and optimized to the new input data, is then trained on this dataset, and its performance is evaluated at different signal-to-noise ratios. Gammatone spectrograms, which have been shown to perform well under high noise conditions in previous research, are used to define the baseline. The performance of this model is then compared with the outcomes of short-time Fourier transform spectrograms, which represent an algorithmic solution to calculate acoustic features at a low computational cost. The results of STFT spectrograms computed using the unfiltered dataset are improved by applying a harmonic filtering technique on the audio segments at increasing separation factors between harmonic, percussive and residual components of the audio signal. The harmonic filtered dataset that provides the best performance is employed in experiments aimed at reducing network hyperparameters. To this end, several ranges of frequency bands of the spectrograms are sliced to further reduce the size of the time-frequency representations as input to the network. Finally, the accuracy of the model is reevaluated using the sliced spectrograms, assessing the relevance of the harmonic components of the signal in the emergency siren detection task.

Datasets

A novel synthetic audio collection has been created with the same techniques, structure and numerosity outlined in Section 4.1.2. As for the previous dataset, the audio files have been equally balanced between noise recordings and simulated ambulance sirens in background noise contexts. However, this improved version includes a more variegate range of background noises. Thirty-four audio files containing a variety of environmental noises, such as car and motorcycle engines, horns, alarms, rain, human conversation, and nature sounds like birds chirping, have been downloaded from web resources [90] for a duration of over 5 hours. They have been added to siren audio files with attenuation for distance and the Doppler effect in several configurations of velocity and directions. These types of noises have been selected as they are challenging to be classified by the algorithm: frames containing tonal components may cause false positive classifications by the neural model; additionally, some noises, such as heavy rain, can overpower siren tones even at high signal-to-noise ratios, potentially resulting in false negatives.

The audio collection has been pre-processed, resulting in monophonic audio segments of 0.5-second duration at a sampling rate of 16 kHz and 32-bit encoding. Considering the numerosity of the previous audio segment collection suitable for training a neural model, this dataset has also been organized with the same criteria. The training set comprises 32 000 noise and 32 000 siren audio files (SNRs equal to -15 dB, -10 dB, -5 dB, and 0 dB), and the test set 6000 noise and 6000 siren audio files for each SNR used in the training set, with the addition of -30 dB, -25 dB, and -20 dB. Figure 4.5 shows examples of siren spectrograms in several background noises used in the experiments, each with a duration of 10 seconds and at different signal-to-noise ratios.

Four additional collections have been created from this unfiltered collection of audio files, resulting from applying the median-filtering harmonic-percussive source separation technique described in Section 3.1.4. Harmonic separation factors equal to 1, 3, 5, and 8 have been considered and applied to the unfiltered audio files to assess the extent to which residual and percussive components of traffic noise affect the siren sound. Figure 4.6 shows an example of a siren spectrogram without noise, with urban traffic noise in the unfiltered situation



(a) Siren + engine noise (SNR=-10 dB).

(b) Siren + wet road traffic noise (SNR=-5 dB).

(c) Siren + heavy rain noise (SNR=0 dB).

(d) Siren + people talking and rain noise (SNR=-5 dB).

(e) Siren + cars and seagull noises (SNR=-15 dB).

(f) Siren + percussive noise (SNR=-10 dB).

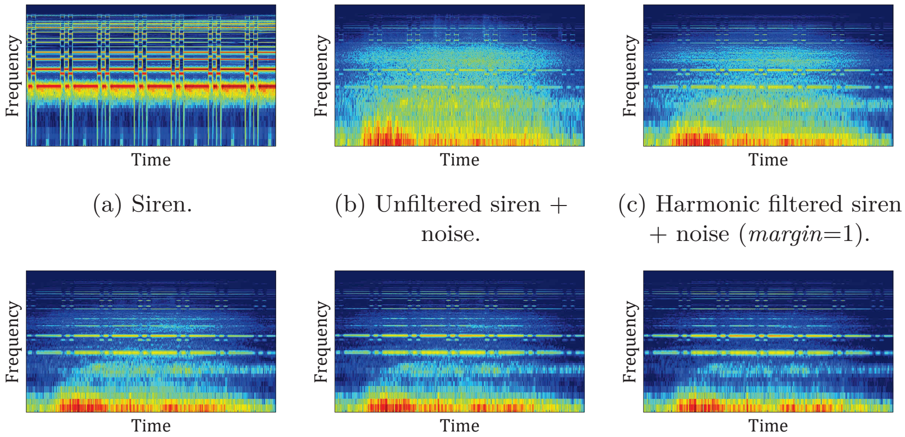
Figure 4.5: Spectrograms of sirens with different types of noises.

and after the harmonic filter application with different separation factors.

Acoustic Feature Computation

The choice of input features was informed by the findings of the previous experiments in emergency siren recognition. Gammatone spectrograms, enhancing acoustic features within the audible range, especially at low frequencies, have been shown to perform well under high background noise conditions. For this reason, they have been used to define the baseline to be reached or overcome with a low computational cost model employing STFT spectrograms as input features. The difference between a gammatone and an STFT spectrogram is that, in the former type, the frequency sub-bands of the ear have high resolution for low frequencies and widen for high frequencies, whereas the STFT spectrogram has a constant bandwidth for all frequency channels.

To reproduce similar training conditions as in previous experiments, the gammatonegrams have been integrated into 64 frequency bins using the procedure described in [92] and saved as grayscale images at a resolution of 496 × 368 pixels. STFT spectrograms have been extracted by computing discrete Fourier transforms (DFTs) on short overlapping windows using the librosa [96] library. Again, 0.5-second audio segments with a sampling rate of 16 kHz have been decomposed with an STFT characterized by a Hanning windowed signal length of 1024 and a hop size of 512 samples. Amplitude spectrograms have been then converted to the dB scale and saved as 2D arrays of size 17 × 513.



(a) Siren.

(b) Unfiltered siren + noise.

(c) Harmonic filtered siren + noise (*margin*=1).

(d) Harmonic filtered siren + noise (*margin*=3).

(e) Harmonic filtered siren + noise (*margin*=5).

(f) Harmonic filtered siren + noise (*margin*=8).

Figure 4.6: Comparison between unfiltered and harmonic filtered siren spectrograms at different separation factors.

Further investigation on acoustic features has been performed by processing the STFT spectrograms with slicing operations to keep only a specific range of frequencies and thus reduce the dimension of the input. For this purpose, the frequency bin centers have been indexed, and rows of the time-frequency matrix above a specific value have been eliminated. The spectrogram reduction criteria have been based on the ranges of values that include the fundamental frequencies and upper harmonic components of the two tones of the Italian siren, having frequencies equal to 392 Hz (G4) and 660 Hz (E5).

STFT spectrograms have been subject to slicing operations to create four datasets:

1. the first with a maximum frequency of 700 Hz, which includes the two fundamental tones (or first harmonics);
2. the second with a maximum frequency of 1400 Hz, which includes the first upper octaves (or second harmonics);
3. the third with a maximum frequency of 2800 Hz, which includes the second upper octaves (or fourth harmonics);
4. the fourth with a maximum frequency of 5600 Hz, which includes the third upper octaves (or eighth harmonics).

Figure 4.7 shows the spectrum of a siren audio file according to the Italian law with the indication of the main harmonic components.

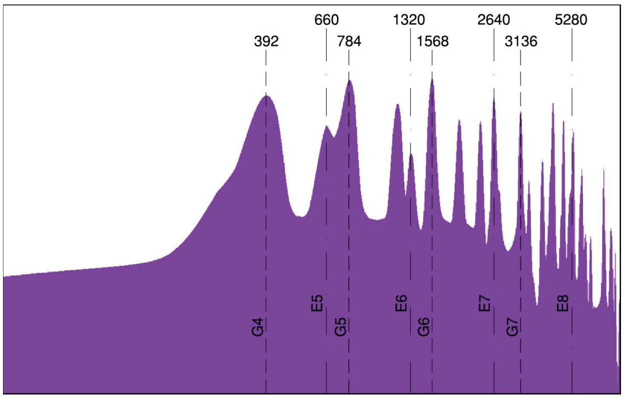


Figure 4.7: Spectrum of a siren audio file according to the Italian law.

CNN Architecture

The convolutional neural network described in Section 4.1.1 has been taken as the neural architecture for the baseline. For training the model at low computational cost, this network has been adapted to take as input 2D arrays of specific shapes (*num_row*, *num_columns*, *num_channels*). An optional dropout layer has been inserted between the two convolutional layers in each block repetition to prevent overfitting, controlled by the *drop_rate* parameter specifying the proportion of activations to be dropped. In the experiments, the dropout rate was tuned with a grid search for the optimal rate, which was found to be between 0% and 15%.

4.2.2 Experiments

The convolutional neural network has been trained with the settings specified in Section 4.1.3, using the accuracy rate as the performance metric.

The experiments have been conducted in the order dictated by the workflow, specifically:

1. The first experiments compared the performance of gammatone spectrograms, which have defined the baseline, and STFT spectrograms computed from the unfiltered audio collection.

2. The second round of simulations involved STFT spectrograms computed from the harmonic-filtered datasets with increasing separation factors to evaluate the improvements provided by this technique in the emergency siren recognition task.

3. Finally, the third group of experiments aimed to reduce the network hyperparameters by decreasing the size of the input features. The harmonic

Dataset	Features	Input size (channels-last)	Number of hyperparameters
Unfiltered	GTs	(368, 496, 1)	460897
Unfiltered	STFTs	(513, 17, 1)	25057
Harmonic (*margin*=1,3,5,8)	STFTs	(513, 17, 1)	25057
Harmonic sliced fundamental	STFTs	(45, 17, 1)	6177
Harmonic sliced octave I	STFTs	(90, 17, 1)	8097
Harmonic sliced octave II	STFTs	(180, 17, 1)	11617
Harmonic sliced octave III	STFTs	(359, 17, 1)	18657

Table 4.7: Comparison of the number of network hyperparameters.

STFT spectrograms that performed best in the previous round of simulations have been subjected to slicing operations in the various frequency ranges related to the harmonic components of the siren signal. The resulting datasets have been input into the neural network, and the accuracy rates have been compared with the previous results.

Table 4.7 illustrates the datasets employed in the experiments, the type of acoustic features computed from them, and the input tensors size that determines the total number of network hyperparameters.

4.2.3 Results and Discussion

In this section, experimental results in the order in which they have been performed and their evaluations are presented and discussed.

The first round of experiments compares the performance of models computed with gammatonegrams (GTs) and STFT spectrograms (STFTs) extracted from the unfiltered audio collection. Table 4.8 presents the testing accuracy obtained with GTs and STFTs at decreasing SNRs. The network trained and tested with GTs achieves an average accuracy of 96.27% and validates the results of the previous research, confirming that GTs are excellent features for sound event detection in conditions of significant noise. STFTs reach an average accuracy of 92.20%, providing comparable outcomes to GTs only at SNR equal to 0 dB. For lower SNRs, their performance decreases significantly as the background noise level increases.

Unfiltered Dataset		
	GTs	STFTs
SNR (dB)	Acc (%)	Acc (%)
0	98.16	97.62
-5	97.63	95.63
-10	96.11	90.28
-15	93.18	85.26
Avg acc (%)	**96.27**	92.20

Table 4.8: Comparison between GTs and STFTs testing accuracy.

The second part of the experiments involves STFT spectrograms computed from filtered audio files with the aim of improving the previous outcomes. Table 4.9 presents the testing accuracy achieved with harmonic datasets with increasing separation factors (*margin*). The best results are obtained by applying a *margin* parameter equal to 3, which returns an average accuracy of 96.72%. Also, the harmonic dataset with a *margin* of 1 achieves a comparable average accuracy equal to 96.35%. For higher separation factors (5 and 8), a gradual loss of performance at low SNRs is denoted. These results indicate that the residual components may not have a significant impact, as experiments with low *margin* parameters of 1 and 3 demonstrate a successful decomposition of the spectrogram, allowing to keep the harmonic components. On the other hand, the decreased performance of harmonic datasets with separation factors of 5 and 8 suggests that the process of separating the residual components also removes part of the harmonic components of the target signal, which therefore loses its sharpness.

	Harmonic Datasets			
	margin=1	*margin*=3	*margin*=5	*margin*=8
SNR (dB)	Acc (%)	Acc (%)	Acc (%)	Acc (%)
0	98.97	99.08	98.70	98.53
-5	98.65	98.76	98.28	97.94
-10	96.29	96.55	94.78	94.38
-15	91.50	92.50	87.45	85.63
Avg acc (%)	96.35	**96.72**	94.80	94.12

Table 4.9: STFTs testing accuracy at different separation factors.

The third group of simulations concerns the spectrograms that achieved the best accuracy in the previous experiment (harmonic dataset with a *margin* coefficient equal to 3), processed by slicing operations. Table 4.10 shows that accuracy rates improve with wider frequency ranges. For a frequency range up to 5600 Hz (third octave), the performance is excellent for SNRs of 0 dB and -5 dB and comparable to the experiments without slicing. The average accuracy equal to 94.02% is a good result, considering that a network hyperparameters reduction of 25% compared to the full spectrogram has been applied. Overall, these findings confirm that the energy contribution of harmonic components has a relevant role in the siren detection task.

The generalization capability of the models that provided the best results is evaluated by testing their performance at signal-to-noise ratios unseen during the training phase, equal to -20 dB, -25 dB, and -30 dB. Table 4.11 reports the accuracy rates comparing the unfiltered dataset with GTs, and the harmonic dataset (*margin* equal to 3) with STFTs. The results confirm the robustness of gammatone spectrograms under high noise conditions and highlight the comparable performance of STFTs when using the harmonic filtering technique.

Harmonic Sliced Datasets				
	1° harm (fundamental)	2° harm (octave I)	4° harm (octave II)	8° harm (octave III)
SNR (dB)	Acc (%)	Acc (%)	Acc (%)	Acc (%)
0	92.87	98.50	98.73	99.42
-5	83.67	93.19	94.79	98.28
-10	76.28	85.10	88.70	93.69
-15	72.42	81.78	83.83	84.67
Avg acc (%)	81.31	89.64	91.51	**94.02**

Table 4.10: STFTs test accuracy at different frequency ranges.

	Unfiltered Dataset	Harmonic Dataset
	GTs	STFTs
SNR (dB)	Acc (%)	Acc (%)
-20	90.15	88.07
-25	89.34	86.30
-30	89.32	86.25

Table 4.11: Comparison between GTs and harmonic STFTs testing accuracy for SNRs unseen in training.

4.3 Conclusions

This research confirmed the effectiveness of convolutional neural networks in the general field of sound event detection and, specifically, for emergency siren recognition. The robustness of these algorithms is largely due to their ability to take as input the time-frequency representations of the signals, which provide detailed information about the acoustic properties of the sound.

One of the main challenges of this research involved the selection of the most suitable acoustic features for the task to be used as input for the convolutional neural network. In this regard, gammatone spectrograms have proven to be a very effective choice, as they closely approximate the auditory response of the human ear, enhancing low frequencies. However, gammatone spectrogram computation is intensive and can pose a challenge for real-time applications. This issue can be overcome by using STFT spectrograms in combination with a harmonic-percussive source separation technique. STFT spectrograms reduce the computational load of the algorithm, and the harmonic filtering enhances the tonal components of the siren signal, providing comparable performance to gammatonegrams.

The studies described in this section have been carried out using synthetic datasets. Subsequent research has focused on developing strategies to compute, from synthetic or non-task-related datasets, robust models that are capable of recognizing emergency siren sounds in real-world environments.

Chapter 5

Few-Shot Learning for Emergency Siren Detection

Few-shot learning is a branch of deep learning that bases the learning of new concepts on few examples from each class instead of requiring a large amount of labeled data like traditional deep learning algorithms. Although supervised learning with one or a limited number of examples has been a topic of interest for several years in computer vision [115, 116], its application to audio signal processing has only been a recent development. Among the different few-shot techniques [117, 118], task-invariant embedding methods have proven well-suited for audio classification. The central idea behind these approaches is that an embedding function is learned from a large-scale training set. This prior knowledge is employed to directly discriminate between similar and dissimilar instances of new classes during the inference by leveraging similarity measures.

In the fields of sound event detection and acoustic scene classification, the literature has explored several approaches to address the challenge of deep learning with limited audio data. The study in [119] is a pioneering work that focuses on transfer learning strategies in comparison with prototypical networks [51], using a fixed taxonomy in training and testing rather than through the concept of meta-learning. In [120, 121], an analysis of meta-learning models applied to acoustic event detection is presented, demonstrating the superiority of these methods in generalizing to new audio events, compared to supervised solutions based on fine-tuned convolutional neural networks. The effectiveness of five different few-shot learning methods, enhanced by an attentional similarity module to detect transient events for sound event recognition, are discussed in [122].

The promising results obtained from few-shot learning have sparked further research in the audio field, leading to the development of strategies to extend the application to more complex and challenging tasks, such as multi-label classification [123], rare sound event detection [124], continual learning [125], unsupervised and semi-supervised learning [126], and sound localization [127].

The application of few-shot learning techniques to an open-set sound event detection problem is presented in [128]. The authors evaluate and compare several few-shot metric learning methods to identify keywords in audio files. The problem is formulated as a binary classification, with keywords belonging to the positive set and all other words constituting the negative set. The results demonstrate the capability of the method to generalize to unseen languages and its potential in audio domains beyond speech.

Recently, the use of few-shot neural networks has been extended to a variety of audio recognition domains, including keyword spotting in devices with a vocal interface [129], bioacoustic event detection [130], sound anomaly detection in industrial machinery [131], speaker identification and activity recognition [132], automatic drum transcription [133], and recognition of underwater acoustic signals in impulsive noise environments [134].

These examples highlight the versatility and potential of few-shot learning in audio applications. An unexplored area is related to emergency siren detection and deserves to be studied properly. In fact, a system able to learn acoustic features from a reduced amount of data can be extremely useful for a feasible and customizable emergency vehicle detection device embedded in vehicles.

5.1 Proposed Approach

This section presents a solution for emergency siren detection, built on prototypical networks and compared with a convolutional baseline. First, the few-shot metric learning strategies are illustrated, then an overview of the proposed emergency siren detection (ESD) workflow is given, and finally, the neural architectures employed in this work are described.

5.1.1 Few-Shot Metric Learning

Few-shot learning aims to solve a classification task given a target domain built on few examples. Hence, it is necessary to adopt strategies to create a model with generalization capability and quick adaptation to new domains. The few-shot metric learning approach usually employs a training set different from the test set. Training is performed in a *C-way K-shot* fashion, where C represents the number of classes (ways) and K the instances (shots) of each class employed in each iteration. This training method mimics the configurations that will arise at inference time, preferring large datasets and a high number of iterations to learn how to discern between different classes given only few input examples. The robustness of the model is evaluated with a metric-based function that returns the similarity measure between instances of the same class.

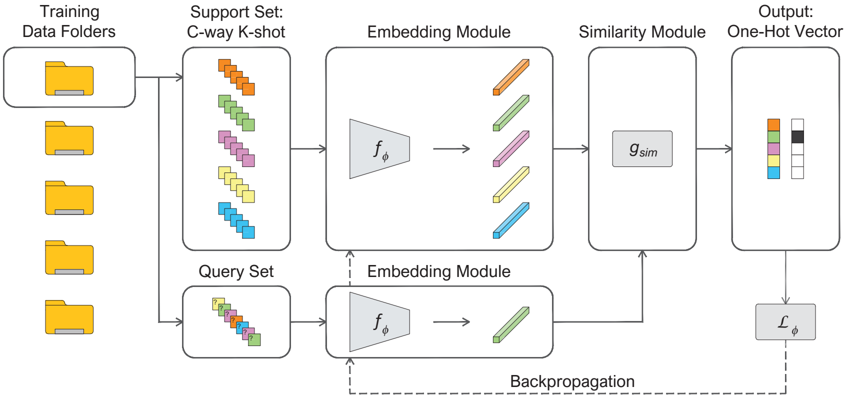


Figure 5.1: Episodic training procedure (*5-way 5-shot* example) of few-shot metric learning.

Episodic Training

A common strategy in few-shot metric learning algorithms is the episodic training [135]. For this purpose, the training set has been organized in F folders, each containing a set of M labeled samples $T = \{(x_1, y_1), \dots, (x_m, y_m)\}_{m=1}^{M}$. The feature vectors $x_m \in \mathbb{R}^D$ have a fixed dimension D, and the labels $y_m \in \{1, \dots, L\}$ represent the L classes. A folder is selected in each iteration (episode), and a mini-batch of data is sampled randomly. A part of the mini-batch constitutes the support set, composed of $C \times K$ examples $S = \{(x_1, y_1), \dots, (x_i, y_i)\}_{i=1}^{C\times K}$ where feature vectors $x_i \in \mathbb{R}^D$ and labels $y_i \in \{1, \dots, C\}$, with $C \leq L$. The remaining samples define the query set, composed of $C \times q$ examples $Q = \{(x_1, y_1), \dots, (x_j, y_j)\}_{j=1}^{C\times q}$. The embedding function f_ϕ incorporates the support set S and query set Q into a lower-dimensional hypothesis space. Due to the reduction in tensor dimensionality and a meaningful representation in the transformed space, similar examples relative to the task are close, while dissimilar examples are easily differentiable. The metric function g_{sim} performs the similarity measure between the support and the query set embeddings, hence the definition of few-shot metric learning. Different metric functions can be used, fixed, or with learnable parameters. The training is repeated until the minimization of the loss function $\mathcal{L}_\phi$, representing the prediction error of the samples in Q conditioned by the comparison with the representation in S.

Figure 5.1 illustrates the episodic training procedure, from the random selection of support and query sets to the embeddings generation, and finally to the similarity measure in an iterative process to minimize the loss function.

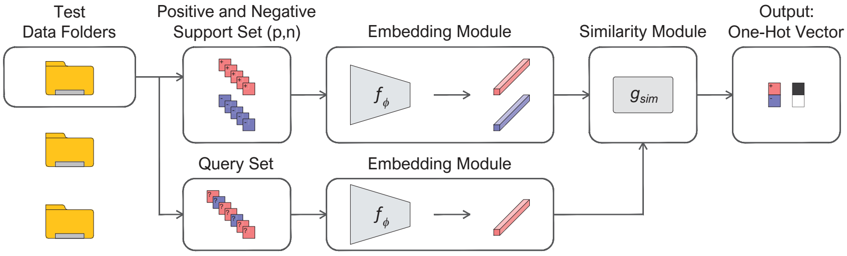


Figure 5.2: Open-set testing procedure of few-shot metric learning.

Open-Set Testing Procedure

The traditional few-shot testing method assumes that only $U < F$ folders containing $R < L$ classes are used in training. The embedding model must be optimized to transfer the knowledge learned to classify samples of the $(L - R)$ classes in the $(F - U)$ folders. Again, the support set S of size $C \times K$ and the query set Q with the instances to be classified are constructed. The problem thus posed is "closed-set" because samples belonging to C well-defined classes are classified.

This work aims to detect samples of only one target class given few labeled instances for the embedding generation. Classifying samples of other categories, whose characteristics need not be necessarily known, is not of interest. The problem of discerning between one positive class and the negative rest is called "open-set" and is reduced to a binary classification task. At inference time, a positive support set $P = \{(x_1, y_{pos}), \ldots, (x_i, y_{pos})\}_{i=1}^{p}$ consisting of a small number of labeled target samples and a negative support set $N = \{(x_1, y_{neg}), \ldots, (x_j, y_{neg})\}_{j=1}^{n}$ containing examples that do not belong to the category of interest are randomly selected. The remaining instances compose the positive and negative query sets. As in training, the embedding function f_ϕ incorporates the support and query sets; the similarity module g_{sim} compares the embeddings and returns the probability that the query sample belongs to the positive class. The algorithm must have generalization capabilities to find the similarity between the embeddings of the (unlabeled) target samples and those computed from the positive support set, discriminating from the negative class.

Figure 5.2 shows the open-set testing procedure, from the random selection of the positive and negative support sets to the embeddings generation, and finally to the similarity measure between the query and the positive support embeddings.

5.1.2 Overview of the ESD Workflow

Contributions of the Work

The main purpose of this research is to provide a comprehensive analysis of few-shot metric learning capabilities in the real-world application of emergency siren detection. A general overview of the objectives of the work is illustrated as follows.

- An analysis of the performance of prototypical networks is presented, employing datasets extracted from three distinct audio collections with different characteristics. The impact of the dataset features and the training/test example combinations on the performance of each prototypical model is thoroughly investigated.

- The proposed method applies the knowledge learned by the few-shot models in discriminating similar or dissimilar instances from a specific audio collection to detect a target sound belonging to a different dataset. In this case, the target is the ambulance siren sound recorded in real-world contexts according to the procedures described in Section 3.1.2, to be recognized using only a few examples for the prototypical embedding computation.

- The focus of this study is on a real-world application. For this reason, the analysis is designed to provide valuable information on the most suitable placement of the sound recording sensor, comparing eight microphone positions inside and outside the cabin.

- The effectiveness of the few-shot technique is validated by comparing it to a convolutional baseline with and without fine-tuning using few examples of the target domain. In addition, noise filtering strategies are evaluated to enhance the performance of the analyzed models.

Pipeline of the Proposed Method

In the following, the pipeline of the proposed approach is outlined in detail.

1. Best few-shot models computation: the raw audio has been pre-processed and transformed into log-Mel spectrograms, organized, and given as input to prototypical networks. Because the episodic training in several *C-way K-shot* combinations returned different performance in the test phase, the model that obtained the best output has been saved for the next step. This procedure has been repeated for three datasets extracted from diverse audio file collections, obtaining three best-performing *C-way K-shot* prototypical models.

2. Best few-shot models evaluation: siren and noise audio recordings have been pre-processed and transformed into log-Mel spectrograms, split over eight audio channels corresponding to eight sensor positions. The best-performing *C-way K-shot* models obtained in step (1) have been used to make predictions about new data taken from the recordings for the ESD task, repeating the procedure for each audio channel.

3. Analysis of prototypical outcomes: the experiments performed in step (2) have provided classification scores distinguished by training model and recording channel, so the best-performing dataset and sensor locations have been selected to compute the baseline models.

4. Baseline models computation: the raw audio belonging to the collection that provided the best prototypical results has been pre-processed, and then log-Mel spectrograms have been computed and organized in a suitable way as input to the CNN employed for the baseline. The network has been trained in two ways: without domain adaptation and with fine-tuning by using combinations of few data taken from the target dataset.

5. Baseline models evaluation: the recordings have been tested on the baseline models computed in step (4), and the results have been compared with few-shot best outcomes (step (3)).

6. Harmonic filtered experiments: after applying the harmonic-percussive source separation technique described in Section 3.1.4, log-Mel spectrograms have been extracted again from the recordings. The inference operations in steps (2) and (5) have been repeated and compared to the experiments with unfiltered data.

Figure 5.3 presents the block diagram of the proposed approach for emergency siren detection.

5.1.3 Neural Network Architectures

Prototypical Network

The peculiarity of a prototypical network is the generation of a representation μ_c of each class, called prototype. Given $S_c = \{(x_1, y_1), \ldots, (x_c, y_c)\}_{c=1}^{K}$ (the support set belonging to the c-class), the prototype μ_c is the mean vector of the embedded support samples, computed through the embedding function f_ϕ with learnable parameters ϕ:

$$\mu_c = \frac{1}{K} \sum_{(x_c, y_c) \in S_c} f_\phi(x_c). \tag{5.1}$$

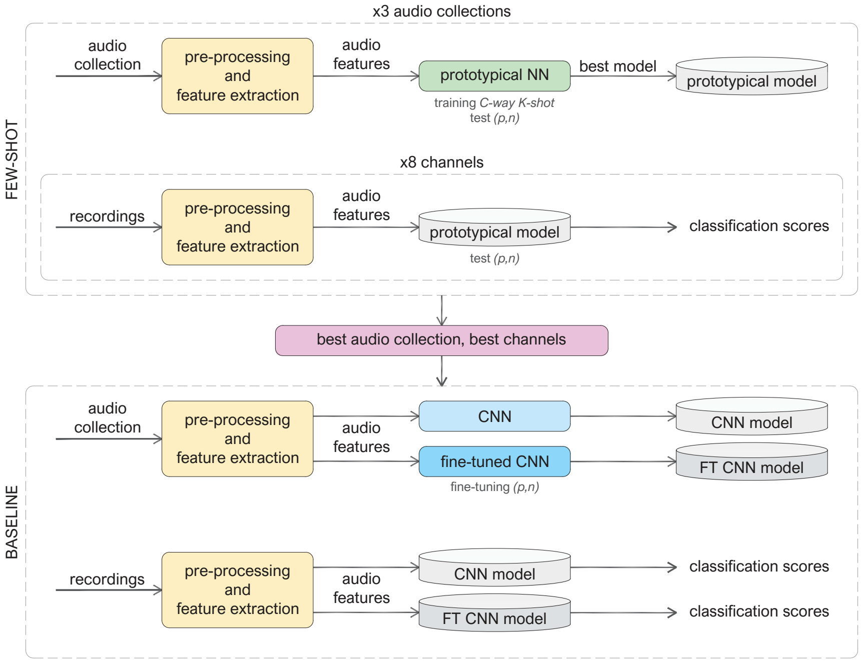


Figure 5.3: The block diagram of the proposed approach for emergency siren detection.

Given a similarity function g_{sim}, represented by the squared Euclidean distance d, the prototypical network computes the relationship between a query sample $x_q \in Q$ and the prototypes via a softmax over distances in the embedding space:

$$p_\phi(y = c \mid x_q) = \frac{\exp(-d(f_\phi(x_q), \mu_c))}{\sum_{c'} \exp(-d(f_\phi(x_q), \mu_{c'}))}, \tag{5.2}$$

where $p_\phi(y = c \mid x_q)$ represents the normalized probability distribution that x_q belongs to the c-class. The training process is done by minimizing $\mathcal{L}(\phi)$ (the negative log-probability of the true c-class via stochastic gradient descent):

$$\mathcal{L}(\phi) = -\log p_\phi(y = c \mid x_q). \tag{5.3}$$

Consequently, in each training episode, the model learns the similarity between query samples and the corresponding prototypes belonging to randomly chosen C-classes.

Prototypical architecture, illustrated in Figure 5.4, is based on the convolutional block defined in [136], consisting of a convolutional layer with a 3 × 3 kernel and 64 filters, a batch-normalization layer, and a ReLU activation layer.

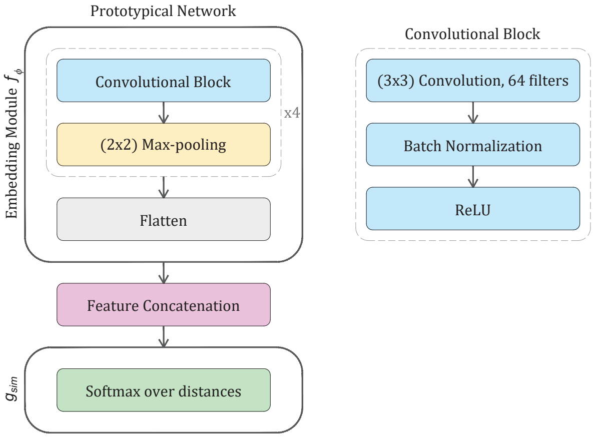


Figure 5.4: Architecture of a prototypical network.

The sequence of four convolutional blocks, each followed by a 2×2 max-pooling layer, composes the embedding function f_ϕ. In the learning process, the feature maps of the support set prototypes and the query set are flattened and concatenated to be compared through the similarity function g_{sim}.

Convolutional Neural Network

The neural architecture used for the baseline is the convolutional neural network designed and optimized for emergency siren recognition, as described in Section 4.1.1.

5.2 Materials and Methods

This section presents the audio file collections used in the experiments. Then, the description of the pre-processing operations on audio data, the feature computation, the network training settings, and the performance metrics are provided.

5.2.1 Datasets

For training, three audio collections have been selected. Non-task-related datasets have been extracted from the Spoken Wikipedia Corpora [137] and UrbanSound8K [83] audio databases. Additionally, the synthetic audio file collection described in Section 4.2.1 and called "A3Siren-Synthetic" has been employed. For inference, the dataset has been computed from an audio collection of recordings performed onboard the equipped vehicle mentioned in Section 3.1.2, called "A3Siren-Recordings."

Spoken Wikipedia Corpora

The Spoken Wikipedia Corpora (SWC) is an audio collection of volunteer readers of Wikipedia articles. The English-language corpus consists of 1339 audio files totaling about 395 hours of recordings from various readers. The audio files, characterized by ogg format, are monophonic with a sampling rate of 44.1 kHz and 32-bit encoding. They are associated with metadata, some of which have textual annotations aligned to the words (start and end time in milliseconds).

In an SWC audio file, a specific word pronounced by a reader represents the target class. A *C-way K-shot* training episode has been performed by taking a support set with C classes (different words) and K instances per class (examples of the same word) contained within a folder (corresponding to a reader). An additional number of instances per class composed the query set. The episodic training has been set up with *C-way K-shot* ranging from *2-way 1-shot* to *10-way 10-shot* and a query set of 16 instances per class. Thus, readers with at least 2 target words repeated 26 times have been kept, considering only audio files with temporally aligned words. Out of 208 readers and more than 2000 classes, 75% have been assigned to the training, 10% to the validation, and 15% to the test. Audio segments have been selected by taking a 0.5-second window in the center of each instance. At inference time, the knowledge acquired to recognize the similarity between the same words spoken by a reader has been transferred to detect a target word (positive set p) discriminating from various random samples of non-target words (negative set n) within an audio file of a reader assigned to the test.

UrbanSound8K

UrbanSound8K (US8K) includes 8732 urban sounds of duration less than or equal to 4 seconds divided into 10 classes and 10 folders, totaling about 8.75 hours. All audio files are in wav format, about 92% of which are stereo and the remaining 8% mono, with sampling rates ranging from 8 kHz to 192 kHz and encoding between 4-bit and 32-bit. The excerpts have been taken from field recordings available at [90].

The training, validation, and test folders have been split with a 7:1:2 ratio, leaving the distribution of the audio files within each folder unchanged and applying standardization operations. In a training episode, a folder (US8K fold) has been selected. C classes (environmental sounds) and K instances per class (examples of the same class also from different audio files) ranging from *2-way 1-shot* to *10-way 10-shot* have been included in the support set. The query set consisted of the remaining instances not used in the support set. The same positive and negative set definition criteria described for the SWC dataset have been applied for the testing phase. In this case, training and testing have been performed with a fixed taxonomy because, during the inference, classes already seen in training have been used. It is not a limitation at this stage because the final purpose is not to evaluate the performance of few-shot techniques on the US8K dataset but to apply different models to siren recordings and compare the outcomes.

A3Siren-Synthetic

A3Siren-Synthetic[1] (A3S-Synth) is the "improved" audio collection described in Section 4.2.1 to train and test the convolutional neural network implemented for the siren/noise classification task. To carry out the few-shot experiments, the training data have been organized into 16 folders, each containing 2000 noise and 2000 siren audio segments. Because only two classes are present, the episodic training has been performed with *C-way K-shot* equal to *2-way 1-shot*, *2-way 5-shot*, and *2-way 10-shot*. The A3S-Synth training collection has been split into 75% for training and 25% for validation. For evaluating the models, 300 noise and 300 siren audio segments for each SNR have been randomly selected from the A3S-Synth test collection and organized in separate folders. This audio collection has also been employed to train the CNN model representing the baseline as a comparison to the performance of few-shot techniques. In the experiments with the CNN, the organization of the original A3S-Synth training audio files has been left unchanged. Again, the split has been 75% for training and 25% for validation.

A3Siren-Recordings

A3Siren-Recordings[1] (A3S-Rec) is the audio collection recorded during the acquisition campaign performed in May 2021 with the equipment and procedures described in Section 3.1.2. Out of 18 audio tracks, only recordings containing siren events have been considered for the experiments, identifying 6 tracks for about 3 hours and 39 minutes. After the standardization and labeling operations, audio files were split into 0.5-second segments. All the siren segments have been assigned to the positive set. Given the wide availability of urban traffic noise, samples preceding each siren event for the duration of one minute have been chosen and attributed to the negative set. Figure 5.5 illustrates the proposed audio selection method, and Table 5.1 shows the A3Siren-Recordings composition.

Siren and noise audio segments have been organized in two ways: (i) in separate folders, each containing the samples selected in the individual record-

[1] `https://github.com/michelacantarini/Few-Shot-Emergency-Siren-Detection` (accessed on 28 February 2023)

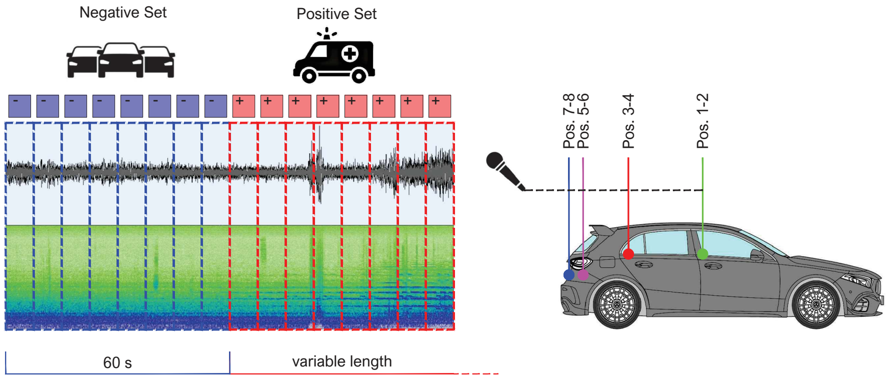


Figure 5.5: A3S-Rec positive and negative audio data selection method.

Recording	Siren (s)	Noise (s)	Location
20210506-1652	53	60	construction site
20210506-1714	19	60	coastal road
20210507-1421	38.5	60	shopping center
20210507-1426	38	60	residential suburb
20210507-1536	26	60	high-speed road
20210507-1640	25	60	city center
Total	199.5	360	

Table 5.1: A3S-Rec audio file composition and recording environment.

ings; and (ii) in a single folder in which all audio files are mixed regardless of the original track. The first strategy allows the algorithm to be evaluated in identifying the siren sound in a specific background noise context. In a single recording, the siren detection task is performed by contextualizing the target sound within a background noise where the siren gradually appears with a low signal level. The second data arrangement aims to discriminate between siren sounds and traffic noises not belonging to the same recording. In this way, the capability of the network to recognize siren sounds in several background noises is assessed. In both cases, the siren detection task is performed separately in the eight channels of the audio tracks.

5.2.2 Experimental Setup

Prototypical networks have been trained with the episodic method in several (C, K) configurations. The validation process has also been performed in an episodic way at the end of a chosen number of training episodes. After each validation step, the average loss was compared with the previous value, and the learned parameters of the model that returned the best performance were stored for testing. The training setup consisted of a learning rate equal to 0.001, Adam [76] optimizer, and 60 000 episodes, of which 1000 were for validation and every 5000 for training, saving the model that performed best in the validation stage.

Then, the baseline model compared with the few-shot methods has been computed with and without fine-tuning for domain adaptation. First, the CNN was trained for 100 epochs with a learning rate equal to 0.001, a decay rate of 0.1 every 30 epochs, and Adam optimizer, saving the model that performed best in validation. Then, the previously trained model was adapted with several (p, n) instances of the new task, according to the (p, n) combinations employed for prototypical support embeddings. Because few data have been chosen for domain adaptation, the network was prone to quick overfitting. For this reason, all the convolutional layers have been frozen, and only the two last linear layers have been re-trained with a low learning rate and a small number of epochs. Training has been performed with 20 epochs, Adam optimizer, decreasing learning rate between 0.0001 and 0.00001, and the early-stopping regulated by the training loss.

Performance has been evaluated in terms of the area under precision-recall curve (AUPRC). Both for prototypical networks and the fine-tuned baseline, the random selection of positive and negative instances could have affected the results because of the variability of the sample characteristics, even within the same class. So, the experiments have been repeated ten times with different random (p, n) examples, and the AUPRC scores have been averaged to generalize the performance over the available data.

5.3 Experiments and Discussion

5.3.1 Few-Shot Model Analysis

The first step of the experiments consisted of finding the best-performing (C, K, p, n) combinations for the classification of a target word (SWC), an environmental sound (US8K), and an ambulance siren (A3S-Synth) as illustrated in Figure 5.6. For all three datasets, the experimental results correlate better scores with higher (C, K, p, n): the best prototypical models are the *10-way 10-shot* cases for the SWC and US8K datasets, the *2-way 10-shot* for the A3S-Synth dataset. The A3S-Synth dataset presents the best score with an AUPRC equal to 0.99 employing $(p, n) = (5, 50)$ and 0.96 averaged over all the (p, n) combinations. In the following, the experimental results are analyzed for each dataset by varying (C, K, p, n).

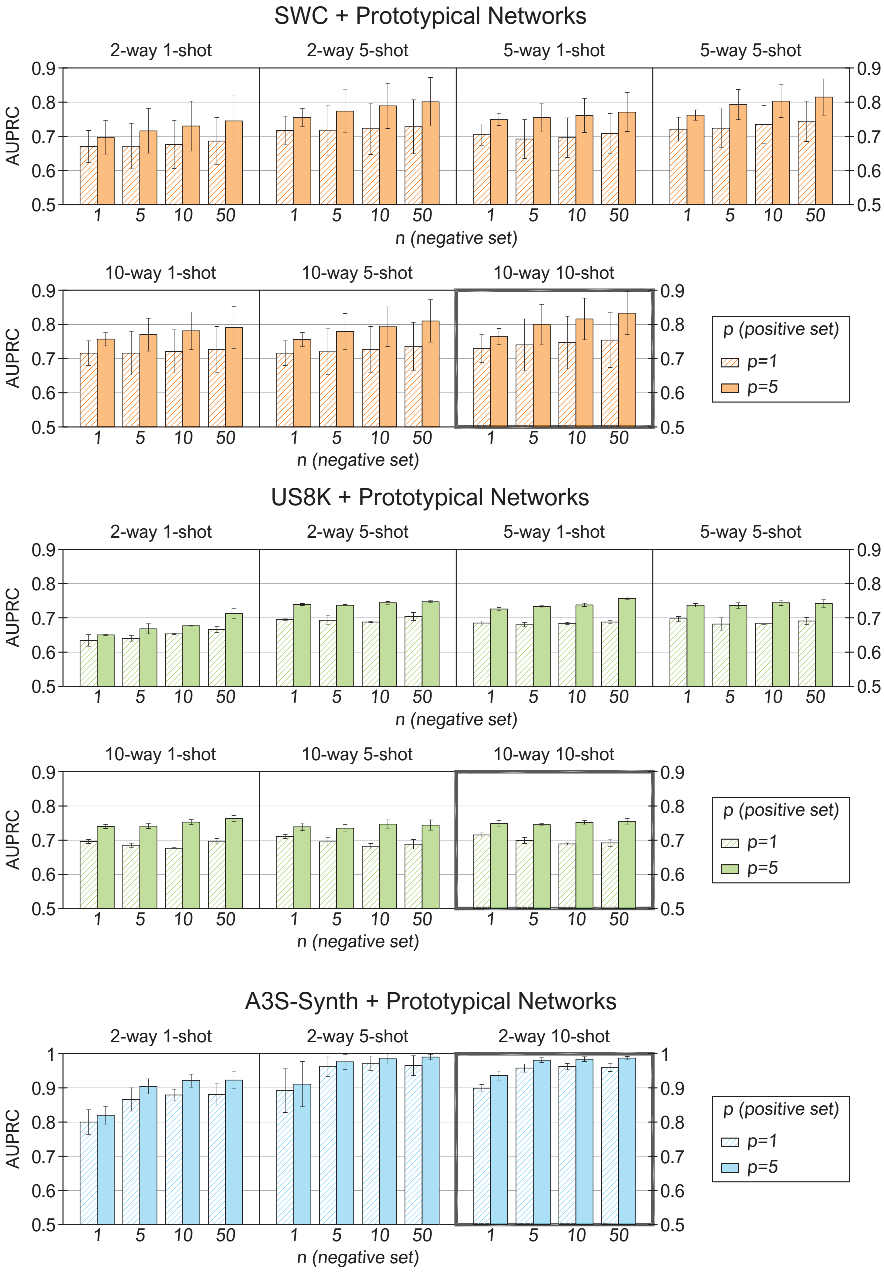


Figure 5.6: AUPRC results for prototypical networks trained and tested with SWC, US8K, and A3S-Synth in several (C, K, p, n) combinations.

Analysis of Results Varying *C* Ways

Optioning multiple values of C ways is allowed only for multiclass datasets, so the first analysis interests the SWC and US8K experiments by fixing $C = (2, 5, 10)$ and averaging the AUPRC scores over all the (K, p, n) configurations. In both datasets, many C ways improve the average scores: multiclass training expands the prior knowledge of the model by increasing the discriminative capability among many classes of sounds and facilitating the discernment between examples belonging to new classes at the inference stage. A possible explanation for the lower performance of US8K is that it comprises only ten classes, and few classes in training are limiting for constructing a model with high generalization capability.

Analysis of Results Varying *K* Shots

The investigation proceeds with the evaluation of the impact of different K values within the same *C-way* setting. For the SWC, many K shots return better performance: a higher number of examples creates a prototype that more closely collects the patterns of the original class. Also, for the A3S-Synth, many K shots improve the AUPRC score. The *1-shot* condition is the least effective, and at the same time, there is no substantial improvement between the *5-shot* and *10-shot* settings. Finally, the dataset that shows the least benefit in using multiple K shots in the prototype generation is the US8K. These results are related to the characteristics of the dataset. Using a dataset with clean recordings and low interclass variability, such as the SWC, more instances for the embedding computation create a more representative feature vector that facilitates the mapping between the positive queries and the corresponding support prototype. On the other hand, the low inter-class variance of support examples with significant background noise levels, stationary sounds, or other sounds not adequately represented by a 0.5-second time window, such as for the US8K and A3S-Synth datasets, could penalize the prototype representation with many examples.

Analysis of Results Varying *(p,n)* Instances

The outcomes of individual *C-way K-shot* cases by varying (p, n) are now explored. For all datasets and (C, K, n) configurations, the performance of prototypical networks with $p = 5$ is better than $p = 1$ because the prototype created by one example is not always representative of an entire class. On the other hand, increasing n does not provide univocal results for all datasets. With the SWC and A3S-Synth, an improvement in AUPRC scores as n increases is noticed, and the $n = 50$ case returns the best results in all simulation contexts. Hence, using more examples to create the negative support prototype enhances the capability of the network to classify the positive instances correctly. However, this is not the case with the US8K dataset, which shows a similarity between n at inference time and K in the training phase, where increasing shots do not produce a more robust prototypical representation.

5.3.2 Siren Detection with Prototypical Networks

Evaluation within Individual Recordings

This analysis has assessed the ESD task within individual recordings composed of audio segments with only traffic noise and others with additional sirens gradually arising from the background. In this way, the models have been tested to promptly identify the target sound in contexts where the background noise is significant, variable, and unpredictable. In the experiments, the performance of the recording sensors has been analyzed for each microphone position inside and outside the passenger compartment to evaluate which setup can provide the best response.

The results of the best prototypical models trained with the SWC, US8K, and A3S-Synth datasets, respectively, and tested on the individual audio tracks of the A3S-Rec dataset, are presented. For the sake of conciseness, complete results for each (p, n) combination are reported only for channels 7–8 that achieved the best performance, as shown in Table 5.2. For channels 1–6, the AUPRC scores averaged over all (p, n) combinations are shown in Table 5.3.

	SWC		US8K		A3S-Synth	
(p, n)	ch7	ch8	ch7	ch8	ch7	ch8
(1,1)	0.68 ± 0.07	0.68 ± 0.07	0.69 ± 0.07	0.68 ± 0.08	0.69 ± 0.10	0.70 ± 0.10
(5,1)	0.75 ± 0.06	0.72 ± 0.07	0.74 ± 0.06	0.73 ± 0.07	0.75 ± 0.12	0.76 ± 0.10
(10,1)	0.75 ± 0.06	0.73 ± 0.06	0.74 ± 0.05	0.73 ± 0.07	0.75 ± 0.11	0.75 ± 0.09
(1,5)	0.71 ± 0.08	0.69 ± 0.10	0.67 ± 0.08	0.66 ± 0.10	0.77 ± 0.07	0.76 ± 0.09
(5,5)	0.77 ± 0.09	0.77 ± 0.09	0.73 ± 0.08	0.73 ± 0.09	0.73 ± 0.08	0.73 ± 0.09
(10,5)	0.78 ± 0.09	0.76 ± 0.09	0.74 ± 0.08	0.73 ± 0.09	0.83 ± 0.08	0.84 ± 0.07
(1,10)	0.71 ± 0.11	0.70 ± 0.11	0.68 ± 0.10	0.67 ± 0.10	0.68 ± 0.10	0.67 ± 0.10
(5,10)	0.80 ± 0.09	0.78 ± 0.10	0.75 ± 0.09	0.74 ± 0.10	0.85 ± 0.06	0.86 ± 0.07
(10,10)	0.80 ± 0.10	0.80 ± 0.10	0.75 ± 0.09	0.75 ± 0.09	0.86 ± 0.08	0.89 ± 0.05
(1,50)	0.72 ± 0.09	0.72 ± 0.09	0.69 ± 0.09	0.67 ± 0.09	0.79 ± 0.09	0.79 ± 0.08
(5,50)	0.81 ± 0.09	0.79 ± 0.11	0.74 ± 0.10	0.73 ± 0.10	0.87 ± 0.05	0.88 ± 0.06
(10,50)	**0.82 ± 0.09**	**0.82 ± 0.10**	**0.77 ± 0.08**	0.76 ± 0.09	0.89 ± 0.05	**0.90 ± 0.05**
avg	0.76 ± 0.09	0.75 ± 0.09	0.72 ± 0.08	0.71 ± 0.09	0.80 ± 0.08	0.81 ± 0.08

Table 5.2: AUPRC of the best SWC, US8K, and A3S-Synth prototypical models tested on individual recordings of channels 7–8 of the A3S-Rec dataset.

The three models provide outcomes with the same trend in all audio channels: in most cases, the AUPRC scores increase along with (p, n). As expected, the scores obtained from the A3S-Synth-trained model are better than the SWC ones, followed by the US8K model outcomes. The A3S-Synth dataset provides the best performance in the combination $(p, n) = (10, 50)$ with an AUPRC score of 0.90 at channel 8, and among the several models, it benefits most

Model	ch1	ch2	ch3	ch4	ch5	ch6
SWC	0.71 ± 0.11	0.71 ± 0.10	0.71 ± 0.11	0.73 ± 0.10	0.67 ± 0.13	0.68 ± 0.12
US8K	0.70 ± 0.13	0.68 ± 0.11	0.70 ± 0.12	0.69 ± 0.12	0.67 ± 0.13	0.66 ± 0.13
A3S-Synth	0.70 ± 0.11	0.72 ± 0.09	0.71 ± 0.10	0.74 ± 0.11	0.68 ± 0.13	0.67 ± 0.11

Table 5.3: Average AUPRC of the best SWC, US8K, and A3S-Synth prototypical models tested on individual recordings of channels 1–6 of the A3S-Rec dataset.

from multiple support examples. From the comparison of the AUPRC values between different microphone positions, the sensors behind the license plate (positions 7–8) demonstrated the best performance, followed by those inside the passenger compartment (positions 1–4) and finally in the trunk (positions 5–6).

Evaluation with Internal Labeling

In the previous simulations, audio segments have been annotated by listening to the audio signals recorded by the sensors behind the license plate and applying the same label to all channels, as the audio data of the external microphones show the two tones of the ambulance siren sooner than the other positions. The behavior of the sensors inside the passenger compartment has also been investigated, focusing on the influence of the sound attenuation of the cockpit. As the chassis is made of soundproofing material, it acts as a barrier to the entry of the siren sound when its level is below the transmission loss of the enclosure at the siren tones frequencies. This fact results in a shorter duration of siren sound events in the internal recordings than in the external ones. To indirectly assess the influence of cockpit attenuation in the ESD task, the experiments have been repeated after revising the annotations for internal channels 1, 2, 3, and 4. Figure 5.7 shows spectrograms of the same siren occurrence recorded by sensors inside the passenger compartment and behind the license plate. In the example, the siren sound in the first 5 seconds of the internal recording is attenuated; thus, the internal labeling considers this audio segment as noise.

Tables 5.4, 5.5, and 5.6 present test results on the data corresponding to channels 1-2-3-4 of the internally labeled A3S-Rec dataset, using the best SWC/US8K/A3S-Synth prototypical models.

For all the datasets and (p, n) settings, the outcomes of the recordings with internal labeling present equal or better AUPRC scores than those with external annotations, with an average relative percentage increment of up to 7%. The performance improvement with the internal labeling is correlated to the noise class attribution of uncertain siren events resulting from cockpit sound attenuation and internal car noise. Whereas the impact of the attribution of not clearly identifiable siren events to the noise class is reflected in higher scores, the algorithm exhibits delayed responsiveness in the siren recognition. Thus,

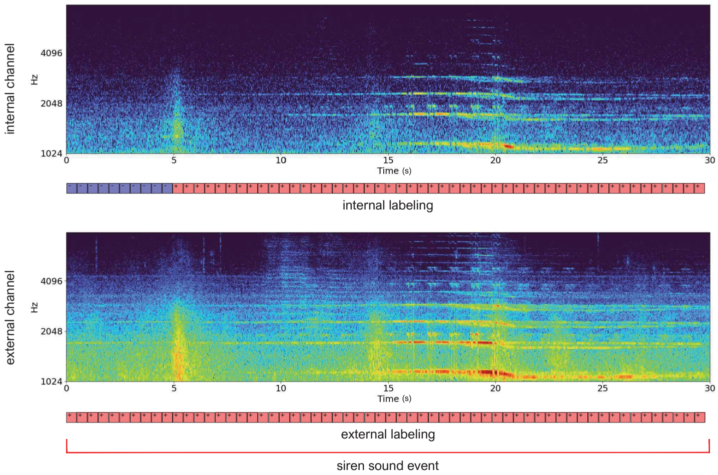


Figure 5.7: Spectrograms of siren recordings acquired by sensors inside (top) and outside (bottom) the passenger compartment with different labeling criteria for the internal channel.

external labeling has been taken as a reference for an unbiased comparison of the effectiveness of the acquisition sensors.

Among the internal sensors, the microphone at position 4 provides the best scores. One possible reason is that ambulances often approached the equipped car from the same direction of travel during the acquisition campaign. Thus, the alarm sound first impacted the rear, incurring less reflection from the source to the recording sensors on the back. The better response of the sensor at position 4 compared to the specular one could be due to the operator sitting near position 3, which represented an absorption surface for the incoming sound.

$(p,\ n)$	ch1	ch2	ch3	ch4
(1,1)	0.71 ± 0.09	0.71 ± 0.08	0.69 ± 0.10	0.71 ± 0.09
(5,1)	0.74 ± 0.09	0.73 ± 0.10	0.75 ± 0.10	0.75 ± 0.07
(10,1)	0.75 ± 0.09	0.76 ± 0.08	0.75 ± 0.10	0.75 ± 0.09
(1,5)	0.71 ± 0.11	0.71 ± 0.10	0.69 ± 0.10	0.75 ± 0.09
(5,5)	0.79 ± 0.09	0.78 ± 0.08	0.79 ± 0.10	0.79 ± 0.09
(10,5)	0.79 ± 0.08	0.79 ± 0.08	0.79 ± 0.09	0.80 ± 0.09
(1,10)	0.72 ± 0.10	0.72 ± 0.10	0.70 ± 0.09	0.75 ± 0.09
(5,10)	0.79 ± 0.09	0.78 ± 0.09	0.79 ± 0.09	0.80 ± 0.09
(10,10)	0.81 ± 0.07	0.82 ± 0.07	0.81 ± 0.08	0.82 ± 0.08
(1,50)	0.71 ± 0.10	0.72 ± 0.09	0.71 ± 0.10	0.77 ± 0.09
(5,50)	0.80 ± 0.08	0.80 ± 0.08	0.80 ± 0.09	0.81 ± 0.09
(10,50)	0.81 ± 0.07	**0.82 ± 0.07**	0.81 ± 0.08	**0.82 ± 0.10**
avg	0.76 ± 0.09	0.76 ± 0.08	0.76 ± 0.09	0.78 ± 0.09

Table 5.4: AUPRC of the best SWC prototypical model tested on the A3S-Rec dataset, considering internal labeling for the recordings of channels 1–4.

(p, n)	ch1	ch2	ch3	ch4
(1,1)	0.69 ± 0.12	0.67 ± 0.10	0.67 ± 0.11	0.68 ± 0.09
(5,1)	0.72 ± 0.13	0.68 ± 0.10	0.70 ± 0.12	0.72 ± 0.11
(10,1)	0.70 ± 0.13	0.69 ± 0.10	0.70 ± 0.12	0.72 ± 0.11
(1,5)	0.71 ± 0.14	0.68 ± 0.13	0.68 ± 0.11	0.69 ± 0.11
(5,5)	0.72 ± 0.13	0.72 ± 0.14	0.73 ± 0.13	0.76 ± 0.12
(10,5)	0.75 ± 0.14	0.73 ± 0.14	0.75 ± 0.13	0.77 ± 0.12
(1,10)	0.70 ± 0.14	0.69 ± 0.13	0.69 ± 0.14	0.69 ± 0.12
(5,10)	0.74 ± 0.15	0.72 ± 0.16	0.74 ± 0.14	0.76 ± 0.12
(10,10)	0.76 ± 0.13	0.74 ± 0.15	0.76 ± 0.13	0.78 ± 0.12
(1,50)	0.70 ± 0.14	0.68 ± 0.13	0.69 ± 0.13	0.70 ± 0.12
(5,50)	0.75 ± 0.14	0.72 ± 0.15	0.75 ± 0.14	0.78 ± 0.12
(10,50)	0.77 ± 0.14	0.74 ± 0.15	0.76 ± 0.14	**0.79 ± 0.11**
avg	0.73 ± 0.14	0.71 ± 0.13	0.72 ± 0.13	0.74 ± 0.11

Table 5.5: AUPRC of the best US8K prototypical model tested on the A3S-Rec dataset, considering internal labeling for the recordings of channels 1–4.

(p, n)	ch1	ch2	ch3	ch4
(1,1)	0.66 ± 0.10	0.65 ± 0.08	0.65 ± 0.12	0.68 ± 0.11
(5,1)	0.70 ± 0.12	0.68 ± 0.11	0.72 ± 0.10	0.75 ± 0.10
(10,1)	0.70 ± 0.13	0.69 ± 0.10	0.71 ± 0.11	0.75 ± 0.11
(1,5)	0.69 ± 0.11	0.68 ± 0.08	0.67 ± 0.10	0.70 ± 0.10
(5,5)	0.76 ± 0.12	0.77 ± 0.09	0.76 ± 0.10	0.80 ± 0.10
(10,5)	0.77 ± 0.11	0.77 ± 0.09	0.78 ± 0.10	0.81 ± 0.09
(1,10)	0.69 ± 0.11	0.69 ± 0.07	0.69 ± 0.09	0.71 ± 0.09
(5,10)	0.77 ± 0.11	0.78 ± 0.08	0.78 ± 0.10	0.81 ± 0.09
(10,10)	0.79 ± 0.11	0.81 ± 0.07	0.80 ± 0.08	0.82 ± 0.09
(1,50)	0.69 ± 0.11	0.69 ± 0.10	0.70 ± 0.10	0.71 ± 0.10
(5,50)	0.77 ± 0.12	0.79 ± 0.09	0.79 ± 0.10	0.81 ± 0.10
(10,50)	0.80 ± 0.10	0.81 ± 0.08	0.80 ± 0.09	**0.83 ± 0.10**
avg	0.73 ± 0.11	0.74 ± 0.09	0.74 ± 0.10	0.77 ± 0.10

Table 5.6: AUPRC of the best A3S-Synth prototypical model tested on the A3S-Rec dataset, considering internal labeling for the recordings of channels 1–4.

Evaluation Across All Recordings

In this set of experiments, the few-shot techniques have been assessed across all the recordings. The detection in individual recordings can be interpreted as identifying the background noise perturbation produced by the siren signal. On the other hand, the detection across all recordings represents a more challenging task performed in several acoustic contexts, varying both in terms of sound intensity and spectral content. Channels 4, 7, and 8 have been considered in the experiments as they provided the best results in the analysis within individual recordings. The experiments have been conducted with support sets composed of $p \in \{10, 20, 50\}$ and $n = 50$ to evaluate the influence of increasing positive support examples.

Table 5.7 presents the prototypical results across all the recordings acquired by microphones in positions 4-7-8. Again, with a fixed $n = 50$, a more significant number of p improves the scores, and the best outcomes are obtained by the A3S-Synth model with data belonging to channel 7.

Training set	*(p,n)*	ch4	ch7	ch8
SWC	(10,50)	0.59 ± 0.05	0.72 ± 0.07	0.76 ± 0.04
	(20,50)	0.62 ± 0.05	0.77 ± 0.02	0.78 ± 0.03
	(50,50)	0.64 ± 0.05	0.81 ± 0.03	0.80 ± 0.02
US8K	(10,50)	0.59 ± 0.05	0.65 ± 0.03	0.66 ± 0.03
	(20,50)	0.62 ± 0.04	0.67 ± 0.02	0.69 ± 0.02
	(50,50)	0.63 ± 0.02	0.67 ± 0.01	0.69 ± 0.01
A3S-Synth	(10,50)	0.67 ± 0.03	0.82 ± 0.02	0.82 ± 0.02
	(20,50)	0.73 ± 0.05	0.84 ± 0.02	0.83 ± 0.03
	(50,50)	0.73 ± 0.03	**0.86 ± 0.02**	0.85 ± 0.02

Table 5.7: AUPRC of the best prototypical models tested across all recordings of channels 4-7-8 of the A3S-Rec dataset.

In addition, noise reduction effects have been considered by applying the harmonic-percussive source separation technique [93] to the A3S-Rec dataset. Table 5.8 presents prototypical results across all the recordings acquired by microphones in position 4-7-8 after harmonic filtering with a separation factor $\beta_h = 3$. An appreciable improvement provided by the filtering operations is observed, especially for the external channels. The best AUPRC scores are attributed to the A3S-Synth model with data belonging to channel 8.

Training set	*(p,n)*	ch4	ch7	ch8
SWC	(10,50)	0.67 ± 0.05	0.86 ± 0.01	0.88 ± 0.01
	(20,50)	0.71 ± 0.02	0.86 ± 0.01	0.87 ± 0.01
	(50,50)	0.71 ± 0.02	0.86 ± 0.00	0.88 ± 0.01
US8K	(10,50)	0.60 ± 0.05	0.76 ± 0.02	0.80 ± 0.03
	(20,50)	0.62 ± 0.04	0.78 ± 0.02	0.82 ± 0.02
	(50,50)	0.62 ± 0.03	0.78 ± 0.02	0.83 ± 0.02
A3S-Synth	(10,50)	0.70 ± 0.04	0.85 ± 0.02	0.90 ± 0.01
	(20,50)	0.75 ± 0.02	0.87 ± 0.01	**0.91 ± 0.01**
	(50,50)	0.75 ± 0.02	0.86 ± 0.01	**0.91 ± 0.00**

Table 5.8: AUPRC of the best prototypical models tested across all recordings of channels 4-7-8 of the A3S-Rec dataset with harmonic filter.

Analyzing the experiments with the harmonic filtered dataset, the results show that channels 7–8 yield better performance than channel 4. One possible explanation is that the high noise in the external recordings is easily separated and assigned to the percussive and residual components, emphasizing the harmonic siren sound. In cleaner internal recordings, filtering operations do not improve appreciably over unfiltered audio data. Additionally, using more positive examples often does not lead to better outcomes. The reason is that in the filtered condition, spectrograms highlight the harmonic content of the signal, and therefore even few instances can create a representative prototype.

The AUPRC scores of the filtered audio data of channel 7 are lower than those of channel 8. Despite being in specular positions, the microphone at position 7 is located on the left side of the license plate. The corresponding recordings may be affected by noises with tonal components from cars in the other direction of travel, which explains the loss of performance with the filtered audio data.

5.3.3 Siren Detection with Baseline

The last experiments involved classifying the audio files of the A3S-Rec dataset with the baseline computed by the CNN described in Section 4.1.1, using the A3S-Synth dataset for the training. The CNN performance has been evaluated across all the recordings with and without fine-tuning for domain adaptation. For a comparison with prototypical networks, the same (p, n) combinations of instances used to construct the support embeddings have been employed to update the weights of the two last linear layers.

Table 5.9 presents the outcomes of the baseline model without fine-tuning tested across all the recordings of A3S-Rec (channels 4-7-8) in unfiltered and harmonic filtered conditions.

Filtering	ch4	ch7	ch8
no	0.60	**0.65**	**0.65**
harmonic	0.62	0.64	0.64

Table 5.9: AUPRC of the baseline model without fine-tuning across all the recordings of channels 4-7-8 of the A3S-Rec dataset.

Although the results do not differ significantly, the best scores are attributed to the external channels in the unfiltered conditions, with an AUPRC equal to 0.65. The reason is the affinity between source and target domains, as the synthetic siren audio files have been generated simulating siren alarms immersed in urban traffic noise in the outdoor environment. On the other hand, inference on filtered data shows a slight decrease in performance at channels 7-8. Because the training was conducted on unfiltered data and the filtering accentuates any harmonic components, generic tonal sounds recorded by the external sensors may be confused with the siren alarm.

Table 5.10 illustrates the results of the baseline model with fine-tuning, again in unfiltered and harmonic filtered conditions.

The analysis of the fine-tuned baseline results mirrors the trend of prototypical AUPRC scores with the A3S-Synth model, shown in Tables 5.7 and 5.8. In

Filtering	$(p,\ n)$	ch4	ch7	ch8
no	(10,50)	0.45 ± 0.07	0.78 ± 0.04	0.80 ± 0.02
	(20,50)	0.65 ± 0.03	0.81 ± 0.01	0.81 ± 0.01
	(50,50)	0.70 ± 0.03	**0.84 ± 0.02**	0.83 ± 0.01
harmonic	(10,50)	0.57 ± 0.07	0.82 ± 0.01	0.84 ± 0.02
	(20,50)	0.71 ± 0.04	0.83 ± 0.01	0.86 ± 0.01
	(50,50)	0.73 ± 0.03	0.85 ± 0.01	**0.88 ± 0.02**

Table 5.10: AUPRC of the baseline model with fine-tuning in several (p, n) combinations across all the recordings of channels 4-7-8 of the A3S-Rec dataset.

both unfiltered and harmonic filtered conditions, many (p, n) instances for fine-tuning improve the classification performance. Again, the effectiveness of the noise reduction technique is proven by the best results obtained with filtered data belonging to the external channels. For the internal channel, fine-tuning with only 10 positive examples decreases the performance of the baseline without domain adaptation. In this case, few positive examples affected by cockpit attenuation and rapid model overfitting lead to erroneous learning of the siren class.

This aspect shows an additional advantage of prototypical networks in the low-data regime. Whereas the convolutional neural network used for the baseline has been trained with few epochs to reduce the problem of overfitting on the fine-tuning data, for prototypical networks, this excessive adaptation does not affect the results due to the distance-based metrics, as investigated in [119].

In Table 5.11, the relative percentage increase of the few-shot achievements with respect to (CNN + fine-tuning) is presented.

Filtering	(p, n)	ch4	ch7	ch8
no	(10,50)	48.8%	5.7%	3.0%
	(20,50)	12.8%	4.6%	2.6%
	(50,50)	4.4%	1.8%	2.0%
harmonic	(10,50)	21.6%	4.1%	6.6%
	(20,50)	6.0%	4.3%	4.8%
	(50,50)	2.5%	1.0%	4.0%

Table 5.11: AUPRC relative percentage increase from fine-tuned baseline to best prototypical score across all the recordings of channels 4-7-8 of the A3S-Rec dataset.

In almost all cases, the most significant increments occur in the combination $(p, n) = (10, 50)$ and decrease with higher p values. This fact indicates that by increasing the (p, n) examples, AUPRC scores of the fine-tuned baseline approximate the few-shot outcomes. However, prototypical networks demonstrate their superior efficacy because they perform equally well with a very limited amount of support instances. In addition, the improvement with the lowest number of data used in the $(p, n) = (10, 50)$ combination is more evident in the case of the internal microphone, meaning that the few-shot solution performs better than the (CNN + fine-tuning) when the mismatch between training and testing conditions is high.

5.3.4 Remarks

Summarizing the findings of the experiments from an algorithmic point of view, higher values of (C, K, p, n) are correlated with better scores. Multiclass training helps to expand the prior knowledge of the model and increase its discriminative capability among many sound classes, facilitating discernment between examples belonging to classes unseen during the training. In addition, the numerosity of the instances contributes to the creation of a prototype that more faithfully collects the patterns of the original class. According to this principle, using multiple examples to create positive and negative support prototypes at the inference stage can improve the network performance in classification. However, one aspect that should not be overlooked is the characteristics of the dataset used in training. The robustness of the model depends on intra- and interclass variability, background noise levels, stationarity of sounds and their duration. Additionally, a training set with affinities with the target domain can aid in the classification task. A significant advantage over convolutional neural networks in the low data regime is also related to the overfitting aspect, which in prototypical networks does not affect the results due to distance-based evaluation metrics.

Experiments were conducted with relation networks [136] but did not perform as well as prototypical networks. The difference in performance is attributed to the method used to generate embeddings, particularly considering the high noise level in the recordings. In the relation embedding method, feature maps are summed element-wise, amplifying the noise representation over the siren. On the other hand, the advantage of the prototypical embedding method is the noise and siren feature maps averaging and the consequent redistribution of the noise among all frequencies.

Finally, from an installation perspective, this research has provided insights into the implementation of emergency vehicle detection systems embedded in cars. The key findings are outlined as follows:

- The optimal location for the acquisition sensor is behind the license plate outside the vehicle. This position is less susceptible to cabin attenuation and provides a quick response to siren sound detection. However, weatherproof sensors are necessary due to the exposure to the elements.
- High external noise levels can affect siren detection. Therefore, incorporating a noise reduction filter, such as the one proposed in this study, can improve the performance of external sensors.

- Sensors inside the passenger compartment can be used with deployment benefits. Despite the disadvantages of cockpit soundproofing, people talking, or the sound system, internal microphones in a weather-protected environment are more cost-effective and require less maintenance than external ones.
- In situations where an ambulance is approaching a car in the same direction of travel, the most common and dangerous scenario, the rear of the vehicle represents the best microphone placement.

In conclusion, based on the extensive experiments conducted, prototypical networks have been shown to be a reliable and resilient method for detecting emergency sirens. In particular, few-shot methods can be applied to fine-tune algorithms that can be customized for different car models. With an extremely limited number of recordings made on board a specific type of vehicle, pre-trained neural models for the emergency siren detection task can be employed without the need for adaptation between source and target domains. Traditional fine-tuned convolutional models with a small number of instances have achieved slightly lower performance than prototypical networks, which increased with the number of instances used. Evaluations of the most suitable algorithm for the task to be undertaken must therefore be made on a case-by-case basis, in accordance with the amount of data available, their quality, and by screening different solutions.

Chapter 6

A Novel Prototype of Emergency Vehicle Detection System

As a culmination of the algorithmic investigations regarding emergency siren detection, this study illustrates the design of a complete prototype of an emergency vehicle detection system. This solution focuses on real-time monitoring of the acoustic scenario and understanding the driver's behavior by employing audio and video techniques based on deep learning. The system leverages sound recognition algorithms to detect the approaching of an emergency vehicle from the sound of its siren. When this happens, the intelligence of the system monitors the driver's gaze and evaluates his/her awareness using computer vision algorithms. The integration of these technologies into a commercial vehicle, the creation of new datasets, and the challenges encountered are described as follows.

6.1 Architecture of the Prototype

The proposed emergency vehicle detection system is designed to interact directly with the driver in case of an approaching emergency vehicle. The main objectives expected from such a system are to constantly monitor the occurrence of emergency vehicles, check if the driver is aware of the approaching vehicle, and provide an audio or visual warning if the driver appears unaware.

Emergency vehicle detection relies exclusively on computational audio processing, taking advantage of automatic siren recognition algorithms. On the other hand, driver awareness can be verified by using computer vision algorithms to extract key information from gaze activity. Rapid movements directed toward the left, rear, and right mirrors are of particular interest, representing the driver's search for the source of sound and indicating the awareness of the context. Driver gestures have been studied in a preliminary experimental phase, finding that drivers often direct their gaze to the mirror with small head movements when they hear a siren coming from behind the car.

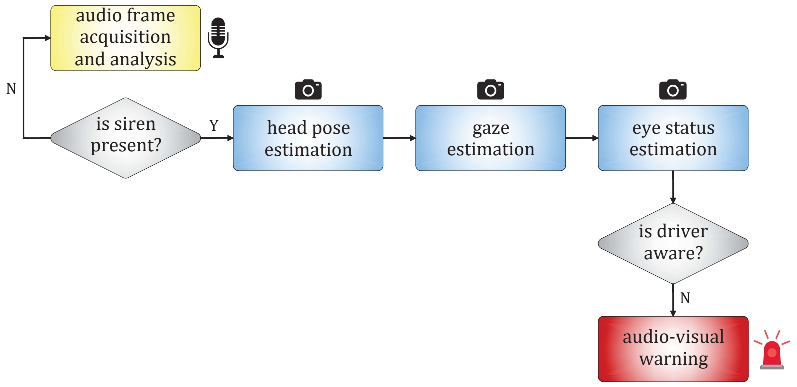


Figure 6.1: Flow diagram of the expected behavior of the system.

The device falls under modern advanced driver-assistance systems developed to support the driver in potentially dangerous situations. It does not meet high or full driving automation requirements under Levels 4 and 5 of the industry standard [138]. These levels refer to scenarios where the car can operate without human presence or intervention, such as pulling over to the side of the road or slowing down. However, the device is intended to assist the driver by providing suggestions on how to respond to hazardous situations.

The flow diagram of the system is depicted in Figure 6.1. The audio acquisition device detects the siren sound in real time, which then activates the camera to track the driver's awareness, focusing on the head pose, gaze orientation, and eye status sequentially. If the driver is found to be unaware of the emergency vehicle, the system issues an audio-visual alert.

6.1.1 Hardware Architecture

The proposed prototype requires a set of hardware components described as follows:

- Audio sensors to monitor the external acoustic scene.
- Image sensors directed toward the driver's face.
- A computational unit for processing the data acquired from the sensors in real time, running the classification algorithms and the glue logic software designed to implement the flow diagram in Figure 6.1.
- A head-up display (HUD) to alert the driver when required by the context.

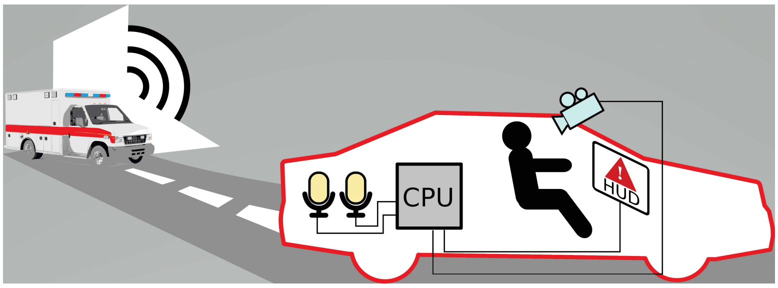


Figure 6.2: Overview of the hardware components of the emergency vehicle detection prototype.

All these devices have been installed on a research vehicle, a Mercedes A-Class, provided by a project partner supporting this study. The car has also been used to collect the datasets needed to train and test the algorithms. An overview of the hardware components involved with the prototype is shown in Figure 6.2.

The software is designed to be run on a single computing device that is efficient and compact to fit the limited space and power supply of the car. The device must have the sufficient computing power to enable real-time execution of both audio and video detection algorithms. Off-the-shelf components have been selected to create a working prototype quickly and with minimal additional effort. An x86 machine has been chosen due to its ability to host a graphics processing unit (GPU) to accelerate deep learning and image processing algorithms while still operating with low power consumption. Although any x86 personal computer could work, the Intel NUC currently represents the smallest form factor for x86 computers that can accommodate an external GPU, such as the GTX 1650 GPU. This small GPU is specifically designed to fit the NUC, and the maximum power consumption of the system is 60 W in the worst case. During the prototyping stage, a power inverter was used to convert the 12 V DC outlet of the car into a 230 V AC source to power the equipment. The NUC also features USB and HDMI connectors and is compatible with any GNU/Linux distribution, allowing for convenient software development.

ECM8000 omnidirectional measurement condenser microphones have been chosen as acoustic sensors. The microphones have been placed according to the setup illustrated in Section 3.1.2, and previous studies in Chapter 5 provided the advantages and disadvantages of each installation. Microphones inside the passenger compartment are unsuitable for recording external sounds as they may pick up interference from conversations or radio. Also, the cabin insulation can attenuate the signals. Recording sensors in the trunk are simultaneously affected by cabin soundproofing and mechanical component noise. Placing the sensor externally, such as behind the license plate, is optimal for detecting sounds from outside, especially from the back, where the driver may have

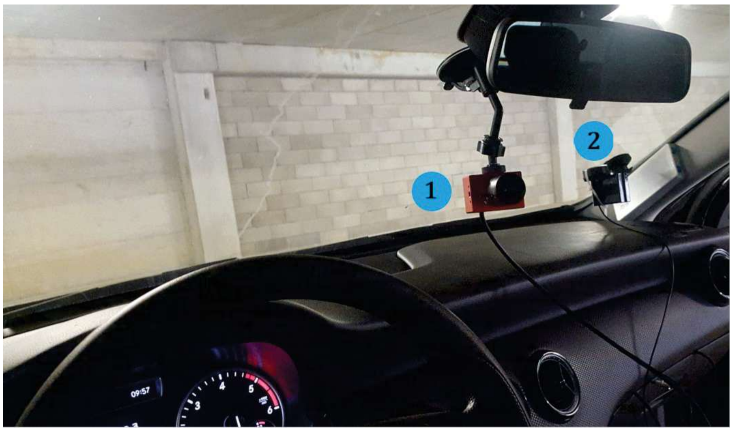


Figure 6.3: Detail of vision system, with the dashboard of the prototype car fitted with (1) the DVS camera and (2) the GNSS receiver.

difficulty perceiving an incoming emergency vehicle. An eight-channel Roland Octa-Capture soundboard has been used to capture sound with the NUC, as a maximum of eight microphones are sufficient to cover all these positions.

For the evaluation of vision sensors, two distinct technologies have been considered: a standard RGB camera and a dynamic vision system (DVS), also known as event camera. The standard RGB camera is the IDS UI-3160CP-C-HQ, a USB 3.0 camera equipped with a 2/3" global shutter CMOS sensor PYTHON 2000 from Onsemi. This camera boasts a full resolution of 2.3 MP (1920 × 1200 pixels) and a frame rate of up to 165 fps. As for the DVS, the DAVIS 346 camera has been evaluated, which can provide both grayscale frames and event data, including location, polarity, and timestamp. Both cameras are synchronized with a pulse-per-second signal generated from an external GPS receiver, based on the u-blox NEO-M8 GNSS device with an external antenna. The position of the camera and GNSS are depicted in Figure 6.3.

6.1.2 Software Architecture

The system architecture of the prototype involves executing several tasks in parallel, which are accomplished by running parallel threads with distinct functionality. Python has been selected as the programming language to implement the software architecture due to its ability to support multi-threading, create flexible graphic user interfaces (GUI), run on multiple operating systems, and bind to popular deep learning, audio processing, and image-processing libraries.

The main process includes the GUI and audio-video processing tasks. The GUI is created using the Kivy[1] library and resembles a car dashboard. Once

[1] https://kivy.org (accessed on 28 February 2023)

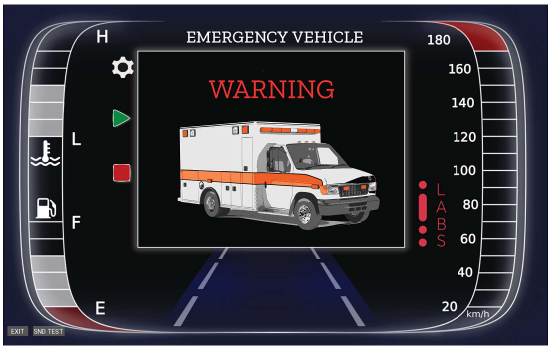


Figure 6.4: The graphic user interface of the EVD prototype.

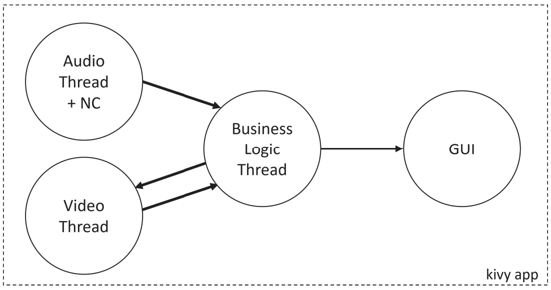


Figure 6.5: Overview of the software nodes involved of the EVD prototype.

started, it displays the detection status and provides a visual alert when a siren is detected, as shown in Figure 6.4. The audio thread employs the `sounddevice` python library and registers a callback function to process audio frames. The callback function calls a previously trained `torch`[2] neural model to detect sirens in traffic and noisy signals.

When the siren is detected, a signal is sent to the video thread to indicate the presence of a siren. At this point, the video processing pipeline evaluates if the user's gaze is directed towards the mirrors (i.e., left, right, and rear view mirrors) to ensure eyes are looking for signs of an emergency vehicle. A similar signal is sent when the siren disappears. An overview of the software architecture is provided in Figure 6.5, where NC stands for neural classification.

6.1.3 Deep Learning Algorithms

The siren sound recognition and driver face monitoring systems, which are part of the prototype, are based on deep learning algorithms. Pre-trained models adapted to the target task have been used for consistency between audio and visual recognition techniques. Convolutional models trained on synthetic data

[2] `https://pytorch.org` (accessed on 28 February 2023)

and fine-tuned with real-world data recorded with the microphones behind the license plate of the research car have been used for emergency siren detection algorithms. In addition to the siren detection, the overall vision system for driver monitoring consists of three subsystems: pose of the head, gaze orientation, and status of eyes. The visual recognition techniques used for each subsystem, along with the datasets employed and preliminary experiments conducted, are briefly explained.

Head Pose Estimation

The head pose estimation has been performed by extracting 3D facial landmarks from RGB or gray-scale images (IR images could also be used for this purpose). MediaPipe Face Mesh [139] has been used, which estimates 468 3D face landmarks from a single camera without requiring deep data. The pose of the head has been estimated from a subset of landmarks, and the main angles in terms of roll, pitch, and yaw have been derived. To reduce the computational load, the area that includes the passenger has been excluded from the computation. An example of landmark detection inside the testing vehicle using MediaPipe Face Mesh is shown in Figure 6.6a.

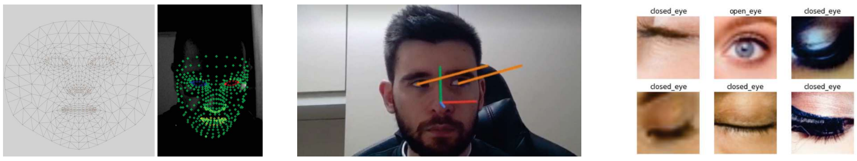
Gaze Estimation

The gaze estimation has been necessary to evaluate whether the driver’s gaze is directed to a region of interest when the approaching emergency vehicle occurs. Both deep and RGB, as well as IR images from Intel RealSense D455, have been used to estimate gaze. Different areas or screens of interest have been defined, namely the front windscreen, the left mirror, and the rear-view mirror. The data from the Intel RealSense D455 camera have been integrated into the Eyeware toolchain[3] to estimate the head pose and track the driver’s gaze. The developed module outputs the head pose (cross-checked with the pose estimated using MediaPipe), the direction of the user’s gaze, and the screen that the driver’s gaze hits. All the data are in a global reference system that is centered on the camera. Figure 6.6b shows an example of gaze estimation using RGB frames. The data are then shared with other applications to correlate the head pose and gaze with the siren detection application.

Eye Status Estimation

Three main sub-images are extracted from the landmarks obtained. Two of these images correspond to the left and right eyes, while the third corresponds to the lips (mouth). The status of the eyes and mouth (opened/closed) is

[3] `https://github.com/eyeware` (accessed on 28 February 2023)



(a) (b) (c)

Figure 6.6: Mesh map for facial landmarks and extracted landmarks (a), gaze estimation from vision system (b), and images of opened/closed eyes (c).

estimated based on these images. A dataset has been created for the scope, composed of 10 000 images of male and female faces belonging to different age ranges and races found on web resources using the SerpApi[4] web scraper. The Facemesh algorithm has been executed to extract facial landmarks, and the eyes have been selected using a bounding box around the landmarks belonging to the left and right eyes. An offset of 20 pixels has also been applied to the bounding box to include more data. An example of images extracted from the created dataset is shown in Figure 6.6c.

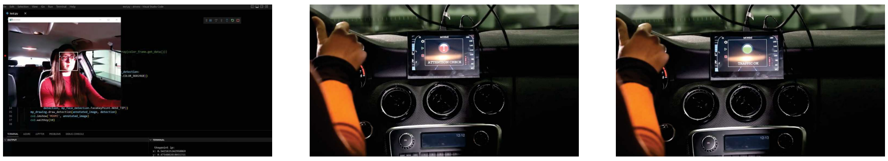
Experimental Setup of the Vision System

Data augmentation has been applied by flipping and performing small rotations (±10°). A MobileNet V2 [140] model has been chosen, where 80% of the sample has been used for training and the remaining 20% for validation. Binary cross-entropy has been set as the loss function for the two classes, which are opened/closed eyes. Transfer learning has been performed by adopting a pre-trained set of weights from Imagenet [141] to adapt the model for the task. Fine-tuning has also been conducted to enhance the overall performance of the model. The resulting training and validation accuracy is above 97.5%. The model could also be further optimized to reduce its weight using TF lite [142] converter. A small reduction in accuracy (97%) has been found from preliminary results, but a 3.5x performance gain has been achieved regarding the time required to predict a single frame.

6.1.4 Open Challenges

The prototype of the emergency vehicle detection device based on driver awareness is an innovative system and presents no significant problems for its engineering, as acoustic and miniature imaging sensors are already integrated into commercial cars. The main challenge is perfecting the deep algorithms and expanding the acoustic and visual detection case histories. On the audio side,

[4] `https://serpapi.com` (accessed on 28 February 2023)



(a) Gaze monitoring system.

(b) Interface in warning state.

(c) Interface in ordinary traffic state.

Figure 6.7: Images from the demonstration video of the EVD prototype.

reducing latencies and minimizing the size of audio frame sizes used by the siren detection algorithm is necessary to make the system suitable for real-world scenarios. In addition, the system needs to generalize to all siren sounds used by emergency vehicles in different countries. From a computer vision perspective, there are several challenges, including the generalization capability of the developed models to a wide range of faces and detection based only on head pose for drivers wearing sunglasses. Other challenges include detection robustness under different lighting conditions and data security from a privacy perspective. In addition, human factors must be considered when deploying the EVD system. The driver must be aware of the system's presence, and the warning information provided must be clear and not distracting.

In conclusion, the results from the feasibility investigation of the prototype system have shown promising outcomes that will be assessed in a real-world testing campaign. Meanwhile, the demo of the prototype has been presented in a video recorded inside the semianechoic chamber of the Department of Information Engineering at Università Politecnica delle Marche, Italy. Figure 6.7 shows images of the driver's face detection and gaze monitoring system and the display with and without warnings presented during the demonstration. The full video can be watched at the link `https://youtu.be/WjyewoCm7NU`.

Chapter 7

An Audio-Visual Dataset for Driving Scene Understanding

The field of perception systems for self-driving cars is expanding rapidly with numerous applications in industry and academia [143]. Of particular interest is the development of technologies for the automatic understanding of driving scenarios, which is becoming more feasible due to the increasing availability of large quantities of driving data. These technologies have the potential to be used in a range of applications, from advanced driver-assistance systems to fully autonomous driving. Audio-visual information is paramount to fully understanding real-world scenarios, as visual and auditory modalities provide complementary information. Visual data help identify features of the road infrastructure traveled, landscape, and natural or artificial lighting; audio provides information about traffic noise or nature sounds, adverse weather conditions, and reverberant environments (e.g., tunnels, underground parking lots). Effective automatic solutions must be able to handle a broad range of sound scenes and sensing conditions, including those that are noisy, sparse, and with moving sources. The ideal system should be able to adapt to changing conditions and maintain robust performance in all situations.

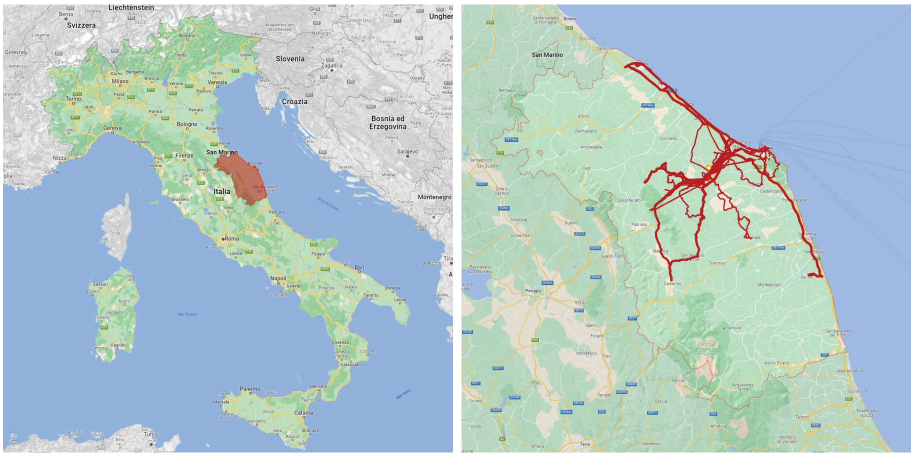
In the past decade, research in computer vision produced a wide variety of real-world driving datasets, which have provided researchers with opportunities to develop new algorithms. Most of the best-known datasets concentrate on specific perception tasks due to the high expenses incurred during data collection and annotation. The KITTI dataset [144] is a pioneering work including recordings in urban, highway, and rural scenarios performed with stereo cameras and a LiDAR sensor during the daytime for 1.5 hours. More challenging scenarios have been collected in nuScenes [145], consisting of 5.5 hours of driving data recorded by multiple sensors in urban, residential, natural, and industrial sites, and in the Waymo [146] dataset that offers 6.4 driving hours in suburban and downtown areas, both recorded in daytime and nighttime with varying weather conditions. The relatively small size of these collections posed new challenges to the construction of vision-based large-scale datasets. A significant example is the ONCE [147] dataset that comprises 144 hours of video data in multiple contexts, time of the day, and weather conditions, recorded with LiDAR sensors and cameras.

While there is a great deal of related work in the computer vision area, datasets on machine listening focus mainly on sound event detection [148–150] and acoustic scene classification [151–153] in urban environments. Also, few works exist on audio-visual classification, e.g., involving dynamic environments [154], urban scenes [155], and urban traffic data [156]. Visual-acoustic multimodal data have been collected with an instrumented car in [157]to improve driving pleasantness by monitoring the state of the vehicle interior and in [158] for obstacle detection and tracking under vehicle vertical dynamics excitation caused by road anomalies.

To the best of our knowledge, multisensor and multimodal recordings conducted in real-world scenarios with significant duration and consistent audio and video quality aimed at a complete comprehension of the car's surroundings are not available on public databases. For this reason, an audio-visual dataset for driving scene understanding, called "A3CarScene," has been created and is presented in the following.

7.1 A3CarScene Dataset

A3CarScene is an audio-visual dataset comprising more than 31 hours of audio and video data recorded while driving a research car on public roads. The sensor equipment consists of eight microphones installed inside and outside the passenger compartment and two dashcams mounted on the front and rear windows of the vehicle. Acquisitions were made in the Marche Region, located in the center of Italy and characterized by variegated landscapes, from the coast in the east to the hilly areas in the center and the Apennine mountains in the west. Regarding its urbanization, the Marche Region presents two main urban centers (Pesaro and Ancona) and many towns with their respective suburban belts, exurban areas with industrial sites and infrastructure connections, and rural lands with scattered villages. The recording campaign was carried out in October and November 2022 for 14 days, covering different routes for a total of 1500 km. The itineraries were planned to encompass diverse areas, focusing on the central part of the region due to logistical reasons. Figure 7.1 shows the location of the Marche Region and the routes traveled during the recording campaign.



(a) (b)

Figure 7.1: Location of the Marche Region in Italy (a) and routes traveled with the equipped car (b).

7.1.1 Experimental Design, Materials and Methods

Acquisition Stage

A Mercedes A-Class research car model equipped with audio and video sensors was used for the recording campaign. This vehicle and the related audio setup have already been employed in other works and have been described in Section 3.1.2. Recordings were made with either only the driver or, at most, one passenger on board. The car was driven within the speed limits imposed by the road infrastructure and with the windows either open or closed as desired. No music sources were activated during driving, and no dialogue was present.

The audio system, including eight condenser microphones connected to an eight-channel audio interface, was integrated with video devices. The video equipment consisted of two cameras Mi DashCam 1S attached with the dedicated mount to the front and rear windows of the car. The front camera was secured to the right of the rearview mirror so as not to interfere with the driver's view, while the rear camera was placed in the high-center position of the rear window. The cameras were powered via a USB cable connected to the USB ports included in the car. Video data were captured with a 1920×1080 pixels resolution and variable fps for up to 30 fps and saved 2-minute segments in mp4 format. The two cameras are equipped with an internal clock that enables the synchronization of video data. Audio data recording is optional and has been enabled to facilitate synchronization with audio devices.

Device	Main Specifications
Behringer ECM8000 microphone	Type: elect. condenser. Polar Pattern: omnidirectional. Impedance: 200 Ohms. Sensitivity: 70 dB. Frequency Response: 20-20 000 Hz. Connector: gold-plated XLR. Phantom Power: +15 to +48 V. Weight: 136 g.
Roland Octa-Capture audio interface	Number of audio channels: 8. Nominal Input Level: input jack 1–6 (XLR type) -56 to -6 dBu, input jack 7–8 (XLR type) -50 to +0 dBu. Nominal Output Level: +0 dBu (balanced). Headroom: 16 dB. Input Impedance: input jack 1–6 (XLR type) 5 k ohms (balanced), input jack 7–8 (XLR type) 10 k ohms (balanced). Output Impedance: 1.8 k ohms (balanced). Frequency Response 44.1 kHz: 20 Hz to 20 kHz (+0/-2 dB). Power Supply: DC 9 V (AC adaptor). Current Draw: 1.45 A. Dimensions: 283.8 (W) x 157.9 (D) x 50.4 (H) mm. Weight: 1.32 kg
Mi DashCam 1S	Dimensions: 87.5 (W) x 18 (D) x 53 (H) mm. Input: 5 V, 1.5 A. Image Sensor: Sony IMX307. Resolution: 1080 p. Camera: FOV 140°, F1.8, 6-glass lens. Frame Rate: variable. Working Frequency: 2412–2472 MHz. Operating Temperature: -10 °C–60 °C.

Table 7.1: Main technical specifications of audio and video recording devices.

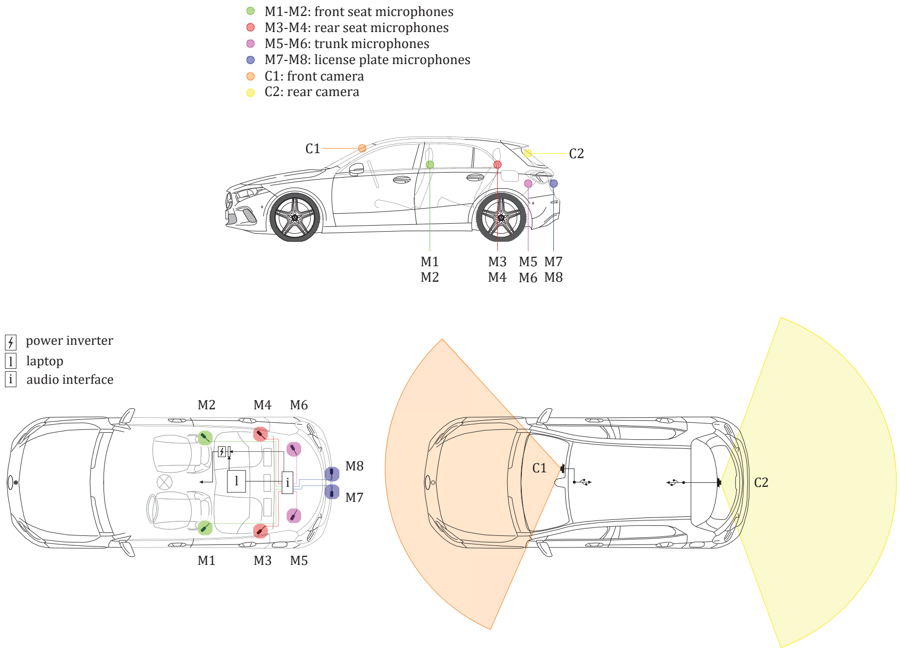


Figure 7.2: Setup of audio and video recording system.

Data were collected by driving planned routes and acquiring real-time audio and video data. Before departure, a check of the operation of all devices was carried out by initiating test recordings. The audio recordings were activated by turning on the audio interface and starting 8-channel recording using the open-source software Audacity[1] installed on the onboard laptop. Video recordings started automatically with the connection to the power supply.

Table 7.1 lists the main technical specifications of audio and video recording devices, and Figure 7.2 schematizes the configuration of the audio and video devices.

Processing Stage

Processing operations were carried out to synchronize audio and video data and to apply data protection laws. Specifically, the following procedures were performed for each recording associated with a specific route.

- Video data from each camera, recorded in 2-minute segments, were merged into a single video file and exported in mp4 format at 25 fps. The synchronization of front and rear videos was verified by comparing the time-frequency representations of the audio acquired by each camera. Open-source software kdenlive[2], based on the ffmpeg [159] library, and Audacity, were used to perform the processing operations on video and audio data, respectively.

- Using the audio tracks recorded by the cameras, video data were aligned with the eight-channel recordings from the microphones, which were then exported to separate tracks in wav format, keeping the 44.1 kHz sampling rate and 32-bit encoding unchanged.

- To comply with General Data Protection Regulation guidelines, license plates and faces were censored with the open-source python tool DashcamCleaner[3]. It is based on the YOLOv5 [160] algorithm for automatic license plate and face recognition using pre-trained models with different parameters that adjust training image resolutions, network depths, and dataloaders. Video files were blurred with 720p_medium_mosaic option, kernel radius of the gaussian filter of 30, and the quality of the resulting video equal to 5.

- Lastly, video and audio data for each itinerary were played simultaneously in the kdenlive software for the manual labeling phase. For each class, markers were applied corresponding to the start and end of each homogeneous context in the road, urban, meteorological, and temporal domains and annotated in a csv file.

[1] `https://www.audacityteam.org/` (accessed on 28 February 2023)

[2] `https://kdenlive.org/en/`

[3] `https://github.com/tfaehse/DashcamCleaner` (accessed on 28 February 2023)

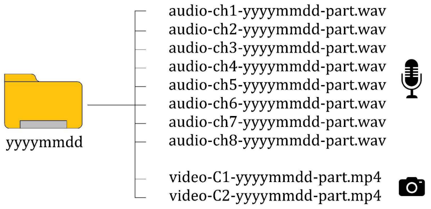

yyyymmdd
audio-ch1-yyyymmdd-part.wav
audio-ch2-yyyymmdd-part.wav
audio-ch3-yyyymmdd-part.wav
audio-ch4-yyyymmdd-part.wav
audio-ch5-yyyymmdd-part.wav
audio-ch6-yyyymmdd-part.wav
audio-ch7-yyyymmdd-part.wav
audio-ch8-yyyymmdd-part.wav
video-C1-yyyymmdd-part.mp4
video-C2-yyyymmdd-part.mp4

Figure 7.3: Organization of audio and video files inside a folder.

7.1.2 Data Description

The dataset consists of 400 files (320 audio and 80 video recordings). The files are organized into 14 folders, named with the acquisition date *yyyymmdd*, and each folder contains all audio and video files recorded on the same day. The duration of the files is variable, depending on the length of the itinerary or the cuts applied to individual recordings. The synchronized audio and video files inside each *yyyymmdd* folder are named with the criterion *file_type-device-yyyymmdd-part*. Audio recordings were stored in eight-channel tracks and exported separately, so audio-type files report the channel number (ch1–ch8) of the corresponding microphone as the device. Videos were shot with two cameras, where C1 is the frontal and C2 is the rear video device. Figure 7.3 shows the generic contents of a folder.

All key information for the scene understanding process of automated vehicles has been accurately annotated. For each route, scene annotations with beginning and end timestamps report the type of road traveled (*motorway*, *trunk*, *primary*, *secondary*, *tertiary*, *residential*, and *service* roads), the degree of urbanization of the area (*city*, *town*, *suburb*, *village*, *exurban* and *rural* areas), the weather conditions (*clear*, *cloudy*, *overcast*, and *rainy*), the level of lighting (*daytime*, *evening*, *night*, and *tunnel*), the type (*asphalt* or *cobblestones*) and moisture status (*dry* or *wet*) of the road pavement, and the state of the windows (*open* or *closed*). Annotations are consistent for audio and video files and are reported in text files in csv format. The metadata folder contains the annotations of each recording date (*metadata-yyyymmdd.csv*) plus an overall one (*metadata.csv*), for a total of 15 csv files.

Table 7.2 shows the structure of metadata files reporting the name of the file (*filename*), timestamps (*start_time* and *end_time*), area identification number (*id*), and labeling categories (*road_type*, *deg_urb*, *weather*, *light*, *pav_type*, *pav_wetness*, and *window*). Each labeling category lists the corresponding attributes (or classes) that have been assigned.

The individual columns of the annotation files are explained in detail as follows.

- *filename* is the string common to audio and video filenames, expressed

filename	start_time	end_time	id	road_tipe	deg_urb	...	windows
yyyymmdd-part	hh:mm:ss	hh:mm:ss	xx	label_1	label_2	...	label_7

Table 7.2: Structure of annotation files (*.csv).

by the date and the part of the recording belonging to the same day (*yyyymmdd-part*).

- *start_time* and *end_time* are the timestamps in which the scene has uniform labeling, expressed in *hh:mm:ss* format.

- *id* represents the number that identifies the area covered, ranging from 001 and 407. The *id* can indicate a single road section or a group of neighboring roads belonging to the same type of infrastructure and degree of urbanization. The purpose of the *id* assignment is related to the split of the dataset into training and test sets so that routes different from those used in the training phase can be chosen for inference.

- *road_type* represents the road classification typology according to OpenStreetMap (OSM) [161], the free geographic database updated and maintained by a community of volunteers through open collaboration. The choice of this classification is related to the worldwide use of OSM and international equivalence between road infrastructure types. In the following, the description of the infrastructures traveled and the equivalence with the Italian regulations according to the Legislative Decree No. 285 of April 30, 1992 “Codice della Strada”[4] are given.

  1. Motorway: limited access highway with tolls and interchanges (in Italy, A-category road).
  2. Trunk: ring road or expressway, also a road of minor importance having interchanges instead of grade-separated intersections (in Italy, B-category road).
  3. Primary: national, regional, or provincial road of major importance, e.g., that connecting provincial capitals and thus of national significance (in Italy, B-category road).
  4. Secondary: another regional or provincial road of minor importance (in Italy, C-category road).
  5. Tertiary: main urban road (in Italy, D-category road).
  6. Residential: road in an urban residential area (in Italy, E-category road).

[4] (https://www.bosettiegatti.eu/info/norme/statali/1992_0285.htm) (accessed on 28 February 2023)

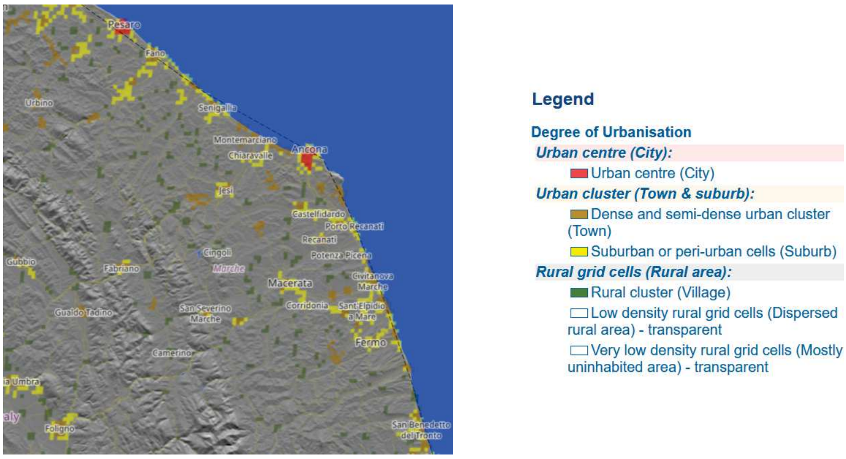


Figure 7.4: Degree of Urbanisation map of the Marche Region in Italy.

7. Service: a service way to access, for example, a non-residential area, a parking lot, or a private area (in Italy, F-category road).

- *deg_urb* represents the classification that indicates the character of an area. It is inspired by the "Degree of Urbanisation" [162], a methodology for the delineation of cities and urban and rural areas for international and regional statistical comparison purposes endorsed by the United Nations Statistical Commission. The Degree of Urbanisation classifies the entire territory of a country along the urban-rural continuum, combining population size and density thresholds to capture the full settlement hierarchy. Global Human Settlement data with global coverage, as illustrated in Figure 7.4, can be viewed interactively on the web[5]. According to the Degree of Urbanisation legend, territories are represented by the following attributes.

  1. City: urban center.
  2. Town: dense and semi-dense urban cluster.
  3. Suburb: suburban and peri-urban cells.
  4. Village: rural cluster.
  5. Exurban area: low-density grid cells (industrial sites, rural or mixed-use areas).
  6. Rural area: very low-density rural grid cells.

- *weather* describes the weather conditions detected during the route. The options are:

[5] https://ghsl.jrc.ec.europa.eu/visualisation.php# (accessed on 28 February 2023)

1. Clear: sunny sky with no or insignificant cloud cover.
2. Cloudy: sky with clouds but not completely covered.
3. Overcast: sky completely covered with clouds.
4. Rainy: overcast sky with light to moderate to significant rainfall.

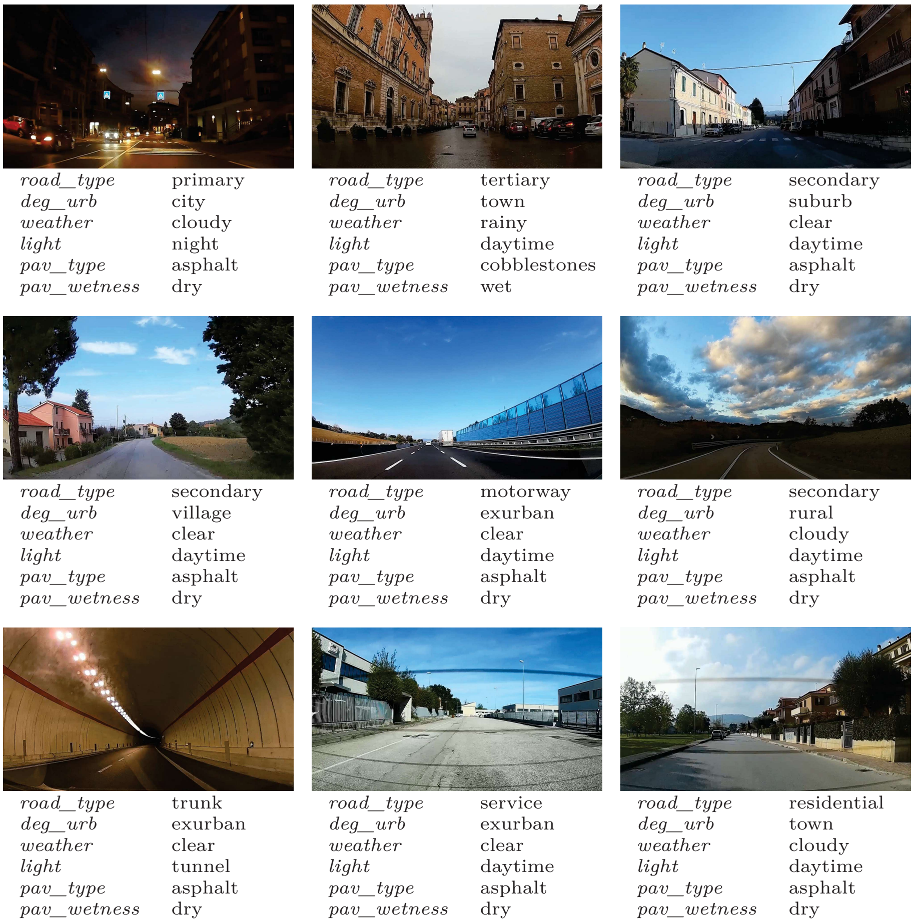
- *light* relates to the time of day when recordings were made and passage in closed environments with artificial lighting (tunnels, covered parking lots). The following lighting conditions occur in the recordings: daytime, evening, night, and tunnel.
- *pav_type* represents the type of road pavement encountered in the routes, i.e., asphalt or cobblestones.
- *pav_wetness* indicates the moisture of the road pavement (dry or wet).
- *window* indicates the state of the windows during the recordings (open or closed). This feature is not descriptive of the car's surroundings but was included to assess the impact of the external noise on the performance of the audio sensors inside the passenger compartment.

The dataset includes 31 hours, 20 minutes and 8 seconds of recordings for each audio and video sensor. The individual classes in each labeling category are imbalanced proportionally to the territory characteristics and weather conditions encountered during the acquisition campaign. Figure 7.5 shows examples of video frames of the routes with some associated annotations.

7.2 Value of the Data and Future Insights

This study lays the foundation for a strand of research aimed at understanding the environment surrounding automobiles. In the emerging paradigm of increasingly less model-centric and more data-centric artificial intelligence [163], data quality, quantity, and engineering are essential for building reliable and efficient AI-based systems.

This large-scale audio-visual dataset provides a wide range of driving scenarios, showing several road infrastructures, including pavement types and wetness, diverse urbanization contexts, and varying weather and lighting conditions. Understanding the type of road infrastructure being traveled allows for the automatic activation of advanced driver-assistance systems suitable to the context. For example, driving on a motorway requires speed control and appropriate distancing from vehicles in the same lane. At the same time, a residential street may feature pedestrian or cyclist crossings, so object detection becomes paramount. The degree of urbanization makes it possible to contextualize the



road_type	primary
deg_urb	city
weather	cloudy
light	night
pav_type	asphalt
pav_wetness	dry

road_type	tertiary
deg_urb	town
weather	rainy
light	daytime
pav_type	cobblestones
pav_wetness	wet

road_type	secondary
deg_urb	suburb
weather	clear
light	daytime
pav_type	asphalt
pav_wetness	dry

road_type	secondary
deg_urb	village
weather	clear
light	daytime
pav_type	asphalt
pav_wetness	dry

road_type	motorway
deg_urb	exurban
weather	clear
light	daytime
pav_type	asphalt
pav_wetness	dry

road_type	secondary
deg_urb	rural
weather	cloudy
light	daytime
pav_type	asphalt
pav_wetness	dry

road_type	trunk
deg_urb	exurban
weather	clear
light	tunnel
pav_type	asphalt
pav_wetness	dry

road_type	service
deg_urb	exurban
weather	clear
light	daytime
pav_type	asphalt
pav_wetness	dry

road_type	residential
deg_urb	town
weather	cloudy
light	daytime
pav_type	asphalt
pav_wetness	dry

Figure 7.5: Video frames of the A3CarScene dataset with some associated annotations.

area and provide relevant suggestions to the driver, such as the nearest services in rural and exurban areas or traffic advisories in urban centers. Detection of weather and lighting conditions has immediate feedback in the activation of windshield wipers and headlights, while tire alignment can be automatically changed based on pavement characteristics and moisture.

Thus, the information gathered from this study is a valuable asset for those involved in developing and testing advanced driver-assistance systems, as well as for automotive research in general. The data obtained from different sensor configurations, including acoustic and visual signals, facilitate the technical evaluation and design of intelligent systems. Researchers and developers can leverage the real-world dataset obtained from this research to train and evaluate deep learning algorithms for driving scene recognition using audio, video, or a combination of both. In addition, the dataset may be useful for manufacturers who wish to compare the effectiveness of their systems with those of competitors.

The studies to be undertaken will focus on the potential of data, especially audio data, in understanding the driving scenario. The experiments will be conducted in a single mode, and the performance resulting from audio data in single-modality will be compared with that obtained from video data and in multimodality. Suitable architectures for this study are the L3-Net ("look, listen and learn") [164] model, designed to learn the correspondence between audio and video inputs by predicting whether they are correlated, and the OpenL3 [165] network, an open-source implementation pre-trained on AudioSet [79]. An optimal solution has been proposed in [155] and is based on OpenL3, in which the auditory scene recognition and visual scene recognition are performed separately, and the subsequent results are concatenated to obtain multimodal outcomes. In this specific case, multi-annotations should be considered for the design of the neural architecture, which involves the use of separate classification branches or the study of advanced multi-label learning strategies [166]. Investigating the correlation between the various features and the contribution each makes to the target task is another aspect of research that needs to be pursued.

Finally, the dataset is also suitable for other applications not covered at this stage. By extending the labeling to objects and acoustic signals of interest, it is possible to perform tasks that include, but are not limited to, the identification of road damages, intersections, and warning acoustic signals, as well as object recognition and detection of obstacles out of sight.

Chapter 8

Other Contributions

This section presents two research studies involving acoustic themes from different perspectives. The first study is in the field of building acoustics and focuses on a predictive tool for the speech transmission index in school environments. The second study describes a series of experiments aimed at understanding the characteristics and production modalities of the sound in a vintage musical instrument: the Rhodes electric piano.

8.1 A Predictive Tool for the Speech Transmission Index Assessment

Speech communication is a complex phenomenon that involves different modalities of speaker-listener interactions in conversational environments and encompasses the aspects of speech quality, vocal effort, perceptual delays, and speech intelligibility. In particular, speech intelligibility, defined as the "rating of the proportion of speech that is understood," [167] assumes a key role in environments where the focus is on speech understanding, such as lecture rooms. For this reason, acoustical standards and guidelines currently in use are designed to ensure good speech intelligibility by defining reference values of acoustical descriptors to maximize the occupant comfort and functional performance of educational environments.

The present research focuses on one of the acoustic descriptors for school buildings, the speech transmission index (STI), "a physical metric that is well correlated with the intelligibility of speech degraded by additive noise and reverberation" [168]. A prediction tool was implemented either stand-alone or in combination with an artificial neural network to calculate the STI values of school classrooms, and the results were compared with the findings of measurements made in classrooms of different grades, building types, and sizes.

8.1.1 Materials and Methods

Speech Transmission Index and Reference Standards

As mentioned, STI is an objective descriptor to predict the intelligibility of speech transmitted from talker to listener by a transmission channel, whose measurement procedures, prediction methods and reference values are regulated by standards. Specifically, IEC 60268-16 [169] (BS EN 60268-16 [170]) defines the methodologies for the objective rating of speech intelligibility by the speech transmission index, including both measurement procedures and computational methods. The direct STI measurement technique applies a specific test signal to the transmission channel, and by analyzing the received signal, the speech transmission quality of the channel is derived and expressed in a value between 0 and 1. On the other hand, STI can be calculated with a method called "statistical" or "indirect" from the measurement of reverberation time (RT) in specific positions of the analyzed room. The UNI 11532-1 [171] standard describes the general aspects common to all application areas that best represent the acoustic quality of an environment. This standard presents reference values in relation to the use destination of the environment, prediction methods, and evaluation techniques that constitute a common operational methodology, referring to IEC 60268-16. It also incorporates the concept of "intelligibility rating for speech communications" expressed in UNI EN ISO 9921, consisting of a discrete qualification that depends on the range in which the STI falls according to a five-point scale of speech comprehension quality (*bad*, *poor*, *fair*, *good*, *excellent*), as shown in Table 8.1. Finally, UNI 11532-2 [172] defines acoustic quality descriptors and specific reference values for the educational sector.

STI values	IR
0.00<STI≤0.30	Bad
0.30<STI≤0.45	Poor
0.45<STI≤0.60	Fair
0.60<STI≤0.75	Good
0.75<STI≤1.00	Excellent

Table 8.1: Correlation between speech transmission index (STI) and intelligibility rating (IR) according to UNI 11532-1.

Characteristics of Lecture Rooms and Measurement Equipment

In this research, direct measurements of the RT have been performed in thirty-five classrooms of several grades belonging to buildings with different structural characteristics and construction areas located in the Marche region in Italy by applying the assessment procedure described in UNI 11532-2. For each classroom, the STI has been calculated with the indirect method described in the IEC 60268-16, and the intelligibility rating expressed in UNI EN ISO 9921 has been assigned. The measurement campaign has been conducted in a heterogeneous sample of school buildings in terms of structure type, year of construction, educational stages, activities performed, built area, materials, and construction techniques. Design documentation has been collected for each school building, and the geometric survey and visual analysis of classroom finish materials have been conducted.

ID	Type		Name of room	length	width	height	surface	volume
				[m]	[m]	[m]	[m^2]	[m^3]
1	SCH 1	University	140/1	18.2	9.0	3.4	163.8	556.9
2	SCH 1	University	140/2	11.9	8.8	3.9	105.2	410.3
3	SCH 1	University	140/3	15.3	8.9	3.9	136.5	537.6
4	SCH 1	University	155/D1	12.3	8.9	3.4	109.4	371.9
5	SCH 1	University	155/D2	12.3	8.9	3.4	109.4	371.9
6	SCH 1	University	155/D3	12.3	8.9	3.4	109.4	371.9
7	SCH 1	University	155/D4	12.3	8.9	3.4	109.4	371.9
8	SCH 1	University	160/1	10.2	8.9	3.4	90.9	309.1
9	SCH 1	University	160/2	10.4	8.9	3.4	93.2	317.0
10	SCH 1	University	AT1	13.4	9.3	3.0	125.2	375.7
11	SCH 1	University	AT2	13.8	9.3	3.0	128.6	385.9
12	SCH 1	University	AT3	17.9	6.3	3.0	113.1	339.4
13	SCH 1	University	EN1	11.3	8.9	3.4	100.7	342.3
14	SCH 1	University	EN3	11.2	6.4	3.4	71.4	242.7
15	SCH 1	University	S1	7.5	7.4	3.0	55.5	166.5
16	SCH 1	University	S2	10.4	10.6	3.5	110.2	385.8
17	SCH 1	University	S3	14.3	16.2	3.0	231.7	695.0
18	SCH 2	Primary	1B	7.3	6.8	3.3	49.6	163.8
19	SCH 2	Primary	2A	7.8	7.4	3.3	57.7	191.2
20	SCH 2	Primary	3A	6.8	6.1	3.3	41.5	136.9
21	SCH 3	Primary	2C	7.4	6.8	3.0	50.3	151.0
22	SCH 3	Primary	3C	7.8	7.4	3.0	57.3	172.0
23	SCH 4	Primary	4B	7.9	7.5	3.0	59.3	177.8
24	SCH 4	Primary	4A	8.1	7.8	3.0	63.4	190.3
25	SCH 5	Secondary	1	7.9	7.3	3.1	57.3	177.6
26	SCH 5	Secondary	2	7.9	7.5	3.2	59.2	189.3
27	SCH 5	Secondary	3	8.0	7.6	3.1	60.1	186.3
28	SCH 6	Secondary	4	6.8	6.7	3.0	45.2	135.5
29	SCH 6	Secondary	5	7.3	6.4	3.0	46.7	140.0
30	SCH 7	Secondary	6	8.4	6.7	4.4	56.3	247.6
31	SCH 7	Secondary	7	10.7	7.5	6.5	80.3	521.6
32	SCH 7	Secondary	8	7.1	7.3	3.0	51.7	155.1
33	SCH 8	Secondary	9	7.2	6.2	3.0	44.6	133.9
34	SCH 8	Secondary	10	7.8	6.7	3.0	52.3	156.8
35	SCH 8	Secondary	11	6.0	9.0	3.0	53.8	161.3

Table 8.2: List of classrooms indicating school type (school number and grade), name of the room, and geometric dimensions.

The list of geometric dimensions of each classroom is provided in Table 8.2. The acoustic characterization of the classrooms has been carried out in compliance with ISO 3382-2 [173] for the RT$_{30}$ measurement procedure at the positions defined in the UNI 11532-2 standard. Four positions have been selected (P1–P4), three along the longitudinal axis of the classroom and one representative of the most unfavorable condition in terms of distance from the speaker and proximity to the noise produced by the indoor plant.

The system setup comprised a laptop and an Edirol FA-101 external firewire soundcard connected to an Echo Speech Source (Type 4720). This small active loudspeaker box provides calibrated acoustic signals through the Dirac room acoustics software, and placed at a human speaker position (1.50 m from the floor). The signal has been acquired with a B&K 2250 sound level meter that outputs the impulse response and reverberation time. Measured RT_{30} and computed STI descriptors for each classroom are presented in Table 8.3.

ID	RT_{30} [s]	STI					IR
		P1	P2	P3	P4	avg	
1	0.77	0.57	0.52	0.46	0.43	0.50 ± 0.06	fair
2	0.70	0.52	0.40	0.37	0.34	0.41 ± 0.08	poor
3	0.67	0.60	0.58	0.35	0.33	0.47 ± 0.14	fair
4	0.69	0.63	0.57	0.54	0.52	0.57 ± 0.05	fair
5	0.65	0.62	0.57	0.54	0.52	0.56 ± 0.04	fair
6	0.72	0.55	0.51	0.53	0.49	0.52 ± 0.03	fair
7	0.68	0.60	0.55	0.54	0.52	0.55 ± 0.03	fair
8	0.78	0.59	0.42	0.51	0.39	0.48 ± 0.09	fair
9	0.80	0.58	0.50	0.45	0.40	0.48 ± 0.08	fair
10	0.68	0.73	0.61	0.59	0.45	0.60 ± 0.11	fair
11	0.56	0.70	0.60	0.60	0.56	0.62 ± 0.06	good
12	0.61	0.82	0.62	0.54	0.50	0.62 ± 0.14	good
13	0.75	0.61	0.56	0.55	0.41	0.53 ± 0.09	fair
14	0.72	0.56	0.52	0.49	0.43	0.50 ± 0.05	fair
15	1.81	0.29	0.25	0.28	0.29	0.28 ± 0.02	bad
16	1.81	0.28	0.26	0.28	0.27	0.27 ± 0.01	bad
17	1.80	0.25	0.28	0.29	0.30	0.28 ± 0.02	bad
18	0.78	0.66	0.64	0.62	0.57	0.62 ± 0.04	good
19	1.20	0.54	0.57	0.54	0.53	0.55 ± 0.02	fair
20	1.20	0.53	0.56	0.55	0.54	0.55 ± 0.01	fair
21	1.38	0.48	0.50	0.52	0.48	0.50 ± 0.02	fair
22	0.99	0.54	0.61	0.54	0.47	0.54 ± 0.06	fair
23	0.82	0.57	0.62	0.63	0.59	0.60 ± 0.03	fair
24	1.50	0.44	0.47	0.41	0.39	0.43 ± 0.04	poor
25	0.83	0.62	0.60	0.59	0.54	0.59 ± 0.03	fair
26	1.14	0.58	0.55	0.53	0.50	0.54 ± 0.03	fair
27	0.85	0.57	0.58	0.57	0.56	0.57 ± 0.01	fair
28	1.36	0.54	0.49	0.49	0.43	0.49 ± 0.05	fair
29	0.96	0.58	0.56	0.49	0.44	0.52 ± 0.06	fair
30	0.63	0.77	0.70	0.70	0.70	0.72 ± 0.04	good
31	0.81	0.64	0.49	0.47	0.47	0.52 ± 0.08	fair
32	0.66	0.65	0.53	0.51	0.51	0.55 ± 0.07	fair
33	0.67	0.54	0.52	0.49	0.55	0.53 ± 0.03	fair
34	2.15	0.38	0.31	0.33	0.34	0.34 ± 0.03	poor
35	2.00	0.44	0.37	0.38	0.38	0.39 ± 0.03	poor

Table 8.3: Measured RT_{30} (average), STI (P1–P4 and average), and IR (referred to average STI) for classrooms under acoustic speech intelligibility assessment.

STI Prediction with Simulation Tool

According to the UNI 11532-1 standard, predictive methods are allowed to compute the room impulse response in indoor environments in order to optimize the acoustic descriptor under consideration. For this purpose, a fully predictive tool of speech transmission index in room acoustic environments,

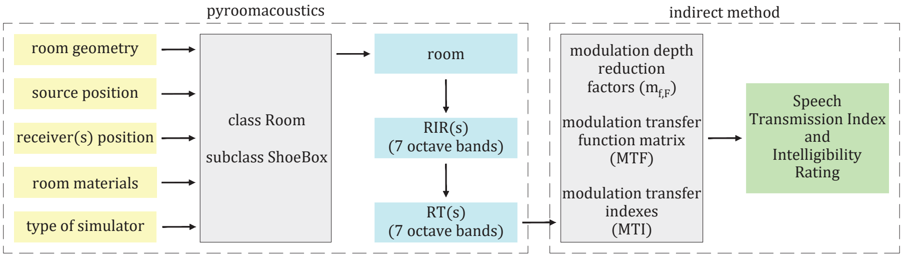


Figure 8.1: Block diagram of the *pyeSTImate* tool.

called *pyeSTImate*[1], has been developed. This application aims to compute the STI without recourse to direct measurements of reverberation times. RIRs can be simulated with different techniques in the literature, given the geometry of the room, constituent materials, furniture and kind of occupants, and source and receivers' positions. After calculating reverberation times from the RIRs, STIs are determined using the indirect method. The extensive case history of classroom measurements has been used to fine-tune the simulator settings: comparing the STI values derived from the measured RTs and the results of different simulation methods allowed us to assess which settings most accurately reproduce the actual environment. The result is a tool for acoustics engineers suitable for the analysis of existing rooms, as well as for the renovation and design of new spaces.

The core of the pyeSTImate tool is *pyroomacoustics* [174], a software package aimed at the rapid development and testing of audio array processing algorithms, properly adapted and extended to predict the STI from dimensional room data, materials, and source/receiver positions. As illustrated in Figure 8.1, pyeSTImate is composed of two main blocks. The first, based on pyroomacoustics, returns the simulated room impulse responses according to three simulation methods of the user's choice (image source method or ISM [175], ray tracing [176, 177], and hybrid ISM/ray tracing modeling). After the simulation of room impulse responses follows the calculation of reverberation times in octave bands, which is given as input to the second part of the algorithm designed for STI computation using the indirect method and, consequently, the intelligibility rating of the room.

STI Prediction with Artificial Neural Network

Whereas pyeSTImate requires detailed input geometric data, especially regarding the surfaces to be associated with each material, in some cases, only some of this information is available, e.g., floor plans and photos of the room. To ad-

[1] `https://github.com/michelacantarini/pyeSTImate` (accessed on 28 February 2023)

dress this issue, the algorithm has been adapted to generate a synthetic dataset of classrooms characterized by randomly chosen sizes and materials, and in each of them, the average speech transmission index over four listening positions has been calculated. The dataset, composed of reduced input data compared to the full pyeSTImate tool, has been employed to train an artificial neural network (ANN) capable of predicting STI with good approximation by providing only classroom size and material characteristics. This further application estimates the speech transmission index and speech comprehension quality with very limited details about the target room, demonstrating its usefulness in preliminary acoustic analyses.

A dataset of 1000 classrooms has been generated, keeping as features only the geometry (length, depth, and height) and the absorption and scattering coefficients in octave bands of the interior finishes. An ANN has been devised to solve a regression problem and predict a continuous value between 0 and 1. To find the model that has the greatest generalization capability and provides the best results with data unseen during the training, the performance of the algorithm has been investigated by varying the size of the training set, the number of hidden layers and neurons in each of them through a grid search [112]. The experiments have been carried out with training sets of 100 and 1000 examples and neural architectures composed of one, two and three hidden layers, each with a number of nodes between 8 and 1024. For training, the entire dataset has been randomly split into training and validation sets, with 80% and 20% of the total number of samples, respectively. For testing, the 35 classrooms already investigated with instrumental measurements have been considered. Training has been performed with the ReLU activation function in the hidden layers, a learning rate equal to 0.001, Adam [76] optimizer, and 5000 epochs with the early-stopping.

Performance Metrics

The following metrics have been monitored to evaluate the accuracy of the tool in comparison with the results of in situ measurements:

- the absolute error (AE), absolute percentage error (APE) and intelligibility rating error (IRE) between average STI values from measurements and simulations in each classroom;
- the mean absolute error (MAE), mean absolute percentage error (MAPE) and mean intelligibility rating error (MIRE) over classrooms belonging to schools of the same grade and the entire test set of classrooms.

If the (mean) absolute error and (mean) absolute percentage error represent a regression loss between STI values, the metric (mean) intelligibility rating error quantifies the error associated with the discrete classification of the speech comprehension quality. The MIRE values fall in the range [0,1] and can also be expressed as a percentage. A label from 1 to 5, where 1 indicates bad quality and 5 corresponds to excellent quality has been assigned to the five attributes of the speech comprehension quality scale (*bad*, *poor*, *fair*, *good*, *excellent*). The ratio of the absolute value of the difference between the measured and simulated IR labels and the range of the actual values averaged over the number of the analyzed classrooms provides the mean intelligibility rating error, defined in Equation (8.1) as:

$$MIRE(l, \hat{l}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} \frac{|l_i - \hat{l}_i|}{max(l) - min(l)} \tag{8.1}$$

where

- l_i is the true intelligibility label of the ith sample,
- $\hat{l}_i$ is the predicted intelligibility label of the ith sample,
- $max(l) - min(l)$ represents the range of the actual values,
- $n_{samples}$ is the number of samples in the test set.

In addition to the previously mentioned metrics, the accuracy of the results has also been monitored in terms of just noticeable difference, the subjective limen representing the discernible difference of a room acoustic parameter. The study associated with the just noticeable difference in STI, i.e., the variation in STI values for which 50% of subjects can perceive the difference, determined a STI JND equal to 0.03 in simulated sound fields, but a STI JND of 0.1 is considered more realistic in everyday listening situations [178]. For this reason, the number of JND units (STI JNDs) between STI values from measurements and simulations have been calculated with both thresholds.

8.1.2 Results

Assessment of Speech Intelligibility through *pyeSTImate*

Table 8.4 summarizes the mean absolute error (MAE), mean absolute percentage error (MAPE), mean intelligibility rating error (MIRE), and mean just noticeable difference units with thresholds equal to 0.03 (mean STI JNDs(0.03)) and 0.1 (mean STI JNDs(0.1)) computed for classrooms of the same grade and overall the classrooms.

From the exploration of the summary results, the best performance are obtained from the RIR simulation methods of ray tracing and hybrid rather than the ISM modeling. The main issue of the ISM simulator is the choice of the

	Classrooms	MAE	MAPE (%)	MIRE (%)	mean STI JNDs(0.03)	mean STI JNDs(0.1)
ISM	University (IDs 1–17)	0.11	27.68	15.29	3.7	1.1
	Primary (IDs 18–24)	0.05	9.34	5.71	1.7	0.5
	Secondary (IDs 25–35)	0.07	13.53	3.64	2.3	0.7
	Overall	0.09	19.56	9.71	2.9	0.9
Ray tracing	University (IDs 1–17)	0.05	13.97	4.71	1.7	0.5
	Primary (IDs 18–24)	0.03	5.79	0.00	1.1	0.3
	Secondary (IDs 25–35)	0.05	10.23	1.82	1.8	0.5
	Overall	0.05	11.16	2.86	1.6	0.5
Hybrid	University (IDs 1–17)	0.05	14.16	4.71	1.7	0.5
	Primary (IDs 18–24)	0.03	5.56	0.00	1.0	0.3
	Secondary (IDs 25–35)	0.05	10.13	1.82	1.8	0.5
	Overall	0.05	11.17	2.86	1.6	0.5

Table 8.4: Comparison of errors between average STIs from measurements and simulations.

maximum order of reflections to be assigned. This model assumes the walls as perfect reflectors, so later reflections due to scattering are not considered, and the activation of the reverberant tail is performed through a sufficiently large number of assigned reflections. This aspect affects the reverberation time estimation and, consequently, the STI values. With regard to simulations using pure ray tracing and hybrid ISM/ray tracing methods, the most accurate modeling of RIRs is provided by ray-tracing-based methods due to the contribution of scattering in the simulation of diffuse reflections. Examining the results for groups of same-grade classrooms, all three simulation methods report more accurate STIs for primary classrooms, and in particular, the hybrid method yields the lowest MAE and MAPE values of 0.03 and 5.56%, respectively, correct intelligibility ratings in all classrooms, and mean STI JNDs within both thresholds. Next, the secondary classrooms also present the best achievements with the hybrid method, returning MAE equal to 0.05, MAPE equal to 10.13%, MIRE of 1.82% (1 incorrect prediction out of 11 classrooms), mean STI JNDs(0.03) equal to 1.8, and mean STI JNDs(0.1) within the threshold. Finally, for university classrooms, the ray tracing method gives MAE of 0.05, MAPE of 13.97%, MIRE of 4.71% (4 incorrect predictions out of 17), mean STI JNDs(0.03) of 1.7, and mean STI JNDs(0.1) within the threshold.

Assessment of Speech Intelligibility through Deep Learning Models

As in the previous assessment of speech intelligibility through pyeSTImate, Table 8.5 reports the overall summary of the results calculated for the same grade and overall classrooms. Specifically, the comparison of errors between average STIs computed by the ANN and pyeSTImate, and between the ANN outcomes and measurements are presented.

From the analysis of the results, the lowest STI errors in individual, same-

	Classrooms	MAE	MAPE (%)	MIRE (%)	mean STI JNDs(0.03)	mean STI JNDs(0.1)
Comparison with *pyeSTImate*	University (ID 1–17)	0.07	15.39	9.41	2.4	0.7
	Primary (ID 18–24)	0.04	8.54	5.71	1.5	0.4
	Secondary (ID 25–35)	0.04	8.14	3.64	1.3	0.4
	Overall	0.06	11.74	6.86	1.9	0.6
Comparison with measurements	University (ID 1–17)	0.10	27.28	11.76	3.5	1.0
	Primary (ID 18–24)	0.05	9.03	5.71	1.6	0.5
	Secondary (ID 25–35)	0.07	13.34	5.45	2.2	0.7
	Overall	0.08	19.25	8.57	2.7	0.8

Table 8.5: Comparison of errors between average STIs from ANN and pyeSTImate, and from ANN and measurements.

grade and overall classrooms are found between ANN predictions and tool outputs. The motivation is related to the dataset used to train the ANN because the neural network learned the relationship between input data and STIs generated by the simulation tool, so the inaccuracy of the deep learning model must be added to the one associated with the simulator. In fact, the comparison with measurements returns in most case studies amplified errors with respect to the comparison with pyeSTImate outcomes.

8.1.3 Remarks

This research suggests an important role of prediction methods in speech intelligibility in the main acoustic feature dimensions. The results have shown that comparable measurements and calculations starting from real input data are surprisingly informative on the level of acoustic detail and the degree to which listeners are able to utilize it for speech comprehension. In the experiments, the pyeSTImate tool has shown good accuracy in predicting speech transmission indices for small and wide classrooms, regardless of grade, year of construction, and finishing materials. Also, the implementation of an artificial neural network that can predict the speech transmission index in lecture rooms with a discrete approximation and a reduced number of input data can be considered a fast method for preliminary assessments of speech intelligibility in classrooms.

In summary, the predictive tool has demonstrated versatility and computational robustness that enable its use for preliminary assessments of speech intelligibility, to design the optimal type of scholar buildings and for sound amplification systems in classrooms in compliance with the Italian regulation. This study represents a starting point for several future works. Some insights for further research include the sensitivity analysis of the tool results to varying absorption and scattering coefficients of materials, the implementation of complex geometries, and the design of an interface for data entry. The deep learning model can be improved with a training set obtained from real measurements or innovative methods for generating synthetic data.

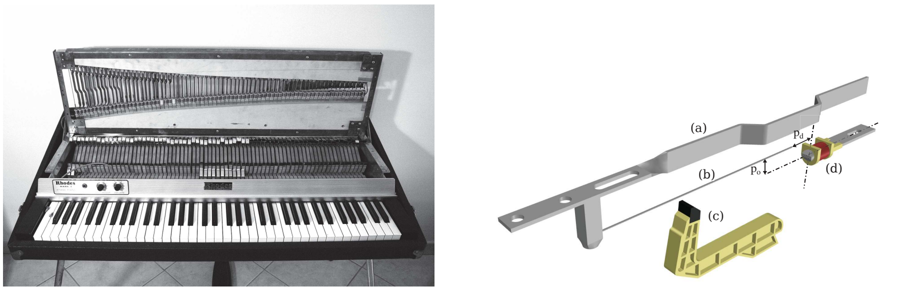
8.2 A Study on the Tone Characterization of the Rhodes Electric Piano

The Rhodes electric piano is an electromechanical keyboard device that was first released in 1946 [179] and was subsequently produced for over four decades, becoming an iconic instrument now commonly known as *the* electric piano. Despite some academic works discussing its operating principle and proposing several physical modeling strategies [180–182], the inharmonic modes that characterize the attack transient have not been the subject of a dedicated investigation. The present study aims to fill this gap by analyzing the spectrum at the pickup output, using a psychoacoustic model to assess the perceptual relevance of the inharmonic components and then conducting scanning laser Doppler vibrometry [183] experiments on the Rhodes asymmetric tuning fork. The investigation compares the modes of the Rhodes piano to those of its individual parts, enabling the extraction of important information about their roles and their relation with natural modes. Based on this analysis, numerical experiments are conducted to demonstrate the intermodulation of the modes caused by the magnetic pickup and to allow the tones produced by the Rhodes from the collected data to be closely matched. With the advent of sampling synthesis and physical modeling, the Rhodes piano has again been the subject of interest from musicians, with a vast number of sampling libraries and digital emulations of its peculiar timbre. For this reason, the extraction of the distribution of the most relevant modes found throughout the keyboard range of the Rhodes piano can give valuable insights for sound synthesis purposes.

8.2.1 Main Components of the Rhodes Piano

The Rhodes electric piano was designed and perfected by Harold Rhodes from World War II to the 1980s. This musical instrument is composed of a harp that hosts asymmetric metal forks, a piano-like action, and a set of pickups, one per fork. The forks are composed of a thin rod called the *tine*, and a bigger beam known as the *tonebar*. The fundamental mechanism of sound production is relatively straightforward: the action of the keyboard expedites a hammer that impacts the tine. Subsequently, the free extremity of the tine oscillates in the presence of an electromagnetic pickup, leading to the production of a time-dependent voltage at the extremity. The electrical signal is subsequently amplified. The keyboard operation comprises a damping mechanism to inhibit tine vibration when a key is released. Additionally, a sustain pedal is provided to preclude dampers from stopping the notes, as in an acoustic piano.

Figure 8.2a shows a picture of a Mark I Stage Rhodes with the cover lid removed and the harp lifted from its operating position. The asymmetric tuning



(a) Overall view of Mark I Stage Rhodes. (b) Details of the Rhodes F1 tuning fork.

Figure 8.2: The Rhodes electric piano: overall view and tuning fork detail.

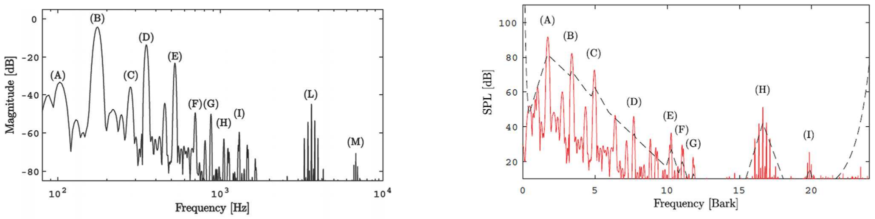
forks and their pickups are shown upside down, on top. The hammers and the tine dampers are standing at rest position below. On the front panel, the Rhodes hosts two knobs and an output jack connector left right above the keyboard. Figure 8.2b depicts a 3D rendering of the asymmetric tuning fork of the Rhodes piano, the resonating element patented by Harold Rhodes. The tuning fork comprises two prongs with different shapes and masses connected by a metal joint: the component in (a) represents the tonebar that also has the function to constrain the fork to the wooden harp, and (b) depicts the tine that is hit by a plastic hammer with neoprene tip (c). Finally, in (d), the pickup at the end of the tine is also shown. The pickup-tine distance p_d and offset p_0 are particularly important in determining the spectral envelope, as they greatly impact the distortions added to the resulting spectrum.

8.2.2 Experimental Analysis of the Inharmonic Overtones

Modal Analysis of the Tones at Pickups

As the Rhodes is an electroacoustic instrument, modal analysis has been carried out by recording several tones from the output of pickups. Each tone has been captured via the piano output jack at a 48 kHz sampling rate, using a Focusrite 2i2 soundcard. The frequency domain analysis has been performed through discrete Fourier transform using 8192 bins and a Blackman-Harris window.

Figure 8.3a presents the DFT of the attack of a *pp* (*pianissimo*) F3 tone, exhibiting the fundamental f_0 and the first five harmonics, along with several inharmonic tones. Earlier research suggested that the tine oscillates following a sinusoidal motion; thus, all harmonics should be generated by the pickup nonlinearity. However, some modes arise that are not harmonically related to the fundamental (B). Of these, mode (A) is below the fundamental, and mode (C) is separated from it by exactly f_0. Modes (I), (L) and (M) have no harmonic relation with the fundamental, and each displays lower and higher ancillary



Marker	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)	(I)	(L)	(M)
Frequency [Hz]	102.5	175.8	278.2	351.6	527.3	703.2	879	1054.8	1307	3633	6814
Ratio	0.58	1.0	1.58	2.0	3.0	4.0	5.0	6.0	7.4	20.66	38.76

(a) Spectrum of a *pp* F3 tone attack, recorded at the pickup output for 200 ms after the onset.

Marker	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)	(I)
Frequency [Hz]	175.8	351.6	527.3	879	1288/1307	1462/1481	1639/1657	3633	6814
Ratio	1.0	2.0	3.0	5.0	7.3/7.4	8.3/8.4	9.3/9.4	20.66	38.76

(b) Audibility of the inharmonic tones and their sidebands of the F3 pickup tone.

Figure 8.3: Modal analysis at the pickup and related psychoacoustic model.

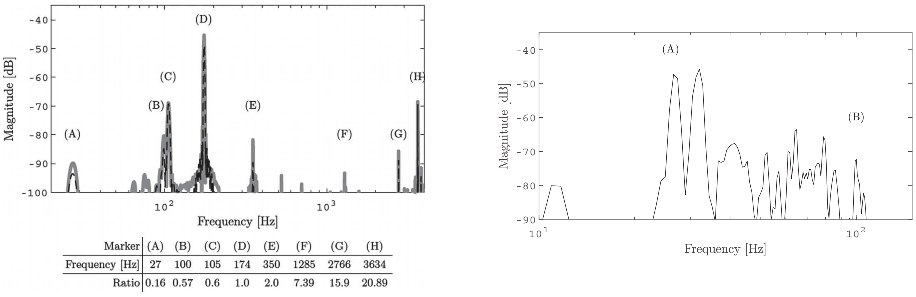
modes that are spaced by exactly f_0. This aspect suggests some modulation occurring between the fundamental and these inharmonic modes, referred to as sideband *partials* or, simply, *sidebands*.

A psychoacoustic model that considers the effects of frequency masking [184] according to a Bark-scale asymmetric spreading function shifted down by 10 dB, and the hearing threshold [185] has been employed to understand the perceptual relevance of the inharmonic components. The curve obtained from this operation is compared to the F3 pickup tone in Figure 8.3b. As can be seen, some of the harmonics (A–D) are louder than the audibility threshold. The same applies to mode (E) and some of its sideband partials (F–G). Partials (H) and (I) are clearly perceivable as well as their sidebands.

Modal Analysis of the Tuning Fork

The presence of strong non-linearities affects the tone of the Rhodes, adding components potentially absent in the tuning fork. Their evidence in the pickup output tone requires an investigation method that allows direct measurement of the tuning fork. A scanning laser Doppler vibrometer (SLDV) has been used to analyze the operational vibration of the tuning forks in a Rhodes piano. The SLDV system consists of a Polytec laser doppler vibrometer head (OFV-303), a velocity decoder, a controller, and a personal computer for data processing. The setup measures the operational deflection shapes of the tuning forks allowing for the identification of modes and their vibration in space.

Experiments have been conducted to measure the free response of the entire tuning fork assembly, with both the tine and the tonebar being probed. Additionally, the tuning forks have been disassembled, and both components have been stimulated to measure the uncoupled free response of each one alone. Extensive measurements have been carried out, examples of which are mentioned.



Marker	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)
Frequency [Hz]	27	100	105	174	350	1285	2766	3634
Ratio	0.16	0.57	0.6	1.0	2.0	7.39	15.9	20.89

(a) Spectra of an F3 tonebar hit by the piano hammer with *pp* (black dashed line) and *mf* (gray thick line) dynamics.

(b) Modes of the Rhodes wood harp obtained by averaging nine points.

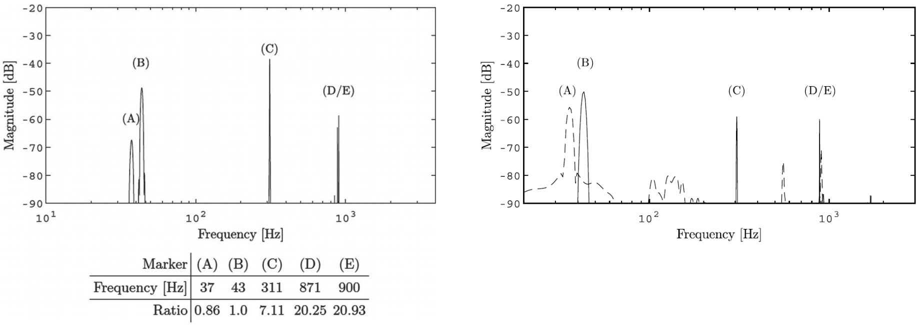
Figure 8.4: Spectra of an F3 tonebar and the wood harp after hammer impact.

An experiment involving the tonebar assembled and stimulated by means of the corresponding hammer at the installation position is described in Figure 8.4. Figure 8.4a shows the comparison between the spectra of an F3 tonebar hit by the piano hammer with *pp* and *mf* dynamics. The two tones exhibit several modes below the fundamental (marker D) and overtones with different magnitudes. The third and fourth harmonics appear in the *mf* tones (gray lines between E and F) due to their increased magnitude. In Figure 8.4b, the Rhodes wooden harp after the F3 key percussion presents several modes between 10 Hz and 100 Hz and only modes (A) and (B).

The same experiment has been performed by pointing the laser of the vibrometer at the tine mounted on the instrument, as shown in Figure 8.5a. Figure 8.5b displays the spectra extracted from the disassembled F1 tine and tonebar separately. As can be seen, the subfundamental is present in the tonebar only, as well as a small tone (E), very close to (D).

Summarizing the SLDV analyses, the following conclusions can be drawn about the vibration of the asymmetric tuning fork.

- The tine and the tonebar both show a strong fundamental, as observed by high-speed camera recordings from [181].
- At least one sub-fundamental mode, introduced by the tonebar, is always present but with a variable frequency.
- Inharmonic modes with ratio 7× and 20× are often observed on F1 and F3 notes, while the mode at 39× has been more rarely seen.
- No evidence of sidebands can be found in SLDV experiments.
- The harp is responsible for several modes, however, these can be observed only in vertical transverse tonebar oscillations and are not found in the



Marker	(A)	(B)	(C)	(D)	(E)
Frequency [Hz]	37	43	311	871	900
Ratio	0.86	1.0	7.11	20.25	20.93

(a) Spectral content of the tine of F1 Rhodes tuning fork mounted on the instrument.

(b) Spectra of individual tine (solid line) and tonebar (dashed line) of the F1 Rhodes tuning fork.

Figure 8.5: Comparison between the spectra of the F1 tine mounted on the instrument and F1 tine and tonebar analyzed separately.

tine motion.

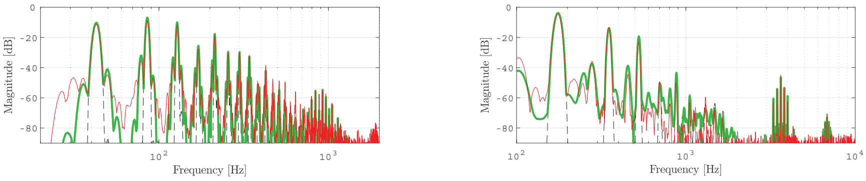
- The tuning mass has little effect on the sub-fundamental modes, while it greatly affects the fundamental and the higher natural modes, however, with no clear correlation with the change in pitch of the fundamental.
- The same applies to the presence of the tonebar that can influence the frequency of the modes by a small amount with no clear correlation. The most important contribution of the tonebar is the enhancement of the fundamental sustain and reduction of the duration of other overtones.

Effect of the Magnetic Pickup

The magnetic pickup is responsible for the generation of harmonic overtones in response to the approximately sinusoidal oscillation of the tine. In addition to harmonic overtones, the Rhodes tone also presents inharmonic overtones with sideband partials as seen in Figure 8.3a. Since no evidence for these partials has been found in the laser vibrometer experiments, it can be argue that this is an effect of the pickup non-linearity. To assess this thesis a discrete-time model based on the modal analysis has been constructed. The model is formulated as follows.

$$y(t) = \frac{dg(x(t))}{dt}, \tag{8.2}$$

where $g(\cdot)$ is the magnetic field and $x(t)$ is the tine displacement, approximated as a sum of decaying sinusoidal modes and the pickup-tine offset p_o:



(a) F1 simulated tone and pickup output.

(b) F3 simulated tone and pickup output.

Figure 8.6: Simulated tone with the sub-fundamental (thick gray line) and without the sub-fundamental mode (dashed), compared to the recorded pickup output (solid thin line).

$$x(t) = p_o + \sum_i A_i e^{-\lambda t} \cdot sin(2\pi f_i t), \tag{8.3}$$

where A_i is the amplitude of the ith mode and λ determines the rate of the exponentially-decaying envelope. The magnetic field is calculated following [181] and stored in a lookup table [186], linearly interpolated to generate the output. The discrete-time derivative is approximated using the backward scheme $x[n] - x[n-1]$.

The frequency, amplitude and decay rate of the main modes of a F1 and F3 tones have been extracted. Specifically, the fundamental, the sub-harmonic, and the three main inharmonic modes, detailed in Table 8.6 have been taken. To complete the model, the pickup-tine distance and offset have been measured.

F1				**F3**			
Frequency	Ratio	Amp	Decay	Frequency	Ratio	Amp	Decay
35.8 Hz	0.83	-38 dB	-9.1 dB/s	102.5 Hz	0.58	-9 dB	-138 dB/s
42.7 Hz	1	-10 dB	-8.2 dB/s	175.8 Hz	1	-4.5 dB	-12 dB/s
306.2 Hz	7.2	-65 dB	-21.1 dB/s	1307 Hz	7.4	-59.5 dB	-294 dB/s
881.8 Hz	20.6	-65 dB	-67.7 dB/s	3636 Hz	20.7	-54.8 dB	-37 dB/s
				6814 Hz	38.7	-46.8 dB	-161 dB/s

Table 8.6: Modes frequency, magnitude and decay detected from SLDV tine recordings used in the simulations: F1 and F3.

The outcomes of the model are shown in Figure 8.6 compared to the pickup output. As can be seen, the match of the first harmonics is very close, as well as the amplitude and frequency of the inharmonic tones and their sidebands. Furthermore, the presence of the sub-fundamental approximates quite well some of the sidelobes seen in the pickup output. Additional low frequency modes could be estimated to synthesize the missing modes, if these are perceptually relevant.

8.2.3 Remarks

The tone production mechanism of the Rhodes electric piano has been studied, focusing on the inharmonic components, as they constitute a notable aspect of the timbre. The findings of the experiments demonstrate that they can derive from the natural modes of the tuning fork or sidebands generated by intermodulation at the pickup. Despite having a small amplitude, these components have been shown to be perceptually relevant using a psychoacoustical model. A laser doppler vibrometer has been used to analyze the modes of the two main prongs of the Rhodes fork, namely, the tonebar and the tine. The inharmonic overtones in the Rhodes timbre are shown to be related to transverse modes of the tine and have specific ratios to the fundamental. Five natural modes of vibration have been identified: the sub-fundamental, the fundamental, the second mode, the third mode, and the fourth mode. The role of the tonebar has also been assessed, showing that it produces the sub-harmonic tone and enhances the sustain of the fundamental. It has been demonstrated that the pickup generates the sidebands, and the model is sufficiently precise in fitting the observed Rhodes spectra using a signal-based approximation, including the generation of these modes and the pickup non-linearity. After these experiments, the ratio of the first six natural modes can be extracted from recordings of Rhodes tones, discarding the sidebands. The results are consistent with the laser vibrometer experiments and allowed for finding two additional natural modes in the lower range of the instrument. The findings provided in this research may serve as a starting point for further studies, e.g., regarding new materials and production processes for the asymmetric tuning fork or digital algorithms for modal synthesis.

Chapter 9

Conclusions and Future Perspectives

In this dissertation, deep learning approaches applied to audio signal processing in the automotive field have been investigated, paying particular attention to the topic of emergency siren detection. The potential of deep learning algorithms in this research area has significant practical implications. By accurately detecting the sounds of emergency sirens and alerting their proximity to the driver, these computational models implemented in emergency vehicle detection systems can improve driver and passenger safety and reduce the response time of emergency services.

The first three chapters of this thesis provided an overview of the main motivations behind this research and described the elements that involve the system design process. In Chapter 1, the topic of emergency siren detection has been introduced with a historical survey of the technologies developed in patents of emergency vehicle detection systems. In a review of the state-of-the-art, the main works concerning algorithmic innovations have been outlined with reference to digital signal processing methodologies and the more recent and more effective machine and deep learning approaches. A summary of the rationale and contributions featured in this research has also been presented. Chapter 2 provided an overview of the theoretical background of the neural networks and related optimization, generalization and regularization strategies used in the development of the systems. The metrics for evaluating the performance of the algorithms have been briefly explained. Chapter 3 presented the methodologies for creating datasets of Italian ambulance siren sounds, moving from an algorithmic approach to real-world data collection. The acoustic features used for time-frequency representations have been described from a computational point of view.

The following two chapters detailed neural approaches and research achievements on the emergency siren detection task. In Chapter 4, synthetic datasets have been designed to train convolutional models and test their performance. The accuracy obtained at this stage confirmed the reliability of convolutional neural networks in detecting emergency siren sounds, even in noisy environments. These results encouraged the development of strategies to reduce the computational cost of the algorithm without loss of accuracy. The study revealed the effectiveness of harmonic filtering in enhancing the tonal components of the siren and reducing background noise, which has also been employed in follow-up studies. Chapter 5 addressed the detection of emergency siren sounds recorded in real-world contexts using meta-learning approaches. Prototypical networks trained with datasets unrelated to the target task have been able to compute high-performance neural models. They recognized similar instances of classes unseen in training from a reduced dataset without relying on domain adaptation strategies. In this context, the research also focused on the performance of multiple recording sensors installed in the car. From the results, it has been possible to gain insights on the most suitable microphone installation, significant for developing an emergency vehicle detection system.

After the algorithmic research, Chapter 6 presented a novel prototype of an emergency vehicle detection device leveraging deep learning algorithms. The system relies on computational audio processing to detect an emergency vehicle approaching. Then, the driver awareness verification is performed with computer vision techniques applied to face and gaze movements that condition the alerting system. The prototype has been tested in a demo performed in the controlled environment of a semianechoic chamber. Finally, Chapter 7 introduced a dataset recorded with a research vehicle fitted with audio and video sensors driven on public roads. The dataset comprises more than 30 hours of data for each sensor, collected by covering 1500 km on diverse roads and landscapes under varying weather conditions, both day and night. The annotations provide all the necessary information for scene understanding. This extensive dataset, analyzed with artificial neural networks, can be leveraged to develop novel driving assistance technologies that rely solely on audio or video data or employ a combination of both.

Overall, this dissertation has contributed to the advancement of the field of sound detection in the automotive industry and has demonstrated the potential of deep learning algorithms to address real-world problems related to sound processing. Future research could further explore the topics faced in this study. The work on emergency siren detection has provided insights for improving current performance and extending the investigation to further applications. While improvements mainly concern the reduction of the latency time and the recognition of sirens from countries worldwide, future developments can deal with localization and tracking to be implemented in the prototype of the emergency vehicle detection system. The device thus designed could also be combined with a detector of the presence of vehicles alongside the car to enable automatic pulling over. The dataset for the acoustic scene understanding has also left wide scope for new research, starting from the realization of a baseline of acoustic-visual scene recognition to the extension to additional tasks such as urban sound event detection and classification, as well as the recognition of objects, road breakdowns and out-of-sight obstacles.

List of Publications

[1] M. Cantarini, L. Serafini, L. Gabrielli, E. Principi, and S. Squartini (2020). “Emergency siren recognition in urban scenarios: Synthetic dataset and deep learning models.” In *Intelligent Computing Theories and Application: 16th International Conference, ICIC 2020, Bari, Italy, October 2–5, 2020, Proceedings, Part I 16* (pp. 207-220). Springer International Publishing. `https://doi.org/10.1007/978-3-030-60799-9_18`

[2] M. Cantarini, A. Brocanelli, L. Gabrielli, and S. Squartini (2021, September). “Acoustic features for deep learning-based models for emergency siren detection: An evaluation study.” In *2021 12th International Symposium on Image and Signal Processing and Analysis (ISPA)* (pp. 47-53). IEEE. `https://doi.org/10.1109/ISPA52656.2021.9552140`

[3] M. Cantarini, L. Gabrielli, and S. Squartini (2022). “Few-Shot Emergency Siren Detection.” *Sensors, 22*(12), 4338. `https://doi.org/10.3390/s22124338`

[4] M. Cantarini, L. Gabrielli, L. Migliorelli, A. Mancini, and S. Squartini (2023, February). “Beware the Sirens: Prototyping an Emergency Vehicle Detection System for Smart Cars.” In *Applied Intelligence and Informatics: Second International Conference, AII 2022, Reggio Calabria, Italy, September 1–3, 2022, Proceedings* (pp. 437-451). Cham: Springer Nature Switzerland. `https://doi.org/10.1007/978-3-031-24801-6_31`

[5] M. Cantarini, L. Gabrielli, A. Mancini, S. Squartini, and R. Longo (2023). “A3CarScene: An audio-visual dataset for driving scene understanding.” *Data in Brief*, 109146. `https://doi.org/10.1016/j.dib.2023.109146`

[6] M. Cantarini, L. Migliorelli, L. Gabrielli, A. Mancini, and S. Squartini (2023). “A Multimodal Driving-Assistance Prototype for Emergency-Vehicles Detection.” *Integrated Computer-Aided Engineering* (submitted).

These works have been supported by Marche Region in implementation of the financial programme POR MARCHE FESR 2014-2020, project “Miracle” (Marche Innovation and Research fAcilities for Connected and sustainable Living Environments), CUP B28I19000330007.

Others:

[1] L. Gabrielli, M. Cantarini, P. Castellini, and S. Squartini (2020). "The Rhodes electric piano: Analysis and simulation of the inharmonic overtones." *The Journal of the Acoustical Society of America, 148*(5), 3052-3064. `https://doi.org/10.1121/10.0002002`

[2] S. Di Loreto, M. Cantarini, S. Squartini, V. Lori, F. Serpilli, and C. Di Perna (2023). "Assessment of speech intelligibility in scholar classrooms by measurements and prediction methods." *Building Acoustics*, 1351010X231158190. `https://doi.org/10.1177/1351010X231158190`

Bibliography

[1] F. Arena, G. Pau, and A. Severino, "An overview on the current status and future perspectives of smart cars", *Infrastructures*, vol. 5, no. 7, p. 53, 2020. DOI: `10.3390/infrastructures5070053`.

[2] T. K. Chan and C. S. Chin, "Review of autonomous intelligent vehicles for urban driving and parking", *Electronics*, vol. 10, no. 9, p. 1021, 2021. DOI: `10.3390/electronics10091021`.

[3] V. K. Kukkala, J. Tunnell, S. Pasricha, and T. Bradley, "Advanced driver-assistance systems: A path toward autonomous vehicles", *IEEE Consumer Electronics Magazine*, vol. 7, no. 5, pp. 18–25, 2018. DOI: `10.1109/MCE.2018.2828440`.

[4] E. Marti, M. A. De Miguel, F. Garcia, and J. Perez, "A review of sensor technologies for perception in automated driving", *IEEE Intelligent Transportation Systems Magazine*, vol. 11, no. 4, pp. 94–108, 2019. DOI: `10.1109/MITS.2019.2907630`.

[5] X. Zeng, F. Wang, B. Wang, C. Wu, K. R. Liu, and O. C. Au, "In-vehicle sensing for smart cars", *IEEE Open Journal of Vehicular Technology*, 2022. DOI: `10.1109/OJVT.2022.3174546`.

[6] Y. Schulz, A. K. Mattar, T. M. Hehn, and J. F. Kooij, "Hearing what you cannot see: Acoustic vehicle detection around corners", *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2587–2594, 2021. DOI: `10.1109/LRA.2021.3062254`.

[7] R. Avanzato, F. Beritelli, F. Di Franco, and V. F. Puglisi, "A convolutional neural networks approach to audio classification for rainfall estimation", in *2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, IEEE, vol. 1, 2019, pp. 285–289. DOI: `10.1109/IDAACS.2019.8924399`.

[8] J Alonso *et al.*, "On-board wet road surface identification using tyre/road noise and support vector machines", *Applied acoustics*, vol. 76, pp. 407–415, 2014. DOI: `10.1016/j.apacoust.2013.09.011`.

[9] G. Pepe, L. Gabrielli, L. Ambrosini, S. Squartini, and L. Cattani, "Detecting road surface wetness using microphones and convolutional neural networks", in *Audio Engineering Society Convention 146*, Audio Engineering Society, 2019.

[10] C Ramos-Romero, P. León-Ríos, B. M. Al-Hadithi, L. Sigcha, G De Arcas, and C Asensio, "Identification and mapping of asphalt surface deterioration by tyre-pavement interaction noise measurement", *Measurement*, vol. 146, pp. 718–727, 2019. DOI: `10.1016/j.measurement.2019.06.034`.

[11] F. G. Praticò, R. Fedele, V. Naumov, and T. Sauer, "Detection and monitoring of bottom-up cracks in road pavement using a machine-learning approach", *Algorithms*, vol. 13, no. 4, p. 81, 2020. DOI: `10.3390/a13040081`.

[12] S. Djukanović, J. Matas, and T. Virtanen, "Acoustic vehicle speed estimation from single sensor measurements", *IEEE Sensors Journal*, vol. 21, no. 20, pp. 23 317–23 324, 2021. DOI: `10.1109/JSEN.2021.3110009`.

[13] G. Szwoch and J. Kotus, "Acoustic detector of road vehicles based on sound intensity", *Sensors*, vol. 21, no. 23, p. 7781, 2021. DOI: `10.3390/s21237781`.

[14] F. P. Weber and M. J. Carmody, *Discriminating acoustic signal detector*, US Patent 2,545,218, 1951.

[15] R. Lawson, *Emergency vehicle warning system*, US Patent 7,061,402, 2006.

[16] A. D. Bachelder and C. F. Foster, *Emergency vehicle traffic signal preemption system*, US Patent 7,327,280, 2008.

[17] L. Chan *et al.*, *Systems and methods for providing awareness of emergency vehicles*, US Patent 10,584,518, 2020.

[18] A. G. Lemmons and S. B. Riley, *Emergency vehicle alarm system and method*, US Patent 8,258,979, 2012.

[19] L. McKenna, *Emergency vehicle alarm system for vehicles*, US Patent 5,495,243, 1996.

[20] D. Agnew, S. Lüke, M. Fischer, and D. Krökel, *Emergency vehicle detection with digital image sensor*, US Patent 9,576,208, 2017.

[21] V. Yaldo and X. F. Song, *Systems and methods for emergency vehicle response in an autonomous vehicle*, US Patent 10,431,082, 2019.

[22] L. G. Dill and R. B. Molitor, *Siren actuated warning device for automobiles*, US Patent 3,014,199, 1961.

[23] B. Stefanov, *Wailing siren detecting circuit*, US Patent 4,158,190, 1979.

[24] B. E. Warren, *Vehicle warning system*, US Patent 4,587,522, 1986.

[25] B. Bernstein and G. L. Sohie, *Emergency signal warning system*, US Patent 4,956,866, 1990.

[26] W. E. Brill, *Emergency vehicle detection system*, US Patent 6,362,749, 2002.

[27] O. Watkins *et al.*, *Emergency siren detection for autonomous vehicles*, US Patent App. 17/073,680, 2022.

[28] X. Kecheng *et al.*, *Emergency vehicle audio and visual detection post fusion*, US Patent App. 17/149,638, 2022.

[29] P. Schmitt, J. Buddhadev, and A. H. Lang, *Emergency vehicle detection system and method*, US Patent 11,364,910, 2022.

[30] L. Nanni, G. Maguolo, S. Brahnam, and M. Paci, "An ensemble of convolutional neural networks for audio classification", *Applied Sciences*, vol. 11, no. 13, p. 5796, 2021. DOI: 10.3390/app11135796.

[31] B. Bayram and G. İnce, "An incremental class-learning approach with acoustic novelty detection for acoustic event recognition", *Sensors*, vol. 21, no. 19, p. 6622, 2021. DOI: 10.3390/s21196622.

[32] F. Meucci, L. Pierucci, E. Del Re, L Lastrucci, and P Desii, "A real-time siren detector to improve safety of guide in traffic environment", in *2008 16th European Signal Processing Conference*, IEEE, 2008, pp. 1–5.

[33] T. Miyazakia, Y. Kitazonoa, and M. Shimakawab, "Ambulance siren detector using FFT on dsPIC", in *Proceedings of the 1st IEEE/IIAE International Conference on Intelligent Systems and Image Processing*, 2013, pp. 266–269. DOI: 10.12792/icisip2013.052.

[34] J.-J. Liaw, W.-S. Wang, H.-C. Chu, M.-S. Huang, and C.-P. Lu, "Recognition of the ambulance siren sound in taiwan by the longest common subsequence", in *2013 IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, 2013, pp. 3825–3828. DOI: 10.1109/SMC.2013.653.

[35] S. Kiran and M Supriya, "Siren detection and driver assistance using modified minimum mean square error method", in *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*, IEEE, 2017, pp. 127–131. DOI: 10.1109/SmartTechCon.2017.8358355.

[36] Y. Ebizuka, S. Kato, and M. Itami, "Detecting approach of emergency vehicles using siren sound processing", in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2019, pp. 4431–4436. DOI: 10.1109/ITSC.2019.8917028.

[37] H. V. Supreeth, S. Rao, K. Chethan, and U Purushotham, "Identification of Ambulance Siren sound and Analysis of the signal using statistical method", in *2020 International Conference on Intelligent Engineering and Management (ICIEM)*, IEEE, 2020, pp. 198–202. DOI: `10.1109/ICIEM48762.2020.9160070`.

[38] F Beritelli, S Casale, A Russo, and S Serrano, "An automatic emergency signal recognition system for the hearing impaired", in *2006 IEEE 12th Digital Signal Processing Workshop & 4th IEEE Signal Processing Education Workshop*, IEEE, 2006, pp. 179–182. DOI: `10.1109/DSPWS.2006.265438`.

[39] J. Schröder, S. Goetze, V. Grützmacher, and J. Anemüller, "Automatic acoustic siren detection in traffic noise by part-based models", in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2013, pp. 493–497. DOI: `10.1109/ICASSP.2013.6637696`.

[40] D. Carmel, A. Yeshurun, and Y. Moshe, "Detection of alarm sounds in noisy environments", in *2017 25th European Signal Processing Conference (EUSIPCO)*, IEEE, 2017, pp. 1839–1843. DOI: `10.23919/EUSIPCO.2017.8081527`.

[41] L. Marchegiani and P. Newman, "Listening for sirens: Locating and classifying acoustic alarms in city scenes", *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 17 087–17 096, 2022. DOI: `10.1109/TITS.2022.3158076`.

[42] O. Ronneberger, P.Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation", in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, ser. LNCS, vol. 9351, Springer, 2015, pp. 234–241. DOI: `10.1007/978-3-319-24574-4_28`.

[43] M. Chavdar, B. Gerazov, Z. Ivanovski, and T. Kartalov, "Towards a system for automatic traffic sound event detection", in *2020 28th Telecommunications Forum (TELFOR)*, IEEE, 2020, pp. 1–4. DOI: `10.1109/TELFOR51502.2020.9306592`.

[44] D. Rane, P. Shirodkar, T. Panigrahi, and S Mini, "Detection of ambulance siren in traffic", in *2019 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET)*, IEEE, 2019, pp. 401–405. DOI: `10.1109/WiSPNET45539.2019.9032797`.

[45] S. Padhy, J. Tiwari, S. Rathore, and N. Kumar, "Emergency signal classification for the hearing impaired using multi-channel convolutional neural network architecture", in *2019 IEEE Conference on Information and Communication Technology*, IEEE, 2019, pp. 1–6. DOI: `10.1109/CICT48419.2019.9066252`.

[46] A. Raman, S Kaushik, K. R. Rao, and M. Moharir, "A hybrid framework for expediting emergency vehicle movement on indian roads", in *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, IEEE, 2020, pp. 459–464. DOI: `10.1109/ICIMIA48430.2020.9074933`.

[47] B. Fatimah, A Preethi, V Hrushikesh, A. Singh, and H. R. Kotion, "An automatic siren detection algorithm using Fourier Decomposition Method and MFCC", in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, IEEE, 2020, pp. 1–6. DOI: `10.1109/ICCCNT49239.2020.9225414`.

[48] V.-T. Tran and W.-H. Tsai, "Acoustic-based emergency vehicle detection using convolutional neural networks", *IEEE Access*, vol. 8, pp. 75 702–75 713, 2020. DOI: `10.1109/ACCESS.2020.2988986`.

[49] V.-T. Tran and W.-H. Tsai, "Audio-vision emergency vehicle detection", *IEEE Sensors Journal*, vol. 21, no. 24, pp. 27 905–27 917, 2021. DOI: `10.1109/JSEN.2021.3127893`.

[50] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788. DOI: `10.1109/CVPR.2016.91`.

[51] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning", *Advances in neural information processing systems*, vol. 30, 2017. DOI: `10.48550/arXiv.1703.05175`.

[52] I. Lee and K. Lee, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises", *Business horizons*, vol. 58, no. 4, pp. 431–440, 2015. DOI: `10.1016/j.bushor.2015.03.008`.

[53] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, 1943. DOI: `10.1007/BF02478259`.

[54] H. Do, "The organization of behavior", *New York*, 1949.

[55] A. M. Turing, *Computing machinery and intelligence.* Springer, 2009. DOI: `10.1007/978-1-4020-6710-5_3`.

[56] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain", *Psychological review*, vol. 65, no. 6, p. 386, 1958. DOI: `10.1037/h0042519`.

[57] B. Widrow and M. E. Hoff, "Adaptive switching circuits", Stanford Univ Ca Stanford Electronics Labs, Tech. Rep., 1960.

[58] M. Minsky and S. Papert, *Perceptrons: An essay in computational geometry*, 1969.

[59] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences", *PhD thesis, Committee on Applied Mathematics, Harvard University, Cambridge, MA*, 1974.

[60] K. Fukushima and S. Miyake, "Neocognitron: Self-organizing network capable of position-invariant recognition of patterns", in *Proc. 5th Int. Conf. Pattern Recognition*, vol. 1, 1980, pp. 459–461.

[61] T. Kohonen, "The self-organizing map", *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990. DOI: `10.1109/5.58325`.

[62] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982. DOI: `10.1073/pnas.79.8.2554`.

[63] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors", *nature*, vol. 323, no. 6088, pp. 533–536, 1986. DOI: `10.1038/323533a0`.

[64] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition", *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989. DOI: `10.1162/neco.1989.1.4.541`.

[65] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series", *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[66] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: `10.1162/neco.1997.9.8.1735`.

[67] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks", *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997. DOI: `10.1109/78.650093`.

[68] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets", *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006. DOI: `10.1162/neco.2006.18.7.1527`.

[69] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. DOI: `10.48550/arXiv.1512.03385`.

[70] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting", *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[71] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate", *arXiv preprint*, 2014. DOI: `10.48550/arXiv.1409.0473`.

[72] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, "Improving language understanding by generative pre-training", 2018.

[73] I. Goodfellow *et al.*, "Generative adversarial networks", *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020. DOI: `10.1145/3422622`.

[74] S. Haykin, "Adaptive systems for signal process", in *Advanced Signal Processing*, CRC Press, 2017, pp. 25–78.

[75] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)", *arXiv preprint arXiv:1511.07289*, 2015. DOI: `10.48550/arXiv.1511.07289`.

[76] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014. DOI: `10.48550/arXiv.1412.6980`.

[77] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors", *arXiv preprint arXiv:1207.0580*, 2012. DOI: `10.48550/arXiv.1207.0580`.

[78] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", in *International conference on machine learning*, pmlr, 2015, pp. 448–456.

[79] J. F. Gemmeke *et al.*, "Audio set: An ontology and human-labeled dataset for audio events", in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 2017, pp. 776–780. DOI: `10.1109/ICASSP.2017.7952261`.

[80] H. Chen, W. Xie, A. Vedaldi, and A. Zisserman, "Vggsound: A large-scale audio-visual dataset", in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 721–725. DOI: `10.1109/ICASSP40776.2020.9053174`.

[81] M. Asif, M. Usaid, M. Rashid, T. Rajab, S. Hussain, and S. Wasi, "Large-scale audio dataset for emergency vehicle sirens and road noises", *Scientific data*, vol. 9, no. 1, p. 599, 2022. DOI: `10.1038/s41597-022-01727-2`.

[82] K. J. Piczak, "ESC: Dataset for environmental sound classification", in *Proceedings of the 23rd ACM international conference on Multimedia*, 2015, pp. 1015–1018. DOI: 10.1145/2733373.2806390.

[83] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research", in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 1041–1044. DOI: 10.1145/2647868.2655045.

[84] Ministero dei Trasporti, *Decreto Ministeriale 17 ottobre 1980 (G.U. n. 310 del 12.11.1980): Modifiche sperimentali delle caratteristiche acustiche dei dispositivi supplementari di allarme da applicare ad autoveicoli e motoveicoli adibiti a servizi antincendi e ad autoambulanze*, Available online: https://croceitalia.it/pdf/dispositivi_supplementari_allarme.pdf (accessed on 28 February 2023), 1980.

[85] J. Smith, S. Serafin, J. Abel, and D. Berners, "Doppler simulation and the Leslie", in *Proc. Int. Conf. on Digital Audio Effects, Hamburg*, 2002.

[86] J. Smith, "Physical audio signal processing", *online book*, 2010, Available online: http://ccrma.stanford.edu/~jos/pasp/ (accessed on 28 February 2023).

[87] C. Anderton, *The Digital Delay Handbook*. Music Sales Corporation, 1985.

[88] S. Disch and U. Zölzer, "Modulation and delay line based digital audio effects", in *2nd Workshop on Digital Audio Effects DAFx*, 1999.

[89] M. Talbot-Smith, "Sound, speech and hearing", in *Telecommunications Engineer's Reference Book*, Elsevier, 1993, pp. 8–1. DOI: 10.1016/B978-0-7506-1162-6.50014-9.

[90] F. Font, G. Roma, and X. Serra, "Freesound technical demo", in *Proceedings of the 21st ACM international conference on Multimedia*, 2013, pp. 411–412. DOI: 10.1145/2502081.2502245.

[91] G. Pepe, L. Gabrielli, S. Squartini, L. Cattani, and C. Tripodi, "Deep learning for individual listening zone", in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, IEEE, 2020, pp. 1–6. DOI: 10.1109/MMSP48831.2020.9287161.

[92] D. P. Ellis, "Gammatone-like spectrograms", 2009, Available online: https://www.ee.columbia.edu/~dpwe/resources/matlab/gammatonegram/ (accessed on 28 February 2023).

[93] J. Driedger, M. Müller, and S. Disch, "Extending Harmonic-Percussive Separation of Audio Signals", in *ISMIR*, 2014, pp. 611–616.

[94] S. N. Levine and J. O. Smith III, "A sines+ transients+ noise audio representation for data compression and time/pitch scale modifications", in *Audio Engineering Society Convention 105*, Audio Engineering Society, 1998.

[95] D. Fitzgerald, "Harmonic/percussive separation using median filtering", in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, vol. 13, 2010, pp. 1–4.

[96] B. McFee *et al.*, "librosa: Audio and music signal analysis in python", in *Proceedings of the 14th python in science conference*, vol. 8, 2015, pp. 18–25.

[97] S. Khan, H. Rahmani, S. A. A. Shah, M. Bennamoun, G. Medioni, and S. Dickinson, "A guide to convolutional neural networks for computer vision", 2018.

[98] D. Bhatt *et al.*, "CNN variants for computer vision: history, architecture, application, challenges and future scope", *Electronics*, vol. 10, no. 20, p. 2470, 2021. DOI: 10.3390/electronics10202470.

[99] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition", *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014. DOI: 10.1109/TASLP.2014.2339736.

[100] Y. M. Costa, L. S. Oliveira, and C. N. Silla Jr, "An evaluation of convolutional neural networks for music classification using spectrograms", *Applied soft computing*, vol. 52, pp. 28–38, 2017. DOI: 10.1016/j.asoc.2016.12.024.

[101] J. Pons, O. Slizovskaia, R. Gong, E. Gómez, and X. Serra, "Timbre analysis of music audio signals with convolutional neural networks", in *2017 25th European Signal Processing Conference (EUSIPCO)*, IEEE, 2017, pp. 2744–2748. DOI: 10.23919/EUSIPCO.2017.8081710.

[102] A. Bansal and N. K. Garg, "Environmental Sound Classification: A descriptive review of the literature", *Intelligent Systems with Applications*, p. 200 115, 2022. DOI: 10.1016/j.iswa.2022.200115.

[103] K. J. Piczak, "Environmental sound classification with convolutional neural networks", in *2015 IEEE 25th international workshop on machine learning for signal processing (MLSP)*, IEEE, 2015, pp. 1–6. DOI: 10.1109/MLSP.2015.7324337.

[104] M. Huzaifah, "Comparison of time-frequency representations for environmental sound classification using convolutional neural networks", *arXiv preprint arXiv:1706.07156*, 2017. DOI: 10.48550/arXiv.1706.07156.

[105] F. Demir, D. A. Abdullah, and A. Sengur, "A new deep CNN model for environmental sound classification", *IEEE Access*, vol. 8, pp. 66 529–66 537, 2020. DOI: 10.1109/ACCESS.2020.2984903.

[106] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification", *IEEE Signal processing letters*, vol. 24, no. 3, pp. 279–283, 2017. DOI: 10.1109/LSP.2017.2657381.

[107] Z. Zhang, S. Xu, S. Cao, and S. Zhang, "Deep convolutional neural network with mixup for environmental sound classification", in *Pattern Recognition and Computer Vision: First Chinese Conference, PRCV 2018, Guangzhou, China, November 23-26, 2018, Proceedings, Part II 1*, Springer, 2018, pp. 356–367. DOI: 10.1007/978-3-030-03335-4_31.

[108] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks", *arXiv preprint arXiv:1606.00298*, 2016. DOI: 10.48550/arXiv.1606.00298.

[109] A. Mesaros, T. Heittola, T. Virtanen, and M. D. Plumbley, "Sound event detection: A tutorial", *IEEE Signal Processing Magazine*, vol. 38, no. 5, pp. 67–83, 2021. DOI: 10.1109/MSP.2021.3090678.

[110] X. Valero and F. Alias, "Gammatone cepstral coefficients: Biologically inspired features for non-speech audio classification", *IEEE transactions on multimedia*, vol. 14, no. 6, pp. 1684–1689, 2012. DOI: 10.1109/TMM.2012.2199972.

[111] A Queiroz and R. Coelho, "F0-based gammatone filtering for intelligibility gain of acoustic noisy signals", *IEEE Signal Processing Letters*, vol. 28, pp. 1225–1229, 2021. DOI: 10.1109/LSP.2021.3084561.

[112] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: a big comparison for NAS", *arXiv preprint*, 2019. DOI: 10.48550/arXiv.1912.06059.

[113] F. Chollet *et al.*, *Keras*, Available online: https://keras.io (accessed on 28 February 2023), 2015.

[114] T. Developers, "TensorFlow", *Zenodo*, 2021. DOI: 10.5281/zenodo.4758419.

[115] M. Fink, "Object classification from a single example utilizing class relevance metrics", *Advances in neural information processing systems*, vol. 17, pp. 449–456, 2005.

[116] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories", *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006. DOI: 10.1109/TPAMI.2006.79.

[117] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning", *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–34, 2020. DOI: 10.1145/3386252.

[118] A. Parnami and M. Lee, "Learning from few examples: A summary of approaches to few-shot learning", *arXiv preprint arXiv:2203.04291*, 2022. DOI: 10.48550/arXiv.2203.04291.

[119] J. Pons, J. Serrà, and X. Serra, "Training neural audio classifiers with few data", in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 16–20. DOI: 10.1109/ICASSP.2019.8682591.

[120] B. Shi, M. Sun, K. C. Puvvada, C.-C. Kao, S. Matsoukas, and C. Wang, "Few-shot acoustic event detection via meta learning", in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 76–80. DOI: 10.1109/ICASSP40776.2020.9053336.

[121] S. Singh, H. L. Bear, and E. Benetos, "Prototypical networks for domain adaptation in acoustic scene classification", in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, pp. 346–350. DOI: 10.1109/ICASSP39728.2021.9414876.

[122] S.-Y. Chou, K.-H. Cheng, J.-S. R. Jang, and Y.-H. Yang, "Learning to match transient sound events using attentional similarity for few-shot sound recognition", in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 26–30. DOI: 10.1109/ICASSP.2019.8682558.

[123] K.-H. Cheng, S.-Y. Chou, and Y.-H. Yang, "Multi-label few-shot learning for sound event recognition", in *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, IEEE, 2019, pp. 1–5. DOI: 10.1109/MMSP.2019.8901732.

[124] K. Shimada, Y. Koyama, and A. Inoue, "Metric learning with background noise class for few-shot detection of rare sound events", in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 616–620. DOI: 10.1109/ICASSP40776.2020.9054712.

[125] Y. Wang, N. J. Bryan, M. Cartwright, J. P. Bello, and J. Salamon, "Few-shot continual learning for audio classification", in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, pp. 321–325. DOI: 10.1109/ICASSP39728.2021.9413584.

[126] H.-P. Huang, K. C. Puvvada, M. Sun, and C. Wang, "Unsupervised and Semi-Supervised Few-Shot Acoustic Event Classification", in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, pp. 331–335. DOI: 10.1109/ICASSP39728.2021.9414546.

[127] P. Wolters, C. Daw, B. Hutchinson, and L. Phillips, "Proposal-based few-shot sound event detection for speech and environmental sounds with perceivers", *arXiv preprint arXiv:2107.13616*, 2021. DOI: 10.48550/arXiv.2107.13616.

[128] Y. Wang, J. Salamon, N. J. Bryan, and J. P. Bello, "Few-shot sound event detection", in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 81–85. DOI: 10.1109/ICASSP40776.2020.9054708.

[129] A. Parnami and M. Lee, "Few-shot keyword spotting with prototypical networks", *arXiv preprint arXiv:2007.14463*, 2020. DOI: 10.1145/3529399.3529443.

[130] V. Morfi *et al.*, "Few-Shot Bioacoustic Event Detection: A New Task at the DCASE 2021 Challenge", in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021)*, 2021, pp. 145–149.

[131] Y. Koizumi, S. Murata, N. Harada, S. Saito, and H. Uematsu, "SNIPER: Few-shot learning for anomaly detection to minimize false-negative rate with ensured true-positive rate", in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 915–919. DOI: 10.1109/ICASSP.2019.8683667.

[132] P. Wolters, C. Careaga, B. Hutchinson, and L. Phillips, "A study of few-shot audio classification", *arXiv preprint arXiv:2012.01573*, 2020. DOI: 10.48550/arXiv.2012.01573.

[133] Y. Wang, J. Salamon, M. Cartwright, N. J. Bryan, and J. P. Bello, "Few-shot drum transcription in polyphonic music", *arXiv preprint*, 2020. DOI: 10.48550/arXiv.2008.02791.

[134] H. Wang, B. Wang, and Y. Li, "IAFNet: Few-shot learning for modulation recognition in underwater impulsive noise", *IEEE Communications Letters*, 2022. DOI: 10.1109/LCOMM.2022.3151790.

[135] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning", *Advances in neural information processing systems*, vol. 29, 2016. DOI: 10.48550/arXiv.1606.04080.

[136] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1199–1208. DOI: `10.1109/CVPR.2018.00131`.

[137] A. Köhn, F. Stegen, and T. Baumann, "Mining the spoken wikipedia for speech data and beyond", in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 2016, pp. 4644–4647.

[138] Society of Automotive Engineers, *SAE j3016 standard: Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles*, Available online: `https://www.sae.org/standards/content/j3016_202104/` (accessed on 28 February 2023), 2021.

[139] Y. Kartynnik, A. Ablavatski, I. Grishchenko, and M. Grundmann, "Real-time facial surface geometry from monocular video on mobile GPUs", *arXiv preprint arXiv:1907.06724*, 2019. DOI: `10.48550/arXiv.1907.06724`.

[140] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520. DOI: `10.1109/CVPR.2018.00474`.

[141] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database", in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255. DOI: `10.1109/CVPR.2009.5206848`.

[142] L. Shuangfeng, "Tensorflow lite: On-device machine learning framework", *Journal of Computer Research and Development*, vol. 57, no. 9, p. 1839, 2020.

[143] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies", *IEEE access*, vol. 8, pp. 58 443–58 469, 2020. DOI: `10.1109/ACCESS.2020.2983149`.

[144] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset", *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013. DOI: `10.1177/0278364913491297`.

[145] H. Caesar *et al.*, "nuscenes: A multimodal dataset for autonomous driving", in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631. DOI: `10.1109/CVPR42600.2020.01164`.

[146] P. Sun *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset", in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2446–2454. DOI: 10.1109/CVPR42600.2020.00252.

[147] J. Mao *et al.*, "One million scenes for autonomous driving: Once dataset", *arXiv preprint arXiv:2106.11037*, 2021. DOI: 10.48550/arXiv.2106.11037.

[148] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection", in *2016 24th European Signal Processing Conference (EUSIPCO)*, IEEE, 2016, pp. 1128–1132. DOI: 10.1109/EUSIPCO.2016.7760424.

[149] A. Mesaros *et al.*, "Sound event detection in the DCASE 2017 challenge", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 6, pp. 992–1006, 2019. DOI: 10.1109/TASLP.2019.2907016.

[150] P. Zinemanas, P. Cancela, and M. Rocamora, "MAVD: A dataset for sound event detection in urban environments", *Detection and Classification of Acoustic Scenes and Events, DCASE 2019, New York, NY, USA, 25–26 oct, page 263–267*, 2019. DOI: 10.33682/kfmf-zv94.

[151] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events", *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015. DOI: 10.1109/TMM.2015.2428998.

[152] A. Mesaros, T. Heittola, and T. Virtanen, "Acoustic scene classification: an overview of DCASE 2017 challenge entries", in *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, IEEE, 2018, pp. 411–415. DOI: 10.1109/IWAENC.2018.8521242.

[153] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification", *arXiv preprint arXiv:1807.09840*, 2018. DOI: 10.48550/arXiv.1807.09840.

[154] J. J. Bird, D. R. Faria, C. Premebida, A. Ekárt, and G. Vogiatzis, "Look and listen: A multi-modality late fusion approach to scene classification for autonomous machines", in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 10 380–10 385. DOI: 10.1109/IROS45743.2020.9341557.

[155] S. Wang, A. Mesaros, T. Heittola, and T. Virtanen, "A curated dataset of urban scenes for audio-visual scene analysis", in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, pp. 626–630. DOI: `10.1109/ICASSP39728.2021.9415085`.

[156] M. Fuentes *et al.*, "Urban sound & sight: Dataset and benchmark for audio-visual urban scene understanding", in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2022, pp. 141–145. DOI: `10.1109/ICASSP43922.2022.9747644`.

[157] E. R. Nascimento, R. Bajcsy, M. Gregor, I. Huang, I. Villegas, and G. Kurillo, "On the development of an acoustic-driven method to improve driver's comfort based on deep reinforcement learning", *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 5, pp. 2923–2932, 2020. DOI: `10.1109/TITS.2020.2977983`.

[158] Y. Weber and S. Kanarachos, "CUPAC–the Coventry University public road dataset for automated cars", *Data in brief*, vol. 28, p. 104 950, 2020. DOI: `10.1016/j.dib.2019.104950`.

[159] S. Tomar, "Converting video formats with FFmpeg", *Linux journal*, vol. 2006, no. 146, p. 10, 2006.

[160] G. Jocher *et al.*, *ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation*, version v7.0, Nov. 2022. DOI: `10.5281/zenodo.7347926`.

[161] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps", *IEEE Pervasive computing*, vol. 7, no. 4, pp. 12–18, 2008. DOI: `10.1109/MPRV.2008.80`.

[162] L. Dijkstra *et al.*, "Applying the degree of urbanisation to the globe: A new harmonised definition reveals a different picture of global urbanisation", *Journal of Urban Economics*, vol. 125, p. 103 312, 2021. DOI: `10.1016/j.jue.2020.103312`.

[163] J. Jakubik, M. Vössing, N. Kühl, J. Walk, and G. Satzger, "Data-centric Artificial Intelligence", *arXiv preprint*, 2022. DOI: `10.48550/arXiv.2212.11854`.

[164] R. Arandjelovic and A. Zisserman, "Look, listen and learn", in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 609–617. DOI: `10.1109/ICCV.2017.73`.

[165] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, "Look, listen, and learn more: Design choices for deep audio embeddings", in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 3852–3856. DOI: `10.1109/ICASSP.2019.8682475`.

[166] W. Liu, H. Wang, X. Shen, and I. W. Tsang, "The emerging trends of multi-label learning", *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 11, pp. 7955–7974, 2021. DOI: `10.1109/TPAMI.2021.3119334`.

[167] UNI, "Assessment of speech communication", Eng, UNI, Standard UNI EN ISO 9921, 2004, Available online: `https://store.uni.com/uni-en-iso-9921-2004` (accessed on 28 February 2023).

[168] R. L. Goldsworthy and J. E. Greenberg, "Analysis of speech-based speech transmission index methods with implications for nonlinear operations", *The Journal of the Acoustical Society of America*, vol. 116, no. 6, pp. 3679–3689, 2004. DOI: `10.1121/1.1804628`.

[169] IEC, "Sound system equipment - Part 16: Objective rating of speech intelligibility by speech transmission index", Eng, IEC, International Electrotechnical Commission IEC 60268-16, 2011, Available online: `https://webstore.iec.ch/publication/1214` (accessed on 28 February 2023).

[170] BS, "Sound system equipment. Objective rating of speech intelligibility by speech transmission index", Eng, BS, British Standards Document BS EN 60268-16, 2011. DOI: `10.3403/30249993U`.

[171] UNI, "Caratteristiche acustiche interne di ambienti confinati - Metodi di progettazione e tecniche di valutazione - Parte 1: Requisiti generali", ita, UNI, Standard UNI 11532-1, 2018, Available online: `https://store.uni.com/uni-11532-1-2018` (accessed on 28 February 2023).

[172] UNI, "Caratteristiche acustiche interne di ambienti confinati - Metodi di progettazione e tecniche di valutazione - Parte 2: Settore scolastico", ita, UNI, Standard UNI 11532-2, 2020, Available online: `https://store.uni.com/uni-11532-2-2020` (accessed on 28 February 2023).

[173] International standard ISO, "Acoustics - Measurement of room acoustic parameters - Part 2: Reverberation time in ordinary rooms", en, International Organization for Standardization, Standard ISO 3382-2, 2008, Available online: `https://www.iso.org/standard/36201.html` (accessed on 28 February 2023).

[174] R. Scheibler, E. Bezzam, and I. Dokmanić, "Pyroomacoustics: A python package for audio room simulation and array processing algorithms", in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 2018, pp. 351–355. DOI: `10.1109/ICASSP.2018.8461310`.

[175] J. B. Allen and D. A. Berkley, "Image method for efficiently simulating small-room acoustics", *The Journal of the Acoustical Society of America*, vol. 65, no. 4, pp. 943–950, 1979. DOI: `10.1121/1.382599`.

[176] D. Schröder, *Physically based real-time auralization of interactive virtual environments*. Logos Verlag Berlin GmbH, 2011, vol. 11.

[177] M. Vorländer, *Auralization*. Springer, 2020. DOI: `10.1007/978-3-030-51202-6`.

[178] J. S. Bradley, R Reich, and S. Norcross, "A just noticeable difference in C50 for speech", *Applied Acoustics*, vol. 58, no. 2, pp. 99–108, 1999. DOI: `10.1016/S0003-682X(98)00075-9`.

[179] R. H. Burroughs, *Piano*, US Patent 2,469,667, 1949.

[180] M. Muenster, F. Pfeifle, T. Weinrich, and M. Keil, "Nonlinearities and self-organization in the sound production of the Rhodes piano", *The Journal of the Acoustical Society of America*, vol. 136, no. 4, pp. 2164–2164, 2014. DOI: `10.1121/1.4899833`.

[181] F. Pfeifle and M. Münster, "Tone production of the Wurlitzer and Rhodes e-pianos", *Studies in Musical Acoustics and Psychoacoustics*, pp. 75–107, 2017. DOI: `10.1007/978-3-319-47292-8_3`.

[182] F. Pfeifle, "Real-time physical model of a Wurlitzer and Rhodes electric piano", in *Proceedings of the 20th International Conference on Digital Audio Effects (DAFx-17)*, 2017, pp. 17–24.

[183] S. Rothberg *et al.*, "An international review of laser Doppler vibrometry: Making light work of vibration measurement", *Optics and Lasers in Engineering*, vol. 99, pp. 11–22, 2017. DOI: `10.1016/j.optlaseng.2016.10.023`.

[184] M. Bosi and R. E. Goldberg, *Introduction to digital audio coding and standards*. Springer Science & Business Media, 2002, vol. 721. DOI: `10.1007/978-1-4615-0327-9`.

[185] E. Terhardt, G. Stoll, and M. Seewann, "Algorithm for extraction of pitch and pitch salience from complex tonal signals", *The Journal of the Acoustical Society of America*, vol. 71, no. 3, pp. 679–688, 1982. DOI: `10.1121/1.387544`.

[186] M. Le Brun, "Digital waveshaping synthesis", *Journal of the Audio Engineering Society*, vol. 27, no. 4, pp. 250–266, 1979.