Università degli Studi di Milano - Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di Laurea Magistrale in Data Science

Audio denoising in the wild

Relatore: Prof. Simone Bianco

Correlatore: Dott. Paolo Napoletano

Tesi di Laurea Magistrale di:
Fabrizio D'Intinosante
Matricola 838866

Anno Accademico 2019-2020

Ringraziamenti

Devo ancora decidere chi ringraziare.

Abstract

Technological innovation and the large-scale application of highly innovative tools have been offering great opportunities for study and development in the field of Artificial Intelligence for years. Among the numerous fields of application of these technologies are voice assistants with all the tasks associated with them. The very nature of these technological solutions means that their use often takes place in hostile, or highly noisy, environments such as urban contexts. From this problem arises the opportunity to investigate the potential of an end-to-end approach that includes a Deep Learning model to perform the denoising task in direct communication with a second model whose goal is to operate the speaker classification. This thesis work aims to verify this potential through a structured path, organized in numerous phases whose purpose is to obtain timely and comparable measurements with previous and subsequent works.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

A/D analog-to-digital. 7

AC Audio Classification. 2

ADC A/D converter. 7, 9

AI Artificial Intelligence. 1

ASR Automatic Speech Recognition. 1, 2, 23

BAK background noise distortion. 75, 77, 83

CNN Convolution Neural Network. 22, 28, 31–33, 60

D/A digital-to-analog. 8

DC Direct Current. 18

DCT Discrete Cosine Transform. 19

DL Deep Learning. 16, 22, 27, 57, 60

DNN Deep Neaural Network. 22

EER Equal Error Rate. 78–80, 83, 86, 87

FAR False Accept Rate. 78, 79

FNN Feedforwark Neural Network. 28–31, 33

FNR False Negative Rate. 79, 80

FPR False Positive Rate. 79, 80

FRR False Reject Rate. 78, 79

GAN Generative Adversarial Network. 36, 39, 41, 90

GAU Gated Activation Unit. 49, 67

HMM Hidden Markov Model. 23

HOG Histogram of gradients. 22

IoT Internet of Things. 2, 23

ISTFT Inverse Short-time Fourier Transform. 37

LSTM Long-Short Term Memory. 41, 49

MFC mel-frequency cepstrum. 19

MFCC Mel Frequency Cepstral Coefficient. 19, 39

ML Machine Learning. 16, 67, 70

MLP Multi-layer perceptron. 28

MSE Mean Squared Error. 71–73

NLP Natural Language Processing. 23

NN Neural Network. 22, 33, 34, 54, 56, 57, 66, 67, 71

NR Noise Reduction. 2

OVL overall quality. 75, 77, 83

PESQ Perceptual Evaluation of Speech Quality. 75–77, 83

RNN Recurrent Neural Network. 22, 28, 29, 50

ROC Receiver Operating Characteristic. 78, 79

SAP Self-Attemptive Pooling. 58, 60, 69

SDR Source-to-Distortion Ratio. 38

SE Speech Enhancement. 2, 24, 25

SEGAN Speech Enhancement Generative Adversarial Network. 39

SIFT Scale invariant feature transform. 22

SIG signal distortion. 75, 77, 83

SNR Signal-to-Noise ratio. 25, 61, 62, 77, 80, 86, 87

SR Speaker Recognition. 2, 24, 69

SSNR Segmental signal-to-noise ratio. 75, 77, 78, 83, 88

STFT Short-time Fourier Transform. 37, 38, 41

SV Speaker Verification. 24, 65, 69, 86, 88

TPR True Positive Rate. 79

TTS text-to-speech. 47, 51

VAD Voice Activity Detection. 61, 69, 77

Chapter 1

Introduction

Nowadays we are surrounded by voice recognition systems that allow us to talk to a computer or device that interprets what we are saying in order to respond to our questions or commands.

While less than twenty years ago such technologies were restricted to certain uses and the high cost made their mass use extremely prohibitive, this no longer applies today. The turning point occurred during the 2000-2010 decade when Google arrived with the launch of Google Voice Search. Due to its nature (it was an app), this made Automatic Speech Recognition (ASR) accessible to millions of people: but this was only the beginning. To have an example of this it is sufficient to use a simple smartphone or even a laptop. The technological development and the consequent increase in the computational capacity of the calculation systems have caused a strong acceleration in research and development for this sector and a significant reduction in production costs. In fact, it is very common today to interact with the voice assistants of our digital devices even for very simple operations such as asking to start playing a certain song from our favourite playlist or knowing what the weather forecast is for the next day. The fact that these ASR technologies have become increasingly cheaper to produce and more efficient to perform their task has spurred every tech company to develop its own voice assistant: some examples of this are Microsoft's Cortana, Apple's Siri or Amazon's Alexa [1], just to name a few of the most famous: it was the introduction of these digital assistants into the voice recognition market that changed the landscape of this technology in the 21st century. Moreover, considering the advancements in Artificial Intelligence (AI) and the increasing amounts of speech data that can be easily collected, it is realistic to think that voice will become the next dominant interface.

However, ASR is not exclusively limited to this type of application. Some of the most common and important uses for this technology are for example authentication, surveillance, security or even forensic investigations [2]. Important applications are

possible in the workplace, healthcare, banking and marketing, even the notorious Internet of Things (IoT) can not work at his best without using voice assistants: digital assistants application in cars such as control radio, listen to messages hands-free, assist with guidance and navigation are just few examples of the combination of ASR and IoT. Obviously, such complex technologies require particular attention to the weak points and the most critical aspects. For ASR systems and others audio-related tasks such as Speaker Recognition (SR) [3] or Audio Classification (AC) [4] for examples, there are many challenges which occur at the time of data acquisition (i.e. audio recording): one of the main challenges concerns environmental factors, in particular background noise [5]. This is a widely studied problem: ASR systems that work well in clean conditions suffer from a drastic loss of performance in the presence of noise [6].

1.1 Main Purposes

The primary objective of this thesis is to improve the performance of a SR system dealing with audio recorded in urban environments. This type of environments are notoriously characterized by the presence of noises of varying intensity coming from the most disparate sources e.g. conversations of other people, music, traffic etc.

The reason for this interest lies in the fact that, as mentioned above, voice assistants are now implemented in different kinds of devices, especially in mobile ones, which are therefore often used in the aforementioned environments. The path chosen in order to improve recognition performances does not concern the SR system itself, but rather the quality of the input audio: for this reason we can identify the main task of this thesis as Speech Enhancement (SE), in particular Noise Reduction (NR).

The final scope of this thesis is to build an end-to-end system that takes in input a noisy audio, performs a denoising operation and, hopefully, identifies the speaker with an accuracy higher than the same audio not subjected to NR: for this reason, the main focus is on the creation of a reliable and efficient environmental noise filtering system.

1.2 Thesis Outline

The purpose of this section is to briefly present the content of each of the next chapters:

- **Chapter 2** contains all the theoretical elements regarding the analysis of signals. These are necessary to understand the choices made during the thesis work including characteristics, extractable features and executable tasks;
- **Chapter 3** has the aim of providing further theoretical elements to the reader, but regarding what is the technical scope of the thesis work, that is Deep Learn-

ing. A brief overview is carried out on the most pertinent concepts, with a final study on the state of the art;

- **Chapter 4** represents the actual corpus of this work. In this chapter there is an explanation of all the work carried out, with an in-depth analysis of the implementation. Each choice is described and motivated in order to provide a logical path as structured as possible;
- **Chapter 5** is the chapter dedicated to the results and their analysis. Each measurement carried out as a conclusion of the work pipeline is presented and commented on with reference to the starting objectives of the thesis work;
- **Chapter 6** contains the conclusions of this work. This chapter contains the latest considerations, the limits of the approach and its possible developments.

Chapter 2

Audio signals processing concepts

With the aim to expose the work done for this thesis is of primary importance to introduce some fundamental concepts in order to understand the choices made during the development of the system. For this reason, the first topic will be the introduction to some basic concepts concerning audio signals. Subsequently, the main features that can be extracted from these signals will also be discussed. Finally, the main tasks concerning the audio signals' analysis, some of which already mentioned in Sec. 1, will be discussed in order to provide a wider overview and to better understand the main thesis task.

2.1 Foundations of digital signal processing

According to [7] sound can be defined as the result of vibration in the air. It is therefore possible to say that the sound wave is a longitudinal disturbance: this means that the air particles move back and forth from their equilibrium position in the direction the wave front is moving. Because this motion is difficult to represent the graphical depiction of sound is typically a two-dimensional graph of acoustic pressure vs. time as is possible to see in Fig. 2.1.

More generally, it is possible to define a *signal* as a physical quantity that varies with time, space or any other independent variable, as mentioned in [8]; for this reason a signal can be mathematically described as a function of one or more variables. It is also important to introduce some further definitions: a physical device that performs an operation on a signal is defined as a *system*, while the operation of passing a signal through a system is called *signal processing*.

There are two types of signals: discrete-time and continuous-time signals. Discrete-time signals are defined at the discrete moment of time and the mathematical function takes the discrete set of values.

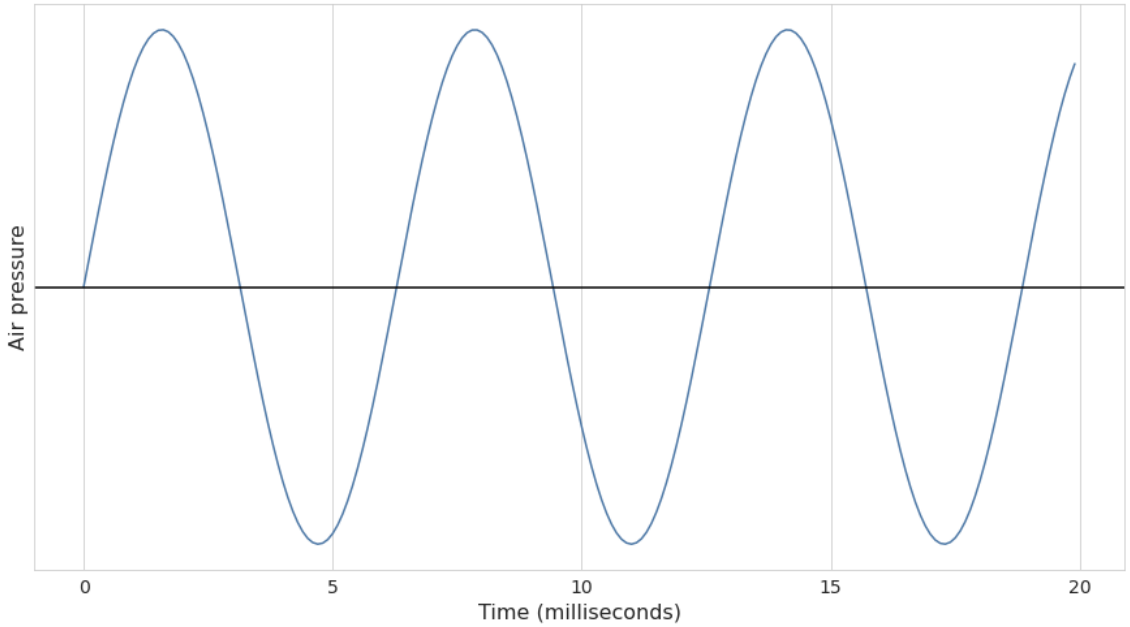

Figure 2.1: In this diagram, time is represented in the horizontal dimension and air pressure differences are represented in the vertical dimension. The straight (black) line down the middle represents the average background air pressure (the air pressure that would exist if there were no sound waves). Points above this line represent higher pressure (more crowded air molecules); points below represent lower pressure (less crowded air molecules).

2.1.1 Sound pressure level

As mentioned, sounds can be defined as vibrations that propagate in the air. The human ear, as well as the acquisition devices for audio signals, are sensitive to these vibrations, in particular to the pressure they exert on the membranes.

Generally, the unit of measurement of pressure (and therefore of sound intensity) is Pascal (Pa) but for acoustic signals it is very common to use the bel, in particular a customary representation: the decibel (dB). There are 10 dB in 1 bel, so a measurement in dB is 10 times the measurement expressed in bel [7]. The dB is a logarithmic way of describing a ratio between two homogeneous quantities. The ratio may be power, sound pressure, voltage or intensity or several other things.

Given two sounds with respective pressure levels p_1 and p_2, the ratio between these measured in decibels is given by:

$$20 \log \frac{p_1}{p_2} \text{ dB} \tag{2.1}$$

or equivalently:

$$10 \log \frac{p_1^2}{p_2^2} \text{ dB}$$

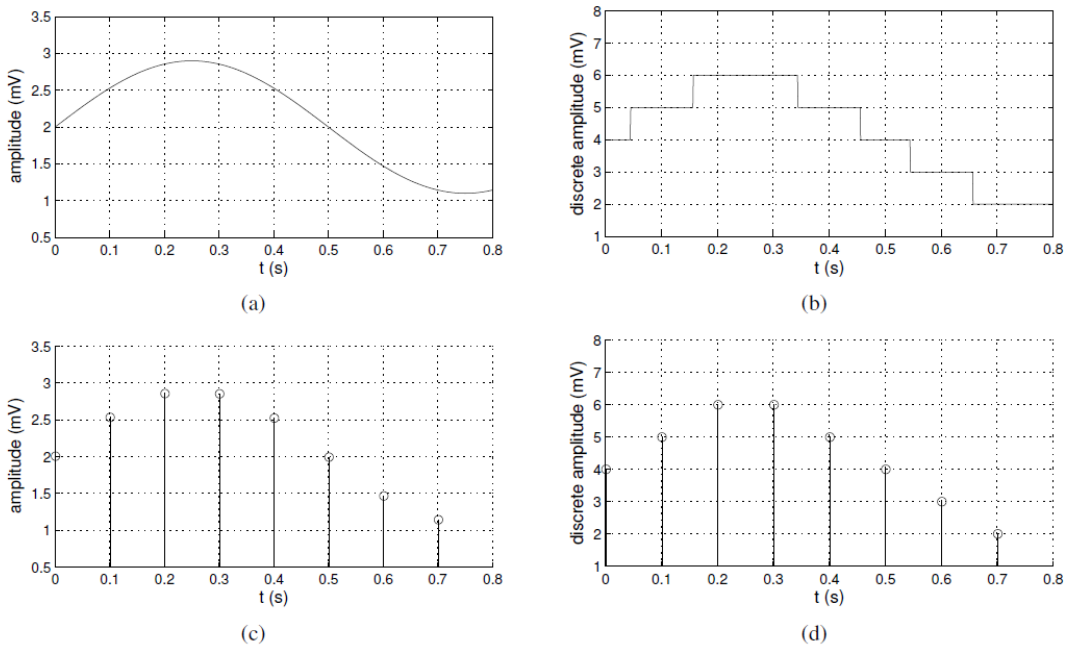

Figure 2.2: (a) Analog, (b) quantized-analog, (c) discrete-time, and (d) digital signals. From [9].

2.1.2 Classification of audio signals

As anticipated in Sec. 2.1 and mentioned in [9] a signal x can be defined as a function $x : \mathcal{D} \to \mathcal{C}$ from the domain $\mathcal{D}$ to the codomain $\mathcal{C}$. Depending on the nature of $\mathcal{D}$ and $\mathcal{C}$, a signal can be classified in four different ways:

1. $\mathcal{D} = \mathbb{R}$, $\mathcal{C} = \mathbb{R}$: analog signal;
2. $\mathcal{D} = \mathbb{R}$, $\mathcal{C} = \mathbb{Z}$: quantized analog signal;
3. $\mathcal{D} = \mathbb{Z}$, $\mathcal{C} = \mathbb{R}$: sampled signal;
4. $\mathcal{D} = \mathbb{Z}$, $\mathcal{C} = \mathbb{Z}$: digital signal, the type of signal that can be processed using a computer.

a graphical representation of this signals classification can be seen in Fig. 2.2.

With reference to the distinction mentioned in Sec. 2.1 a continuous-time signal $x(t)$ where $t \in \mathbb{R}$ is characterized by $\mathcal{D} = \mathbb{R}$ while for a discrete-time signal $x[n]$, where $n \in \mathbb{Z}$, $\mathcal{D} = \mathbb{Z}$. For the discrete-time signals n identifies the discrete time instants t_n. Very often the discrete-time signal x_d is obtained from a continuous-time signal x_c applying a periodically sampling:

$$x_d[n] = x_c(nT_s) \quad -\infty < n < \infty \tag{2.2}$$

with T_s as a fixed quantity called sampling period, measured in seconds (s). An alternative measurement methodology is the sampling rate, measured in hertz (Hz),

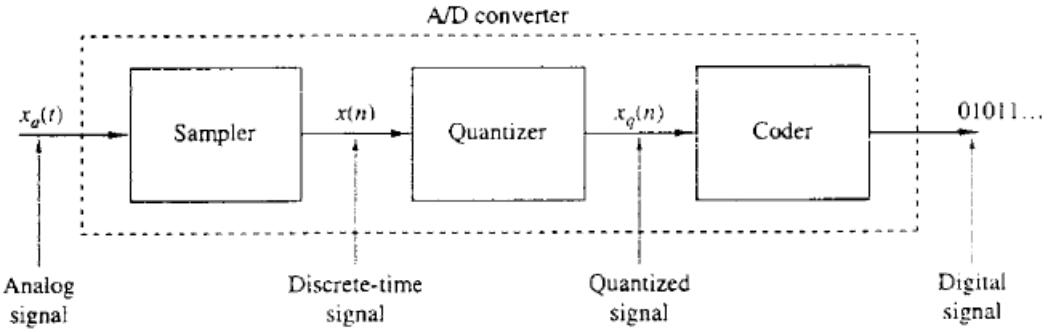

Figure 2.3: Basic parts of an analog-to-digital (A/D) converter. From [8].

calculated as $F_s = 1/T_s$. This value is the number of samples captured per second in order to represent the waveform. Higher sample rates allow higher audio frequencies to be represented.

Considering instead the codomain, when $\mathcal{C} = \mathbb{R}$ the signal is defined as continuous-amplitude, while with $\mathcal{C} = \mathbb{Z}$, as discrete-amplitude. From the possible combinations of these number sets for the domain and the codomain the four classes of signals listed above are realized. Since recorded audios were used for all the thesis work, the reference signal class is the *digital signal*, in particular a sampled signals with a sample rate of 16000 Hz; this means that every audio's second contain 16000 sampled points.

2.1.3 Analog-to-Digital and Digital-to-Analog conversion

As rightly mentioned in [8], signals are analog in nature. The fact that the signals are analog does not prevent them from being processed: these signals may be processed by appropriate analog system such as filters, frequency analyzers etc. with the aim, perhaps, to extract information.

There are, however, several reasons why an analog signal is converted into a digital format: first of all the digital programmable systems allows flexibility in re-configuring the operations simply by changing the program and provide much better control of accuracy requirements. Moreover the digital signals are easy to store on digital devices such as hard-disks, CD and USB sticks. Finally, more sophisticated algorithms can be applied in processing this type of signal and last but not least, very often the digital implementation of the signal processing system is cheaper than the analog counterpart.

At this point, after these premises have been made, it is possible to briefly explain how the process of converting a signal from analog to digital (and its opposite) takes place. The conversion procedure is called analog-to-digital (A/D) conversion, and the devices necessary to carry out this operation are called A/D converters (ADCs). The A/D conversion can be conceptually view as a three-step process, as clearly illustrated in Fig. 2.3. The first step consists in sampling the analog signal according to the modalities already mentioned in Sec. 2.1.2, in fact taking up what was previously said if

$x_c(t)$ is the input signal to the sampler, the output is $x_c(nT_s) \equiv x[n]$, with T_s sampling period. The second part of this pipeline is the quantization. This operation acts on the signal's codomain converting it from a continuous-valued to a discrete-valued. In this context the quantization error is the difference between the unquantized signal $x[n]$ and the quantized output $x_q[n]$. Finally, in the last step, defined as *coding*, each discrete value $x_q[n]$ is represented by a b-bit binary sequence.

It is often preferable to convert the processed digital signal back to analog, for this reason also this procedure is quickly explained. In accordance with his counterpart, this process is called digital-to-analog (D/A) conversion. In short, in order to obtain an analog signal starting from a digital one, it is necessary to "connect the dots" performing some kind of interpolation. The accuracy of this operation is directly related to the quality of the D/A conversion process. There are several interpolation techniques for this process such as linear interpolation between two consecutive samples or quadratic interpolation which consists in fitting a quadratic through three successive samples, just to name a few of them.

2.1.4 Sampling and quantization operations

As anticipated in Sec. 2.1.2 the fundamental operations that allow a signal to be converted from the analog to the digital format are sampling and quantization. While the former operation allows to convert the domain $\mathcal{D}$ from the number sets $\mathbb{R}$ to $\mathbb{Z}$, the latter concerns the conversion of the codomain $\mathcal{C}$ from $\mathbb{R}$ to $\mathbb{Z}$.

The principles of sampling have already been introduced in Sec. 2.1.2. What has not been discussed is the choice of the sampling step and the consequences that a wrong choice could cause. It is very important to determine an appropriate sampling frequency: using a very high sampling frequency means obtaining an excellent approximation of the analog signal but also producing very high volumes of data. A sampling rate that is too low, on the other hand, leads to undersampling phenomena, causing what is called *aliasing*. Indeed, without going into the details of the underlying mathematics, it is important to know that the highest frequency that can be uniquely distinguished when a signal is sampled at a rate $F_s = 1/T$ is equal to $F_N = F_s/2$ and is called *Nyquist frequency* [8]. Signals with frequency components above the Nyquist frequency appear aliased as graphically represented in Fig. 2.4. In an aliased signal, frequency components actually above the Nyquist frequency appear as frequency components lower than the latter. For example, a component at frequency $F_N < f_0 < F_s$ appears as the frequency $F_s - f_0$. This inaccurate frequency is called *alias*, hence the term aliasing.

The quantization operation has already been anticipated in Sec. 2.1.3. This process can be formerly defined as the conversion of a discrete-time continuous-amplitude

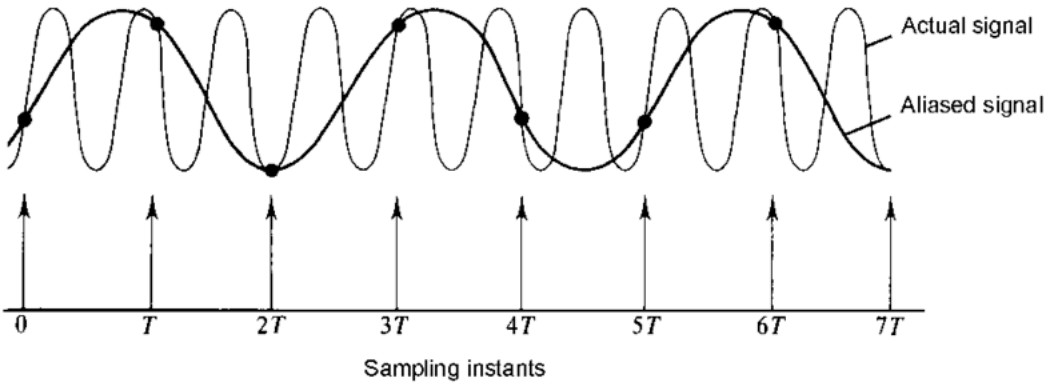

Figure 2.4: An example of aliasing in the time domain. The two signals have the same values at the sampling instants, although their frequencies are different. From [10].

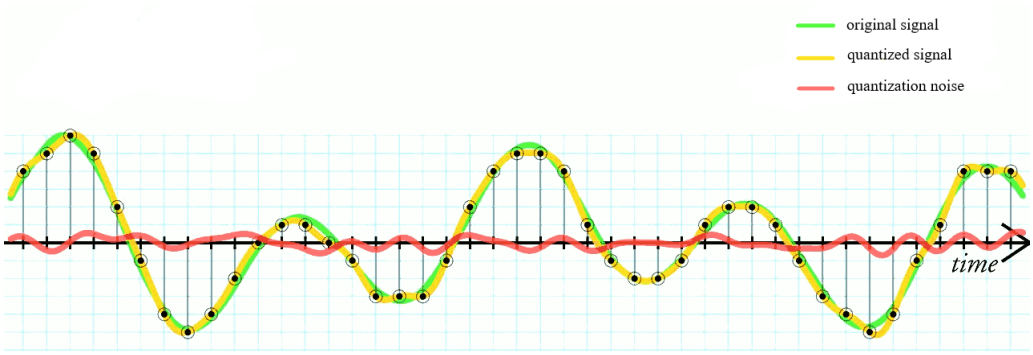

Figure 2.5: This example shows the original analog signal (green), the quantized signal (black dots), the signal reconstructed from the quantized signal (yellow) and the difference between the original signal and the reconstructed signal (red). The difference between the original signal and the reconstructed signal is the quantization error. From `https://en.wikipedia.org/wiki/Quantization_(signal_processing)`.

signal into a digital signal by expressing each sample value as a finite number of digits (instead of an infinite) [8]. The ADC samples the analog signal at uniform time intervals and assigns a digital value to each sample. The value is obtained by dividing the sampled analog input voltage by the voltage of reference and multiplying it by the number of digital codes available. The resolution of the converter (quantizer[1]) is given by the number of bits used. When a continuous random variable is converted to a discrete one or when a discrete random variable is converted to one with fewer levels this operation results in the creation of a quantization noise or quantization error [11]. An intuitive representation of how quantization noise originates is present in Fig. 2.5.

[1] A device or algorithmic function that performs quantization.

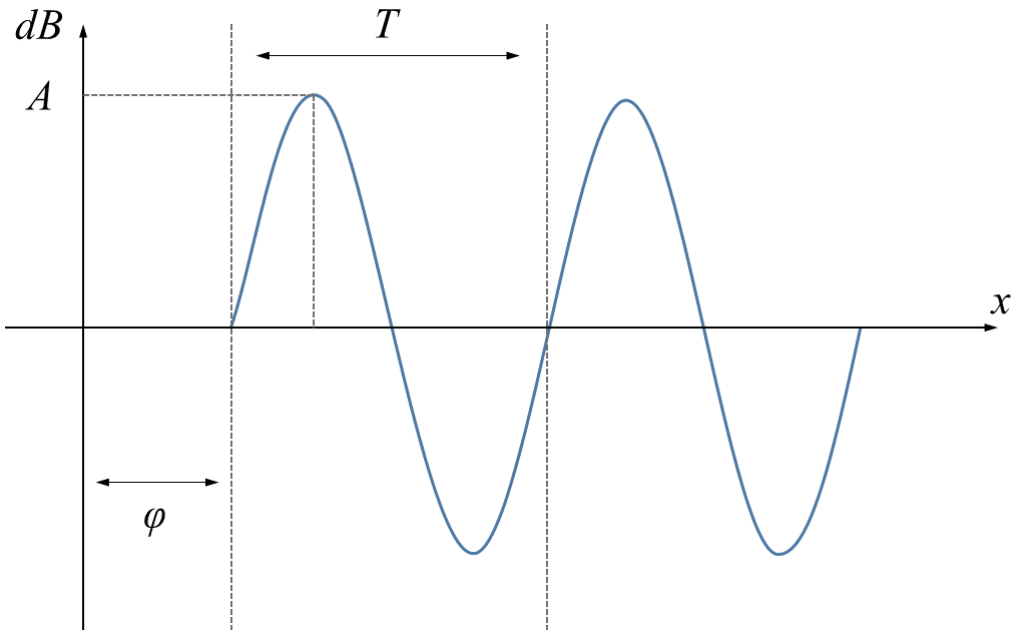

Figure 2.6: Representation of a sinusoidal function with his main features: Amplitude *(A)*, Phase *(φ)* and Frequency *(1/T)*.

2.1.5 Time-frequency domains and Fourier Theorem

Up to this point, speaking of signals as functions, in particular audio signals, reference has always been made to signals in the time-domain: nevertheless it is possible to represent an audio signal also in the frequency-domain. In fact the frequency-domanin representation of a signal is a convenient way of showing his oscillation rate but even more important is the fact that this view of the signal provides another way of analysing it, allowing to find valuable insights into a signal's behaviour [12]. Before briefly introducing the Fourier analysis, it is necessary to introduce the concept of *sinusoid*.

A sinusoid is a waveform that oscillates smoothly over time and is associated with many signals that occur in nature. This curve is mainly composed of three very important features: *amplitude*, *frequency* and *phase offset*. As shown in Fig. 2.6, the Amplitude specifies the maximum distance between the domain axis and the vertical position of the waveform while the Frequency indicates the number of oscillations (cycles) that occur each second of time. The Phase instead refers to the horizontal position of a waveform and specifies (in radians) where in the cycle the oscillation is at $t = 0$. Mathematically a sinusoid is often represented with two distinct equations:

$$\begin{aligned} y &= A\sin(\varpi x + \varphi) \\ y &= A\cos(\varpi x + \varphi) \end{aligned} \tag{2.3}$$

where ϖ refers to the *angular frequency* and is defined as $\varpi = 2\pi/T$.

Once these premises have been made, it is possible to introduce the Fourier analysis

[13]. The Fourier Theorem state that a **periodic** function $f(x)$ which is reasonably continuous may be expressed as the sum of a series of sine and cosine terms (called the *Fourier series*), each of which has specific Amplitude and Phase coefficients known as *Fourier coefficients*. The application of this theorem to sound is known as *Fourier analysis* (the signal's transformation from the time-domain to the frequency-domain called *Fourier transform*) while the opposite operation (the *inverse Fourier transform* which converts the signal from the frequency-domain to the time-domain) is called *Fourier synthesis*.

2.1.6 Fourier transform

The Fourier transform is a fundamental step in converting a signal from the time-domain to the frequency-domain. This conversion allows, as anticipated in Sec. 2.1.5, to obtain a lot of information on the signal that would remain inaccessible in the time-domain. Moreover a signal represented in the frequency-domain can be subjected to a series of operations in order to reduce or enhance certain aspects of that signal and then reconverted into the time-domain. This is the classic pipeline with filters: the defining feature of filters is the complete or partial suppression of some aspect of the signal.

As mentioned in [14], the term *Fourier transform* can be broken into four categories according to the type of signal being analyzed: in fact a signal can be continuous or discrete, periodic or aperiodic. Combining these signal's characteristics a distinction can be made between:

- **Fourier transform** when the signal is aperiodic and continuous;
- **Fourier series** when the signal is continuous but periodic;
- **discrete time Fourier transform** if the signal is aperiodic and discrete;
- **discrete Fourier transform** if the discrete signal is also periodic.

a graphic representation of these types of signals can be seen in Fig. 2.7.

The Fourier series therefore provides that under certain conditions a function can be represented as the sum of harmonics functions, each with own frequency, amplitude and phase. The mathematical formulation for the Fourier series is as follows:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos(\frac{2\pi}{T} kx) + b_k \sin(\frac{2\pi}{T} kx) \tag{2.4}$$

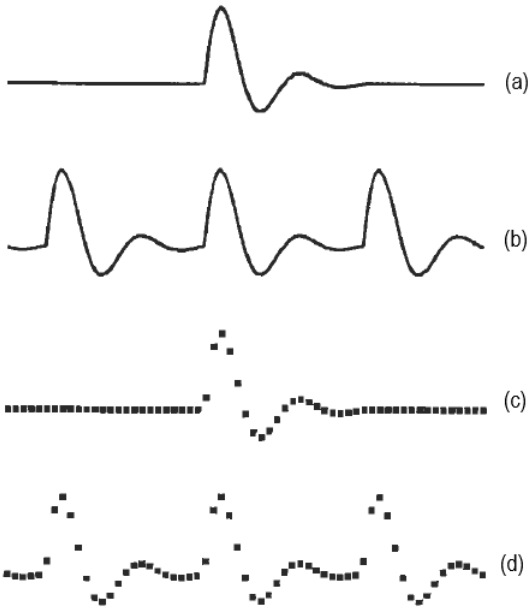

Figure 2.7: (a) Aperiodic-Continuous, (b) Periodic-Continuous, (c) Aperiodic-Discrete, (d) Periodic-Discrete. From [14].

with the uniquely determined coefficients:

$$\begin{aligned} a_k &= \frac{2}{N}\int_{-\frac{N}{2}}^{\frac{N}{2}} f(x)\cos(\frac{2\pi}{N}kx)\ dx \\ b_k &= \frac{2}{N}\int_{-\frac{N}{2}}^{\frac{N}{2}} f(x)\sin(\frac{2\pi}{N}kx)\ dx \end{aligned} \tag{2.5}$$

A well-known example of a Fourier series is the approximation of a square wave given by the sum of the fundamental frequency F_0 (200 Hz) with the harmonics whose frequency is multiple with respect to the fundamental but only for odd numbers, for example $3F_0$, $5F_0$ and so on: Fig. 2.8 shows an example of this approximation.

Anyway, as mentioned earlier it is possible to apply Fourier transform also to continuous and aperiodic signals and to discrete signals whether they are periodic or aperiodic. In fact every continuous function, even if not periodic, can be expressed as an integral of weighted complex sinusoids. Mathematically, the Fourier transform for aperiodic continuous signals takes the following form:

$$F(u) = \int_{-\infty}^{\infty} f(x)e^{-j2\pi ux}\ dx \tag{2.6}$$

As introduced in Sec. 2.1.5, from the Fourier transform (analysis) it is possible to reconstruct the original function, without loss of information, through the inverse

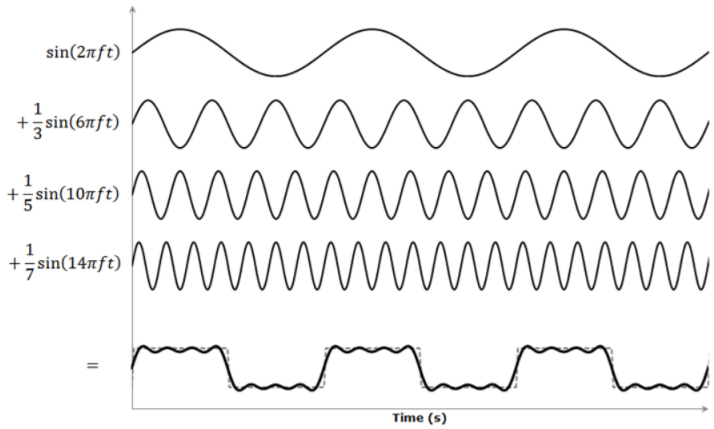

Figure 2.8: Example of square wave approximation by summing sinusoids of different amplitude, frequency and phase. From `http://www.thepulsar.be/article/generating-sine-wave-from-square-waves/`.

process (synthesis), formalized as:

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{j2\pi ux}\, du \tag{2.7}$$

Mathematically, the realizations of the Fourier transform and the inverse transform for discrete signals are very similar to those seen for continuous signals, except that for the former the area under the curve is not taken into considerations, but rather the sum of the point values. For the transform operation the following equation is used:

$$F(u) = \frac{1}{N} \sum_{i=0}^{N-1} f(i) e^{-j2\pi u \frac{1}{N} i} \tag{2.8}$$

while the inverse transform is equal to:

$$f(x) = \sum_{k=0}^{N-1} F(u) e^{j2\pi kn \frac{1}{N}} \tag{2.9}$$

To summarize therefore, the Fourier transform, for any type of signal, allows to convert the signal from the original time-domain to the frequency-domain, thus facilitating operations on the signal itself, while the inverse transform allows to bring back the processed signal to the domain of time. A summary scheme of the procedure is represented graphically in Fig. 2.9: the exposed pipeline is represented by the path (a)-(b)-(c)-(d). However, as is possible to see, an alternative path is also available, which passes directly from point (a) to point (d): the convolution.

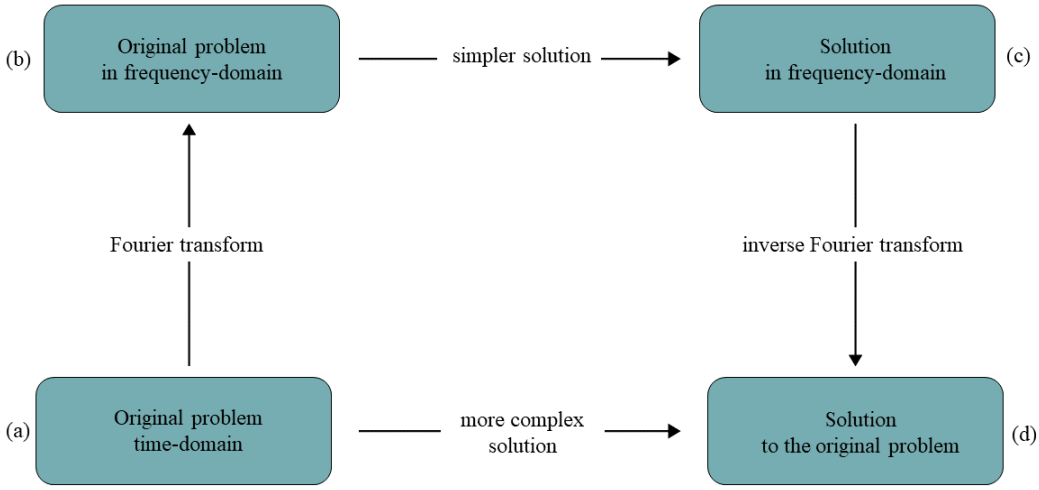

Figure 2.9: A schematic representation of the pipeline involving the use of the Fourier transform and its inverse.

2.1.7 Convolutions and convolution theorem

Convolution is a mathematical way of combining two signals (f and g) to form a third signal ($f * g$) expressing how the shape of one is modified by the other [14]. The reverse operation with respect to convolution is called deconvolution. Convolutions apply indifferently to both continuous and discrete signals, and they can be considered as a formal mathematical operation, just like multiplications, additions and integration.

In the case of continuous signals the convolution operation can be mathematically formalized as:

$$f * g = \int_{s=-\infty}^{\infty} f(s)g(x-s)\,ds \tag{2.10}$$

$$f * g = g * f$$

while for discrete signals:

$$f * g[n] = \sum_{m=-\infty}^{\infty} f(m)g(n-m) \tag{2.11}$$

Visually, considering the continuous case, the convolution process can be explained using Fig. 2.10:

- one of the functions is reflected $g(t) \rightarrow g(-t)$;
- the inverted signal is translated from $-\infty$ to ∞;
- at each translation the product between the two signals is calculated;
- finally, the product area is calculated.

After having briefly introduced the convolutions it is possible to state their theorem.

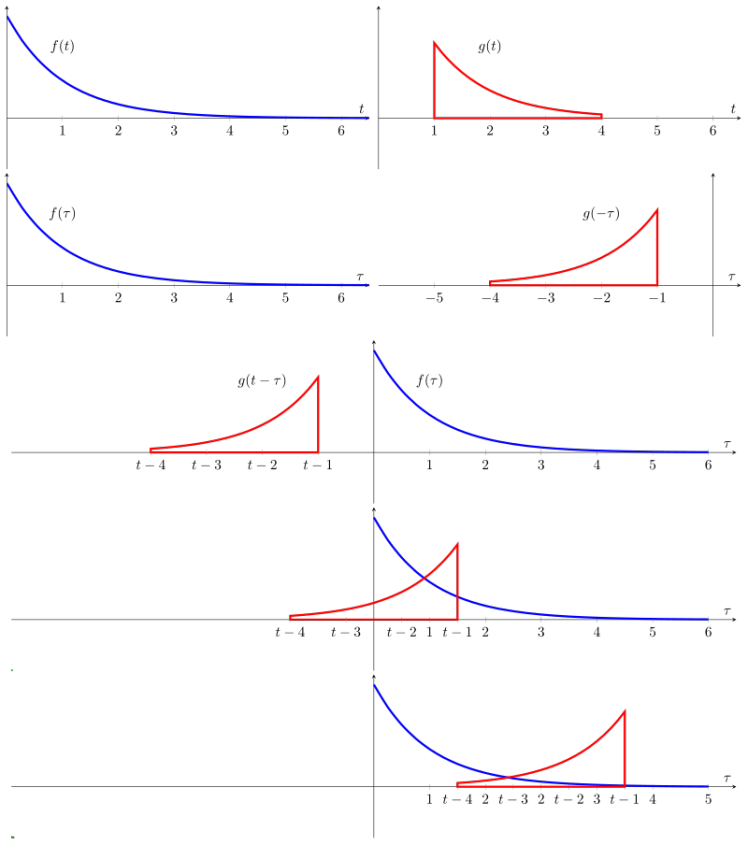

Figure 2.10: Visual explanation of convolution in the case of continuous functions. From `https://en.wikipedia.org/wiki/Convolution`.

Given:

$$g(x) = f * h = \int_{s=-\infty}^{\infty} f(x-s)h(s)\ ds$$

by applying the definition of Fourier transform it is possible to prove the convolution theorem, i.e. the transform of the convolution of two functions is the product of the transforms of the two functions:

$$G(u) = F[g(x)] = F[f(x) * h(x)] = F(u)H(u) \tag{2.12}$$

Therefore, with reference to Fig. 2.9, the convolution, intended as the application to a function f of a suitable function h called *kernel filter*, represents path (a)-(b) and allows to obtain the same results as path (a)-(b)-(c)-(d) but involving fewer operations.

2.2 Main features of audio signals

The feature extraction phase represents a very important step in the audio signal preparation pipeline to perform any task. As anticipated, a digital signal, depending on the sample rate used for sampling from the analog counterpart, can be very heavy computationally. For example, a single-channel audio of ten seconds, with a sample rate of 16000 Hz is conceivable as a 1-D array made up of 160000 samples.

With the aim of carrying out any task among those possible concerning audio signals (these will be discussed in detail later) using Machine Learning (ML) or Deep Learning (DL) models, feature extraction is an important step in increasing the efficiency of such models. However, very often feature extraction can also represent several disadvantages, such as the dilation of the calculation times or the impossibility of returning to the original signal. For this reason, for certain tasks, there are also models capable of dealing directly with audio signals: in this thesis, for example, one of these approaches will be presented. Nevertheless, the features that can be obtained from audio signals represent a fundamental topic that it is not possible to avoid mentioning.

As mentioned in [15], humans can ear in the frequency range between 20 Hz and 20 kHz. Intesity, which is measured in dB is another factor of sound: the lowest sound that a human can ear is of 0 dB while over 120 dB pain threshold is reached. As represented in Fig. 2.11 the audible human range can be used to distinguish between different types of sound. First of all between audible and inaudible and then, in the audible, between speech, music, natural and artificial sounds and noise. As already said, the focus of this thesis concerns speech and noise, and it is on the features most used to represent the first that we will try to give a brief perspective.

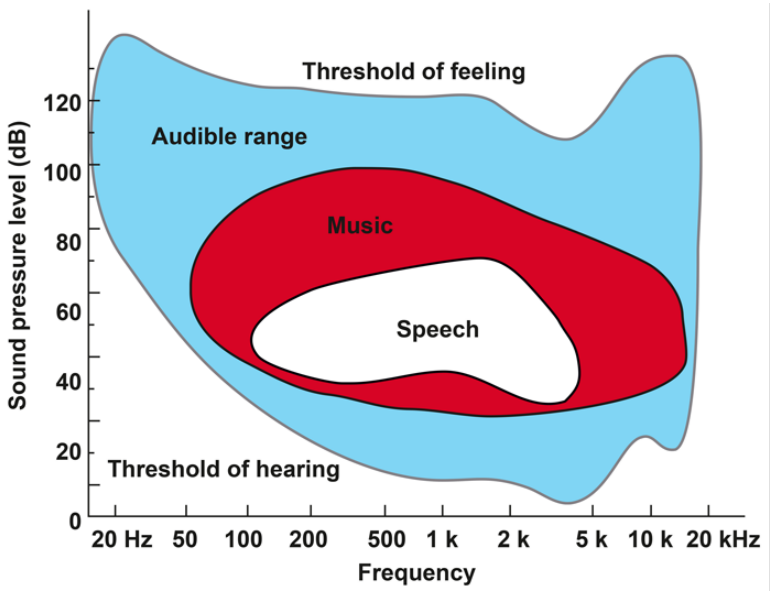

Figure 2.11: Depiction of the human hearing range. From `https://www.entandaudiologynews.com/features/ent-features/post/optimising-hearing-aid-processing-for-music-appreciation`.

2.2.1 Audio features classification

There is no strict classification regarding the derivable features from audio signals. However, it is possible to try to organize them according to an arbitrary classification. In this regard, one could distinguish between classic and advanced features, or they could be classified as physical or perceptual as mentioned in [15]. Other possible distinctions are present in [16], where a distinction is made between low-level features and others, or in [17] in which the features are classified in more detail according to the signal domain or the characteristic of the signal from which they are extracted. In particular the classification operated in [17] takes into account the historical evolution of the audio features as seen in Fig. 2.12. For reasons of conciseness, only some of the most used features will be presented, using a mixed classification among those mentioned.

2.2.2 Classical or low-level features

The features presented here all belong to the set of physical features i.e. they can directly be measured from frequency or time representation of audio signal and computed directly from the amplitude values or other spectral values of signal, as classified in [15]. These features, in fact, are considered as low-level or classical because they are well known and represented, in the past, the most used features for the analysis of audio signals. However, although very advanced features have been created over time, depending on the task of interest, these classic features can still prove to be extremely

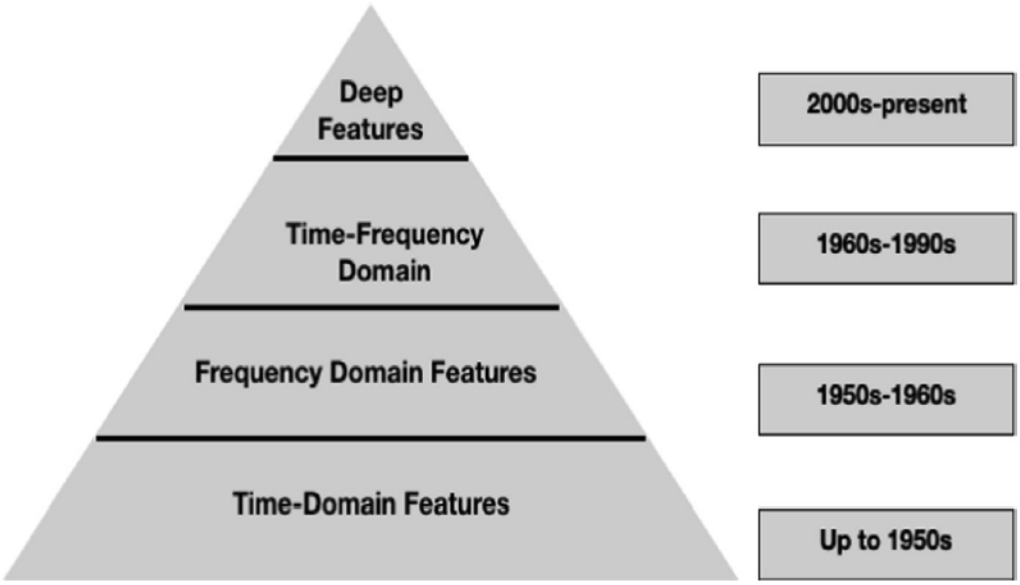

Figure 2.12: Evolution of audio features. From [17].

useful.

1. **Shot-time Energy**: provides the representation of variations of amplitude over the time.
$$STE = \frac{1}{T-1}\sum_{t=1}^{T-1} S_t^2 \tag{2.13}$$

2. **Average**: simply the average value assumed by the signal. In electronics the average value is also called Direct Current (DC).
$$\mu = \frac{1}{N}\sum_{i=0}^{N-1} x_i \tag{2.14}$$

3. **Variance**: just like the mean, variance is one of the statistical measures characterizing a signal and represents the power of the signal fluctuation around the mean.
$$\sigma^2 = \frac{1}{N-1}\sum_{i=0}^{N-1}(x_i - \mu)^2 \tag{2.15}$$

4. **Zero Crossing Rate**: the rate at which the signal changes from positive to zero to negative or from negative to zero to positive.
$$ZCR = \frac{1}{T-1}\sum_{t=1}^{T-1} 1_{\mathbb{R}<0}(s_t s_t - 1) \tag{2.16}$$

5. **Spectral Centroid**: indicates where the center of mass of the spectrum is

located. It measures the "brightness" of the signal.

$$Centroid = \frac{\sum_{n=0}^{N-1} f(n)x(n)}{\sum_{n=0}^{N-1} x(n)} \tag{2.17}$$

6. **Fundamental Frequency**: it is measured using the periodicity of the signal in the time-domain.

2.2.3 Advanced features

Referring to [17], it is possible to find a very long list of features, some more advanced than others. Some of them that are believed to be the most common are set out below. Among these there are some in the frequency-domain, in the time-frequency domain and finally deep features. The features extracted directly from the time-domain can be considered the most classic and least advanced.

Frequency-domain features

Among the extractable features from signals represented in the frequency-domain, Mel Frequency Cepstral Coefficients (MFCCs) is probably the best known and used. A graphic representation of this feature is visible in Fig. 2.13. MFCCs are derived from the cepstral representation of an audio clip. A cepstrum is obtained by taking the inverse Fourier transform of the logarithm of the spectrum of the signal. The difference between the cepstrum and the mel-frequency cepstrum (MFC) is that in the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum. The procedure for extracting MFCCs from a signal is summarized in a few steps:

1. frame the signal into short frames using the windowing[2] technique [18];
2. for each frame calculate the periodogram[3] estimate of the power spectrum [19];
3. apply the mel filterbank to the power spectra, sum the energy in each filter;
4. take the logarithm of all filterbank energies;
5. take the Discrete Cosine Transform (DCT)[4] [20] of the log filterbank energies.

[2] In real time audio signals are non-stationary over the time. To analyze such non-stationary signals windowing technique is employed and the long non-stationary signal is analyzed as short chunks of quasi-stationary signal.

[3] In signal processing, a periodogram is an estimate of the spectral density of a signal.

[4] DCT expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies.

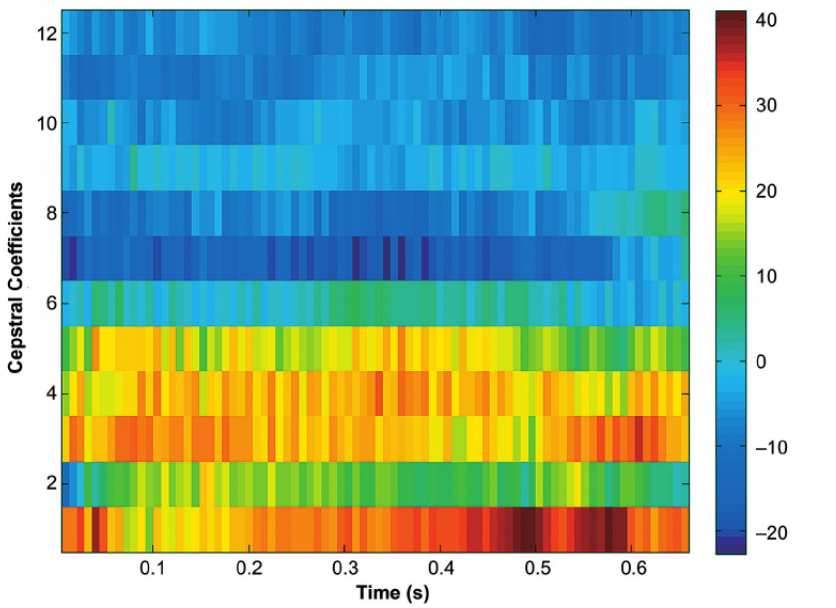

Figure 2.13: Mel-frequency cepstral coefficients (MFCCs) example. From [21].

Time-frequency domain features

A time-frequency transform of a signal is the way of looking into the signal having time on one axis and frequency on another. An example of such a transformation is the spectrogram. A spectrogram displays signal strength over time at the various frequencies present in a waveform: a visual representation is available in Fig. 2.14. The advantage of using the time-frequency domain is that it essentially transforms an audio signal into an image, which at this point allows you to extract typical features of computer vision. An alternative version of the spectrogram, widely used in the analysis of audio signals is the Mel spectrogram. A Mel spectrogram is a spectrogram where the frequencies are converted to the mel scale. The Mel scale is often used because humans do not perceive frequencies on a linear scale. In fact, humans usually do not perceive frequencies on a linear scale. The human ear is more able to detect differences at low frequencies (for example between 500 and 1000 Hz) while it is much more difficult to distinguish between higher frequencies such as between 10000 and 10500 Hz. The principle of the Mel spectrogram is essentially to map frequencies to the Mel scale where a unit of pitch such that equal distances in pitch sounded equally distant to the listener. Graphically, the conversion between the hertz scale and that of Mel appears as in Fig. 2.15 while mathematically a well known formula for converting f hertz to m mels looks like

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) = 1127 \ln\left(1 + \frac{f}{700}\right) \tag{2.18}$$

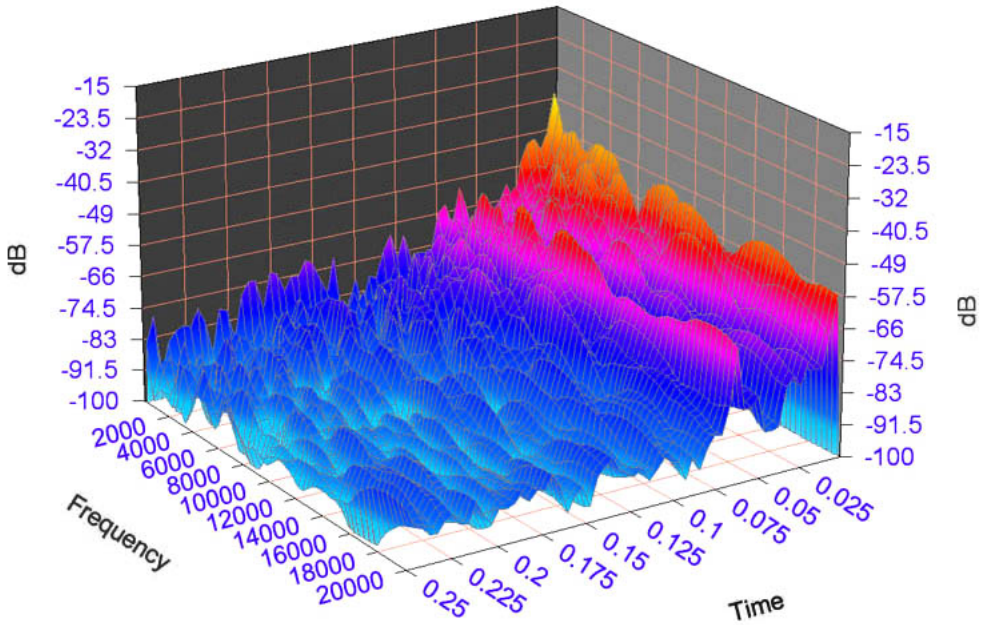

Figure 2.14: Example of 3-D spectrogram representation. From `https://www.yousciences.it/ingegneria/ingsound/misuratori/sonogramma.php`.

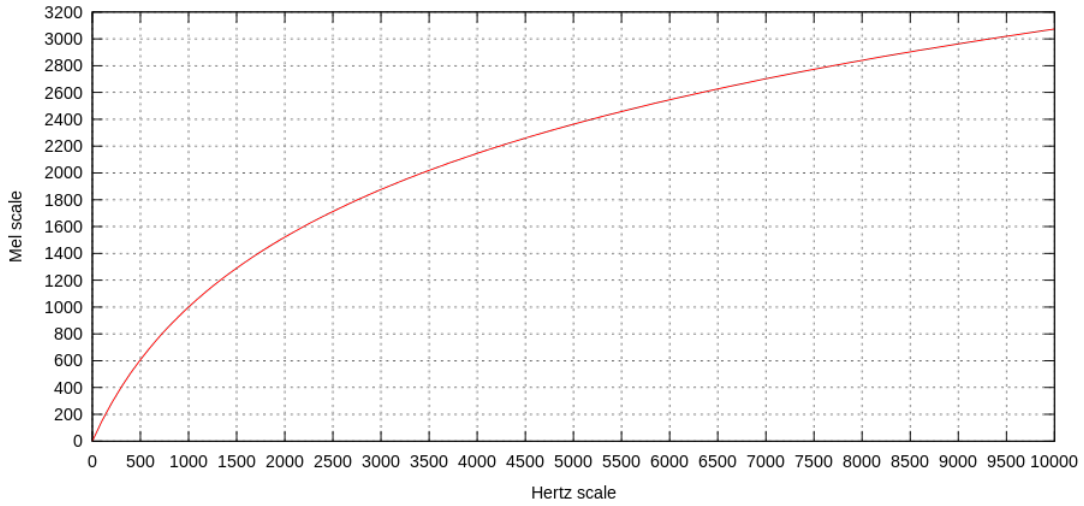

Figure 2.15: Plots of pitch Mel scale versus hertz scale. From `https://www.wikiwand.com/en/Mel_scale`.

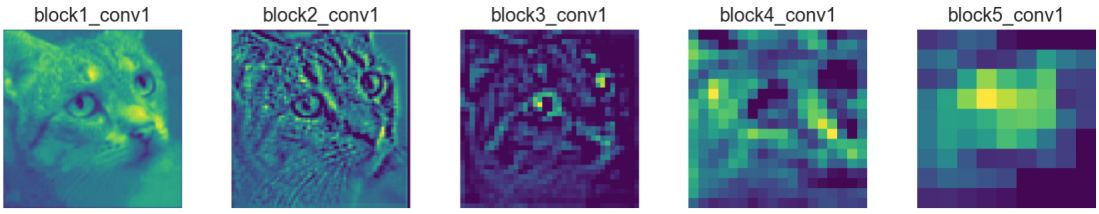

Figure 2.16: An example of deep features extracted from a CNN from an image representing a cat. From `https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2`.

Histogram of gradients (HOG) [22] and Scale invariant feature transform (SIFT) [23] are common features used both in computer vision and in the analysis of spectrograms obtained from audio signals. While the former consists in counting occurrences of gradient orientation in localized portions of an image with the aim to perform object detection and it is useful in extracting the time-frequency information from the spectrograms, the latter is employed on spectrograms of audio signals to detect the local information.

Deep features

The term deep features [24, 25] refers to all the features obtainable from audio signals using models such as Neural Networks (NNs). DL has been proven to be a powerful technique to extract high level features from low level information. The deep features could be extracted from any DL model like Convolution Neural Networks (CNNs), Deep Neaural Networks (DNNs), Recurrent Neural Networks (RNNs) and others. Without going into detail, NN are composed of different layers that extract features from the input signal, regardless it is one-dimensional or two-dimensional (audios, time series or images). Hence, the significance of the deep features highly depends on the layer from which these are extracted. An intuitive example regarding deep features is shown in Fig. 2.16.

2.3 Audio related tasks

The application area of audio signals is very wide. Starting from an audio signal it is in fact possible to perform many tasks: some of these have already been mentioned in Sec. 1. To reduce the scope of the topic, we will focus only on speech signals. Many operations are made possible thanks to the tasks that can be carried out using speech signals, some of which are in common use and are now part of the daily life of each of us.

The purpose of this section is to briefly introduce the most popular tasks concerning speech signals.

2.3.1 Speech recognition

Speech recognition or ASR is the capability of an electronic device to recognize and translate spoken language into text. Speech recognition is often regarded as the front-end for many Natural Language Processing (NLP)[5] components [26]. The speech recognition task is basically divided into two categories: the former is the "speech-to-text", that is the possibility that a computer can report in written form what is said, while the latter is the "text-to-speech", which consists in synthesizing the reading of a written text that is passed to the computer. Most of the modern ASR systems are typically based on statistic models e.g. Hidden Markov Models (HMMs)[6] [27].

ASR has many applications, some of which have already been mentioned in Sec. 1. Some of the most significant uses concern the IoT field: in car-systems simple voice commands may be used to initiate phone calls, select radio stations or play music from a compatible smartphone or, for more advanced systems, home automation represents a very interesting development frontier. All voice assistants such as Apple's Siri, Microsoft Cortana and Amazon Alexa use speech recognition to understand the orders that are given by users. The applications for this particular task are many and potentially applicable in any field. Further examples are transcription systems, hands-free human-machine interaction or all the functionalities to support people with disabilities.

2.3.2 Speaker recognition

Speaker recognition is a multi-disciplinary branch of biometrics that may be used for identification, verification, and classification of individual speakers [28]. Recognizing the speaker can simplify the task of translating speech in systems that have been trained on specific voices or it can be used to authenticate or verify the identity of a speaker as part of a security process.

This task can be divided into two sub-categories: closed-set and open-set setting. While the former is characterized by the fact that all the identities to be verified are contained in the data set used to train the model responsible for the identification reducing the problem to a classification problem, the latter is slightly more complicated. In fact, for open-set setting, the testing identities are not provided to the model during the training phase: this is more close to the practice.

The task can also be distinguished between text-dependent and text-independent: the former consists in the fact that the recognition can be carried out only if a certain

[5] NLP is broadly defined as the automatic manipulation of natural language, like speech and text, by software.

[6] HMM is a statistical Markov model in which the system being modeled is assumed to be a Markov process X with not observable ("hidden") states. HMM assumes that there is another process Y whose behavior "depends" on X. The goal is to learn about X by observing Y.

sentence, on which the model has been trained, is pronounced, while for the latter the model is able to recognize the speaker regardless of the sentence pronounced [29].

With reference to the closed-set type, this is what is commonly called speaker recognition, while the open-set is called Speaker Verification (SV). While in SR the model is trained on a finite set of identities and can only recognize people whose voice data it already knows, SV is more like practice e.g. given a database containing vocal tracks belonging to different people, the speaker verification task consists in verifying whether the identity of the input audio is the same as one of the audio in the database.

There are many possible applications for this technology and these are continually growing. Among the many possible it is worth mentioning access control which consists in adding biometrics factors to usual passwords and tokens or transaction authentication such as mobile banking and purchases. In addition to security, other sectors of application concern the personalization of mobile devices and the optimization of applications that use audio [30].

2.3.3 Blind source separation

Separation of sources consists of recovering a set of signals of which only instantaneous linear mixtures are observed. It is most commonly applied in digital signal processing and involves the analysis of mixtures of signals; the objective is to recover the original component signals from a mixture signal [31].

The most common application for speech is the isolation of the voice of a single person in a scenario where there are many people speaking at the same time.

2.3.4 Speech Enhancement

The speech enhancement task is the main focus of this thesis. In fact, as mentioned in Sec. 1.1, the objective of this thesis is to explore the possibility of improving the performance of a speaker recognition model in urban conditions by acting on the input audio to filter background noise.

In fact, the goal of the SE task is to improve some perceptual aspect of speech that has been degraded by additive noise. The need to enhance speech signals arises in many situations in which the speech signal originates from a noisy location e.g. urban environments, or is affected by noise over a communication channel such as telephone signal interference. Ideally, the goal of SE algorithms would be to improve both quality and intelligibility. In practice, however, it is difficult to achieve this goal. In fact, there are many techniques that allow to reduce the background noise, but at the expense of introducing speech distortion i.e. artifacts, which in turn may impair speech intelligibility. Hence, the main challenge in designing effective SE algorithms is to suppress noise without introducing any perceptible distortion in the signal [32].

These distortions can cause the loss of important information regarding the speaker's voice timbre, making the speaker recognition process more complex.

There are several techniques for performing speech enhancement. The best known consist in the use of some filters, others are based on the spectral analysis of the audio track, while still others consist in the use of models.

There are many applications for this task, as it is a generic task that can potentially be chained to all the other tasks concerning audio signals. The presence of noise in audio signals is in fact a problem that worsens the performance of any task concerning speech or music. Among the most well-known applications there are those concerning mobile devices where SE aims to improve the speech recognition performance of the device.

Overview on noise signals

Noise can be stationary, that is, does not change over time, such as the fan noise coming from PCs. Noise can also be non-stationary, such as the restaurant noise, that is, multiple people speaking in the background mixed in some cases with noise emanating from the kitchen or other types of urban noises such as in the street (e.g., cars passing by, street construction work), in the car (e.g., engine noise, wind). As these examples illustrate, noise appears in different shapes and forms in daily life. Obviously, the task of suppressing or reducing noise that is constantly changing (non-stationary) is more difficult than the task of suppressing stationary noise [32].

The case represented in this thesis uses non-stationary noises. For the speech enhancement task it is very important that the algorithm is able to identify and reduce the noises present at intensities compatible with those encountered in everyday reality. A very useful measure of noise intensity is the Signal-to-Noise ratio (SNR). Generally SNR is defined as the ratio between the power of a signal and the power of the background noise and is measured in dB. In this way, this measurement gives an indication of how strong the signal of interest is compared to the background noise.

$$SNR = 20 \log \left(\frac{\text{Speech signal}}{\text{Background noise}} \right) \text{ dB} \tag{2.19}$$

2.3.5 Pathological speech detection

There are many pathological conditions that can affect the voice. Many of these conditions have their origins primarily in the vocal system and many tools available for detection of speech pathologies are invasive or require expert analysis of numerous human speech signal parameters. So, a reliable, accurate and non-invasive automatic system for recognizing and monitoring speech abnormalities is one of the necessary tools in pathological speech assessment [33].

One of these pathological conditions for which the analysis of audio signals can be useful for diagnosis is dysarthria [34].

Chapter 3

Theoretical Overview

The purpose of this chapter is to provide additional theoretical elements with respect to those presented in the previous chapter regarding the methods adopted during the thesis work. In particular, the central element will be Deep Learning, with its definition, a brief historical introduction and the theoretical elements characterizing it. Subsequently, the state of the art regarding the reduction (removal) of noise from audio signals including a brief mention to the main approach of the thesis work, will be presented.

3.1 Elements of Deep Learning theory

When it comes to Deep Learning, we must consider that we are not talking about an independent discipline. Deep Learning can rather be defined as a set of Machine Learning algorithms that uses a series of layers to progressively extract high-level features from the data [35]. In turn, Machine Learning is defined as the study of algorithms that allow automation through experience [36] thus resulting, by definition, a subset of Artificial Intelligence. This hierarchy is represented graphically in Fig. 3.1.

However, to fully understand the reasons behind the popularity of Deep Learning, it is important to make a small historical introduction before moving on to the explanation of some of the elements characterizing DL and neural networks.

3.1.1 The evolution of Deep Learning

The history of Deep Learning can be essentially divided into 3 main phases called "waves", separated by two periods of quiescience, called "winters".

The first developments in the sector took place during the first wave, that is between 1940 and 1960: it is in this period that the *artificial neuron* (or *perceptron*) [37] was born, i.e. the attempt to imitate the behavior of the biological neuron in the

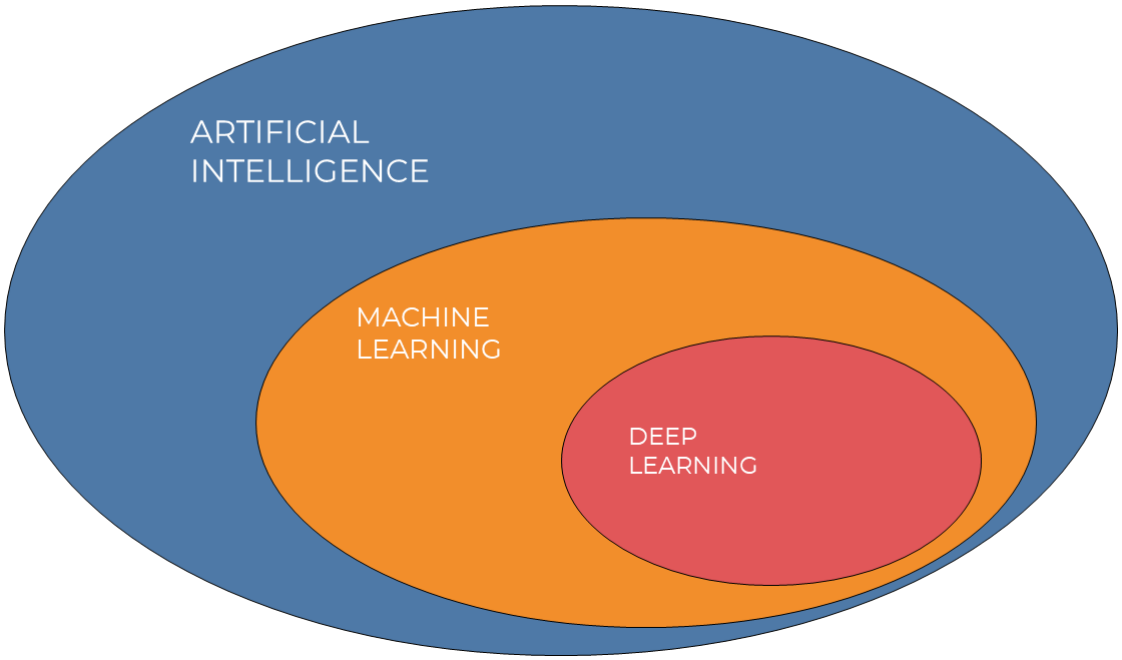

Figure 3.1: Representation of AI subsets.

learning process. This intuition allowed to create some linear models which however proved ineffective in solving some types of problems (e.g. training for XOR function).

After the first winter caused by the loss of interest in the sector there was an increase in research in the period from 1980 to 1990. It is during this second wave that the concepts of *neural network* and *hidden layers* were introduced. During this wave *back-propagation* [38] to train deep neural nets was developed and continue to remain key component of various advanced applications of Deep learning to this date. However, due to the lack of adequate computational resources, another "winter" came.

The third wave, which continues today, started in the early 2000s and led to the creation of deep networks, i.e. neural networks composed of numerous hidden layers. Furthermore, many different types of neural networks have been created over the years depending on the tasks of interest. Moreover, the technological advancement and the increase of computational capabilities enabled researchers around the world to train deeper and deeper neural networks and led to the popularisation of the term Deep Learning [39].

3.1.2 Deep Feedforward Networks

As mentioned in [40], Deep Feedforward Networks or Multi-layer perceptrons (MLPs) represent the cornerstone of deep learning models. There are other types of neural networks, such as RNNs [41] and CNNs [42] for example. As for the CNNs, these will be presented briefly later.

The Feedforwark Neural Network (FNN) tries to approximate some function $f^*(x)$,

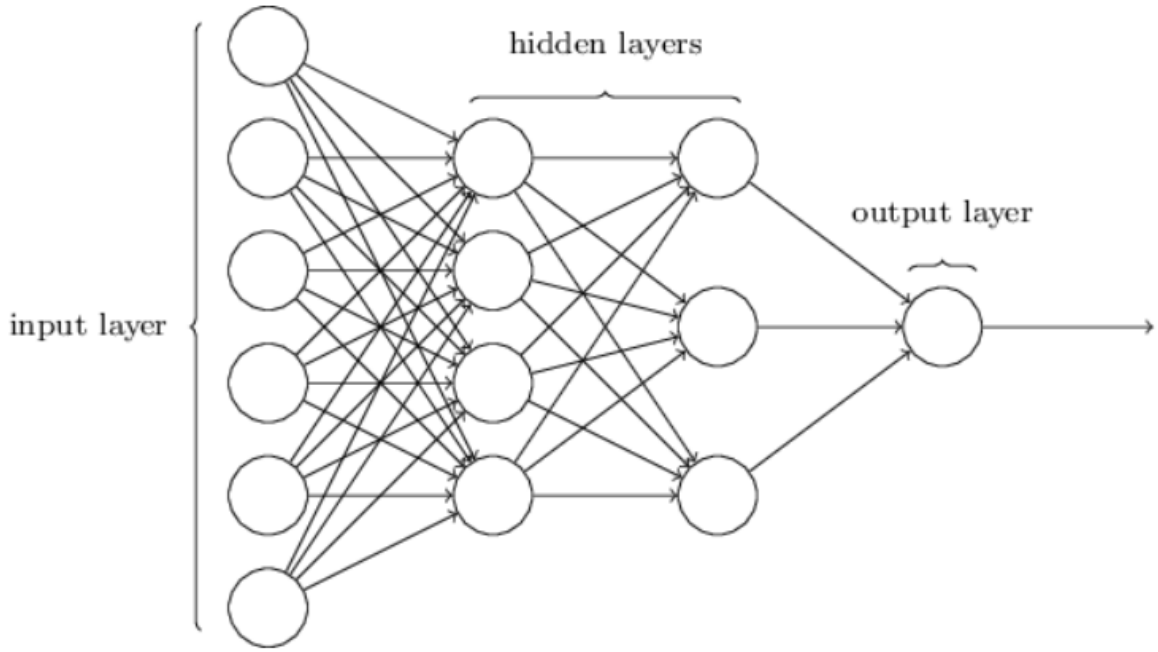

Figure 3.2: Schematic representation of a Feedforward Neural Network (FNN), in particular with a configuration designed for a binary classification task. From [43].

in case of a classification (or a regression) task the notation is $y = f^*(x)$. This notation highlights how the network maps from input x to output y. In particular, a more extensive and detailed writing is:

$$y = f(x, \theta) \tag{3.1}$$

where θ represents the *parameters* (or *weights*) that the network needs to learn in order to perform the approximation task. These networks are called *feedforward* because information flows through the function being evaluated from x, through the intermediate computations used to define f, and finally to the output y. The unidirectional flow of information is what distinguishes the FNNs from the RNNs. Schematically, an FNN is graphically representable as in Fig. 3.2. This flow of input x to output y through the layers that make up the network can be represented as a series of nested functions:

$$f(x; \theta) = f^3(f^2(f^1(x)))$$

the case of the above equation represents a network composed of three layers. The overall length of the chain gives the depth of the model. The first layer is called *input layer*, while the last is the *output layer*; all the layers between the input layer and the output layer are called hidden layers. Structurally each layer is represented by a vector and and the elements of the vector can be thought of as a neuron. In fact, the artificial neuron represents the constitutive unit of neural networks, in particular the FNN ones. A graphical representation of the structure of an artificial neuron is

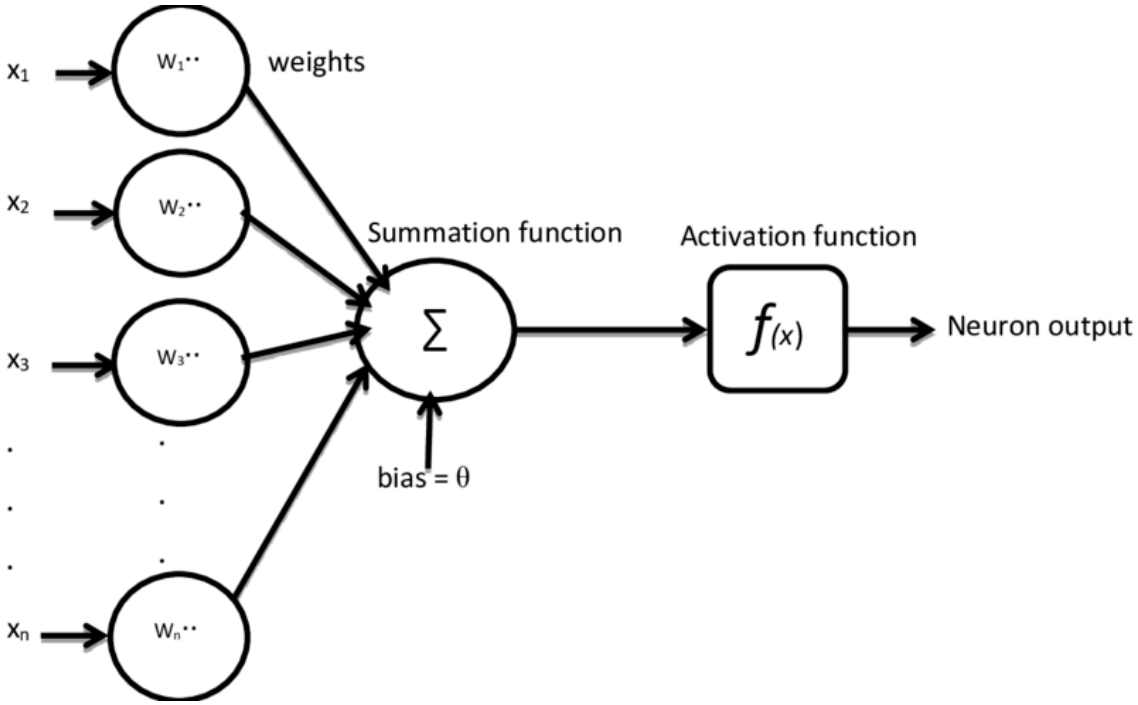

Figure 3.3: The structure of the artificial neuron. From [44].

shown in Fig.3.3. The structure of a neuron is quite simple: each neuron receives as input the outputs of all the neurons belonging to the previous layer of the network, multiplied by their respective weights. At this point these inputs are added together, and a bias is added to this sum. Finally, this sum is subjected to a function, called the *activation function*, which is responsible for applying a nonlinear transformation to the output of the neuron. These structures, and their name derives from the fact that they are inspired by the functioning of biological neurons. Anyway, it is best to think of FNN as a function approximation machines that are designed to achieve statistical generalization.

A concept worth emphasizing with regard not only to FNNs, but all deep learning models, is that their purpose is to be universal function approximators. While in fact in the simpler models, e.g. linear models such as linear regression, the model capacity is limited to linear functions, this is not true for neural networks. Given ϕ, defined as a nonlinear transformation, $\phi(x)$ represents the transformed input. This nonlinear transformation allows linear models to represent nonlinear functions of x: in a nutshell, ϕ represents a mapping. Normally, it would be necessary to find a suitable ϕ to achieve the desired input transformation but the strong point of deep learning is precisely to learn ϕ. In this way the model assumes the following formulation:

$$y = f(x;\, \theta;\, w) = \phi(x;\, \theta)^{\top} w \tag{3.2}$$

this is an example of a deep feedforward network, with ϕ defining a hidden layer.

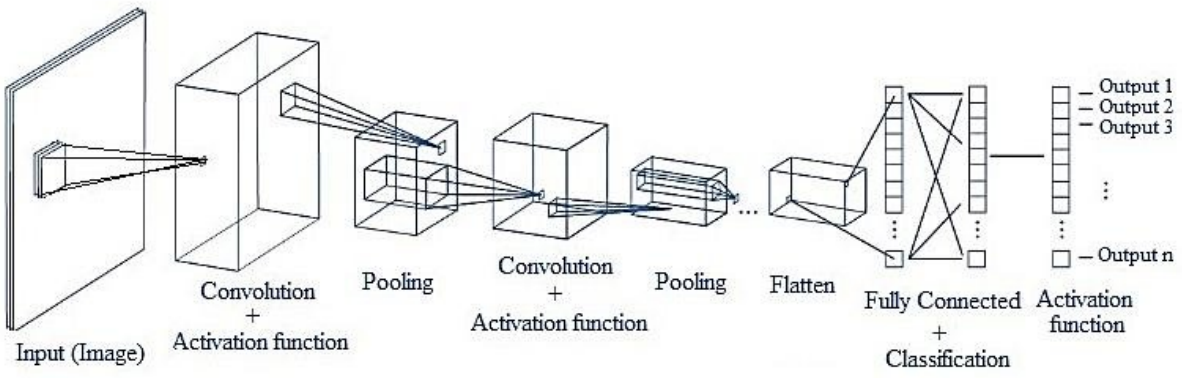

Figure 3.4: The basic structure of convolution neural networks. From [45].

3.1.3 Convolutional Networks

As pointed out in [40], at the structural and basic functioning level, a convolutional network does not differ from a more classical FNN. Rather, the first major difference from other networks lies in the type of data they are specialized for, i.e. data that has a known, grid-like topology. Examples of this type of data can be, as regards the 1D grid, time series, while images when talking about 2D grid (of pixels). As it is easy to guess, the name Convolutional Neural Network indicates that the network uses a mathematical operation called convolution. This operation has already been mentioned and briefly explained in Sec. 2.1.7. As anticipated, therefore, convolutional networks are networks that use convolutions instead of matrix multiplication in at least one layer. This difference is justified by the fact that while in the FNNs the fundamental unit was the artificial neuron, instead in the CNNs they are banks of convolutional filters. Intuitively, a CNN can be graphically represented as in Fig. 3.4.

While the convolutions discussed in Sec. 2.1.7 are the result of a human engineering process that leads to the definition of suitable filters to obtain from the signals or images processed the features considered important (such as eliminating certain frequencies from an audio signal or identifying the angles present in an image), the advantage of CNNs lies in the fact that filters are automatically trained to extract useful features to obtain output y from input x. Referring to the terminology of CNNs, and using the following equation to represent the convolution operation:

$$s(t) = (x * w)(t) \tag{3.3}$$

the first argument of convolution, x, is often referred to as the input while the second argument w is the filter (or *kernel*). The output of the convolution operation is often called *feature map*. Other essential elements in defining a convolutional layer are the kernel size, the padding and the stride. Referring to Fig. 3.5, where a convolution

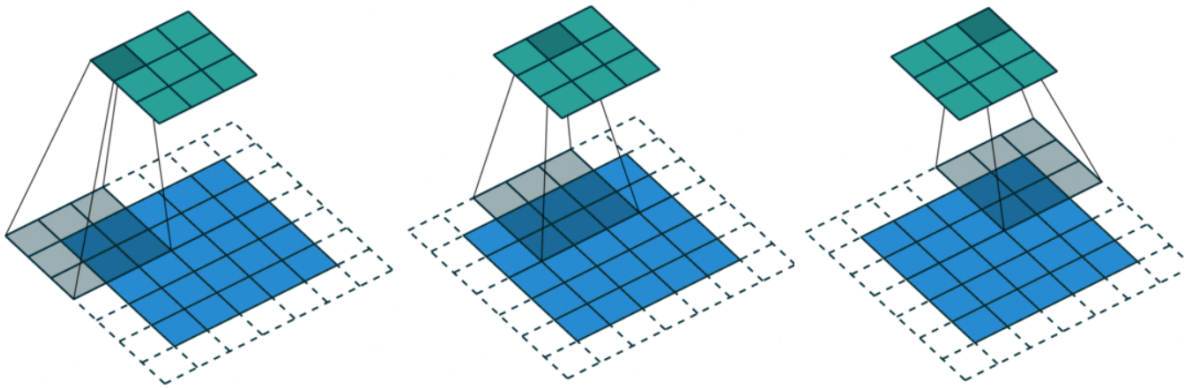

Figure 3.5: Representation of convolution operation. From `https://towardsdatascience.com/review-dilated-convolution-semantic-segmentation-9d5a5bd768f5`.

process is represented on a 2D input, it is possible to identify these elements. The input is represented by the 5×5 blue base, while the kernel, with its 3×3 size, is represented by the gray shadow that operates on the input. The result of the operation, that is the feature map, is represented by the green 3×3 square. By stride we refer to the length of the step taken by the convolution, in the case of the example in the figure the stride is 2 since between each phase the filter performs a translation of 2 values on the input. Finally, it must be considered that the feature map resulting from the convolution operation can take on a smaller, equal or larger size than the original input. This aspect can be controlled using padding, which in the example is represented by the dashed outline of the input. The following formula is able to define, based on the size of the input and the value of the stride and padding, the size of the output[1]:

$$\frac{(W - F + 2P)}{S} + 1 \tag{3.4}$$

with W representing the input width, F the filter width, P the applied padding and finally S the chosen stride parameter. In fact it can be verified that starting from an initial size 5×5, with stride 2 and padding 1, using a filter with size 3, the final output size will be:

$$\frac{(5 - 3 + 2)}{2} + 1 = 3$$

i.e. an image 3×3.

Another building block of convolutional neural networks is represented by the *Pooling* layer. These layers are almost always present in CNN, whether it is 1D or 2D data. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting[2]. There are several pooling function whose job is to replace the output of

[1] `https://cs231n.github.io/convolutional-networks/`.

[2] A model overfits the training data when it describes features that arise from noise or variance

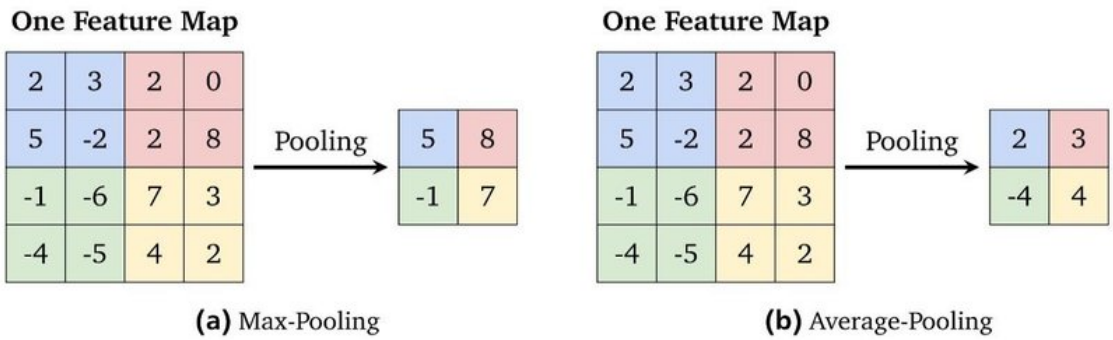

Figure 3.6: Example for the max-pooling (a) and the average-pooling (b) with a kernel size of 2×2 and a stride of 2×2. From [47].

the net at a certain location with a summary statistic of the nearby outputs. The most common forms are max-pooling and average-pooling: while the former operation reports the maximum output within a rectangular neighborhood, the latter include the average: in Fig. 3.6 examples are shown graphically. In all cases, pooling helps to make the representation become approximately invariant to small translations of the input. Invariance to translation means that if we translate the input by a small amount, the values of most of the pooled outputs do not change.

In conclusion, as mentioned in [48], convolutions exploits three aspects that distinguish them from FNNs: *sparse interactions*, *parameter sharing* and *equivariant representations*. While in FNNs every output unit interacts with every input unit, CNNs are characterized by sparse interactions as the kernel is smaller than the input. This allows to store fewer parameters, reducing memory consumption and increasing statistical efficiency. As for the sharing of parameters, just think that in FNNs each element in the weight matrix is used only once when calculating the output of a layer; in CNNs, on the other hand, the network learn only one set of parameters that make up the values of the convolutional filters. Finally, the equivariance to translation is a direct consequence of the parameter sharing. To say a function is equivariant means that if the input changes, the output changes in the same way[3].

3.1.4 Illustration of the neural learning process

The learning process of a neural network is composed of several steps and elements. It should be pointed out that in order to learn effectively, a NN requires the training data to pass over and over again: each passage of all the training data in the network is defined as an *epoch*.

First of all, the weights that make up the neural network, in the different hidden

in the data, rather than the underlying distribution from which the data were drawn. Overfitting usually leads to loss of accuracy on out-of-sample data [46].

[3]Specifically, a function $f(x)$ is equivariant to a function g if $f(g(x)) = g(f(x))$.

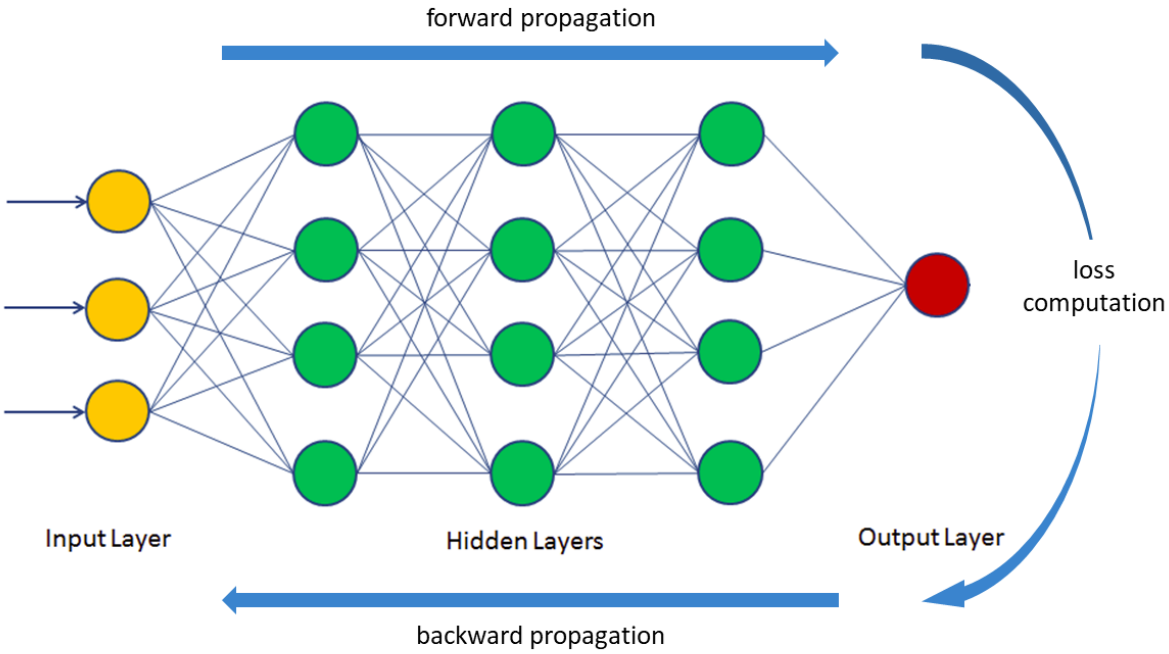

Figure 3.7: The learning procedure for an Artificial Neural Network. The input x is submitted to the network through the feedward propagation in order to obtain an estimate of y, that is $\hat{y}$. The error between y and $\hat{y}$ is calculated using the loss function. Using the back-propagation the error is propagated backwards in the network to correct the weights that make up the different layers.

layers are initialized randomly, or according to certain probability distributions depending on preferences. It is also important to underline how the learning process of a neural network basically consists of an optimization problem. In fact, as explained previously, NNs task is to carry out a mapping from an input x to an output y. Obviously the result obtained from the network i.e. $\hat{y}$ will not coincide perfectly with what is the desired output (ground truth) y. The function that deals with describing the difference between the obtained output $\hat{y}$ and the desired output y is called *loss function* [49] and it is the function that is being optimized. The learning process, therefore, basically consists in submitting the input data x to the network, obtaining the output $\hat{y}$ and comparing it with y through the loss function to calculate the error in the estimate of y and thus being able to adjust the value of the weights that make up the network itself, so that at the next epoch this returns, hopefully, a value of $\hat{y}$ closer to y, reducing the error [50]. The operation of passing the input through the network to obtain the output y is called *forward propagation*, while the operation of propagating the error calculated by the loss function at all weights to make them update is called *backward propagation*, or more commonly back-propagation [51]: an illustration of this process can be seen in Fig. 3.7.

The goal of optimization is to bring the loss value as close to zero as possible, i.e.

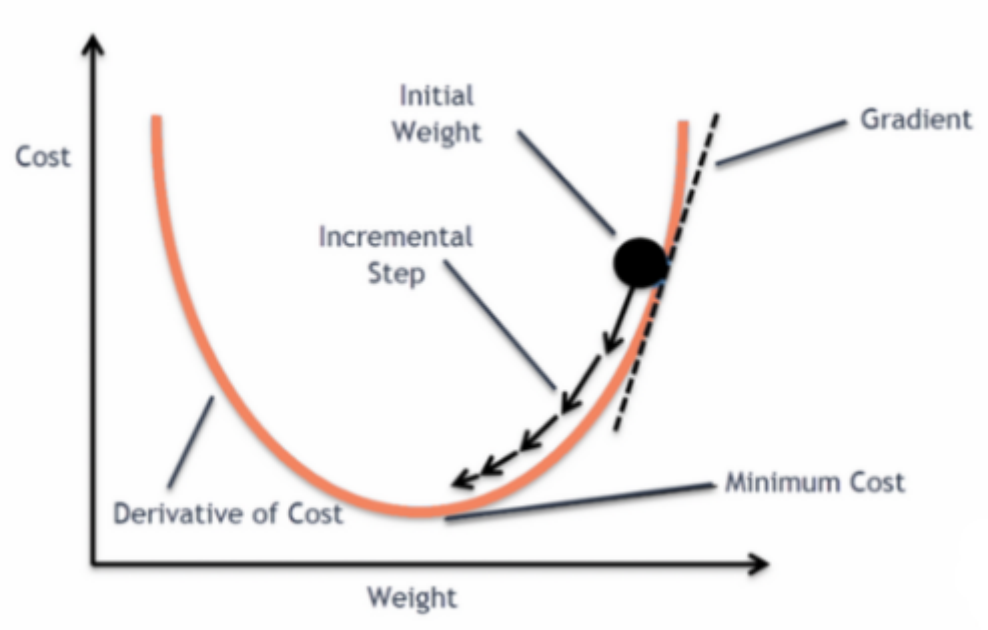

Figure 3.8: Gradient Descent. Cost on the y axis refers to the value of the loss function, in fact the loss function is also called the cost function. From `https://blog.clairvoyantsoft.com/the-ascent-of-gradient-descent-23356390836f`.

to obtain an estimate of $\hat{y}$ as close as possible to the real value of y. The technique used to achieve this is called *gradient descent*[4]. This technique changes the weights in small increments with the help of the calculation of the derivative (or gradient) of the loss function, which allows us to see in which direction to descend towards the global minimum; this is done in general in batches of data in the successive iterations (epochs) of all the dataset that we pass to the network in each iteration. Gradient descent uses the gradient of the loss function, so as to be able to descend towards the minimum point, when updating the parameters. The process consists in chaining the derivatives of the loss of each hidden layer from the derivatives of the loss of its upper layer, incorporating its activation function in the calculation. In each of the iterations, once all the neurons have the value of the gradient of the loss function that corresponds to them, the values of the parameters are updated in the opposite direction to that indicated by the gradient. Mathematically if a multi-variable function $F(x)$ is defined and differentiable in a neighborhood of a point a, then $F(x)$ decreases fastest if one goes from a in the direction of the negative gradient of F at a, $-\nabla F(a)$. Therefore considering a as a single parameter of the network, the weight w_{ij} for example and

$$\nabla F(w_{ij}) = \frac{dError}{dw_{ij}}$$

[4]Stochastic Gradient Descent (SGD) [52], which is the most common implementation of the algorithm, consists in calculating the gradient and updating the parameters not after each epoch but rather after the passage of small groups of data, called mini-batches.

as the derivative of the error with respect to the weight w_{ij}, the rule for updating weights is:

$$w_{ij(n+1)} = w_{ij(n)} - \alpha \nabla F(w_{ij(n)}) \tag{3.5}$$

with α representing the *learning rate* of the *optimizer*. Because we want to move against the gradient, toward the minimum, α needs to be small enough so that:

$$F(w_{ij(n)}) \geq F(w_{ij(n+1)})$$

So, to summarize, with back-propagation we mean a method to change the value of the weights and biases of a neural network so that it can correctly map the input x to the output y. This method first includes a loss function which takes care of calculating the difference between estimated $\hat{y}$ and real y. This information is exploited by an optimization algorithm (the optimizer), which in the above case is called gradient descent.

3.2 State of the Art

The state of the art regarding audio denoising, in particular speech enhancement, is full of very advanced models, very computationally heavy and that reach pretty good performances. Among these networks there are different approaches, some consist of Convolutional Neural Networks such as those presented in Sec. 3.1.3 , others consist of Recurrent Neural Networks or Generative Adversarial Networks (GANs) [53]. Among these approaches, moreover, there are some that use as input audio in their wave form, i.e. raw audio, while others first extract features from the audio and proceed working on these extracted features: almost always these are features extracted from the audio represented in the frequency domain (see Sec. 2.2).

3.2.1 Features approaches

Very often the approaches that exploit features extracted from the audio or audio represented in the frequency-domain turn out to be the most performing as regards the different tasks associated with speech. In particular, when it comes to speaker recognition and speech enhancement, the use of audio representation in the frequency-domain makes it possible to identify fundamental aspects of audio signals.

A very interesting model regarding this typology is represented by the one presented in [54]. This approach is basically based on the implementation of an enhanced version of the well known in the literature U-Net [55]. The U-Net architecture is basically an autoencoder [56] characterized by the presence of skip connections [57] that connect each level of input compression up to the compressed representation to

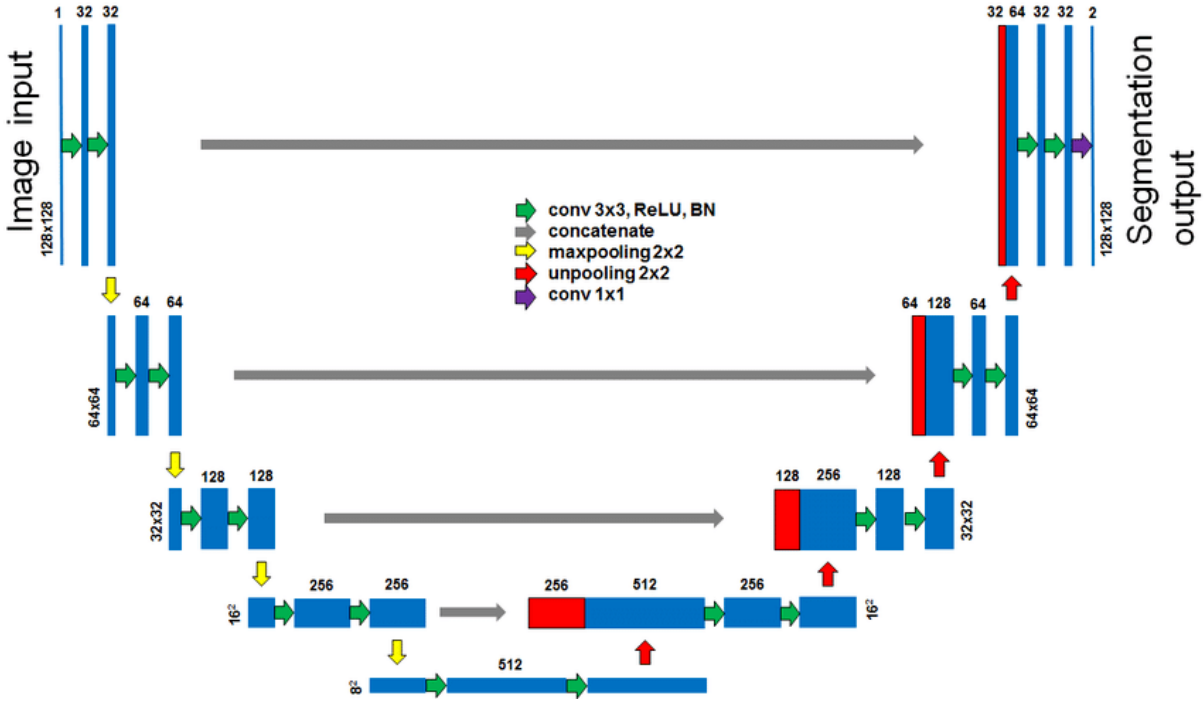

Figure 3.9: U-Net Convolutional Neural Network model. The U-Net model contains two parts: Down-sampling (left half) and up-sampling (right half). From [58].

the corresponding decompression level in the reconstruction from the compressed to the extended representation: a visual representation is shown in Fig. 3.9. The main idea is to supplement a usual contracting network by successive layers, where pooling operations are replaced by upsampling operators. Basically, instead of using a pooling mechanism to compress the input, using synthetic statistics, convolutional layers are used with a stride parameter (see Sec. 3.1.3) greater than one so that the input to each layer shrinks more and more until it reaches a very compressed representation compared to the initial dimensions. One important modification in U-Net is that there are a large number of feature channels in the upsampling part, which allow the network to propagate context information to higher resolution layers. As a consequence, the expansive path is more or less symmetric to the contracting part, and yields a u-shaped architecture. The network only uses the valid part of each convolution without any fully connected layers.

The innovative approach of this work presented in [54] consists in the use of Short-time Fourier Transform (STFT) [59] in input to train the model in order to create a mask that then multiplied by the same input returns the audio cleaned of noise after having performed the Inverse Short-time Fourier Transform (ISTFT). Mathematically, the process can be represented as:

$$\hat{Y}_{t,f} = \hat{M}_{t,f} \cdot X_{t,f} \tag{3.6}$$

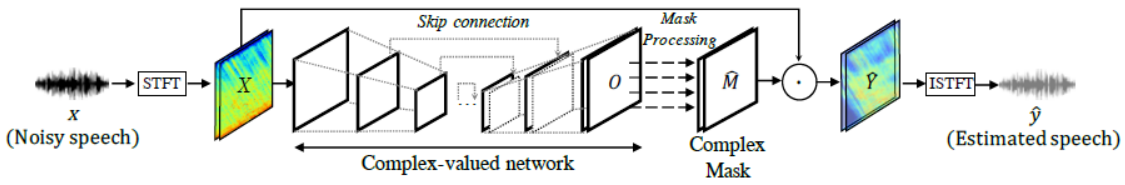

Figure 3.10: Illustration of speech enhancement framework with DCUnet. From [54].

Table 3.1: Quantitative evaluation results of noisy audio, Wiener filtering [61] algorithm and Large-DCUnet-20 method. Higher score means better performance where bold text indicates highest score per evaluation measure. CSIG: Mean opinion score (MOS) predictor of signal distortion CBAK: MOS predictor of background-noise intrusiveness COVL: MOS predictor of overall signal quality PESQ: Perceptual evaluation of speech quality SSNR: Segmental SNR.

	CSIG	**CBAK**	**COVL**	**PESQ**	**SSNR**
Noisy	3.35	2.44	2.63	1.97	1.68
Wiener	3.23	2.68	2.67	2.22	5.07
Large-DCUnet-20	**4.34**	**4.10**	**3.81**	**3.24**	**16.85**

with $\hat{Y}_{t,f}$ representing the enhanced audio, $\hat{M}_{t,f}$ the estimated mask and $X_{t,f}$ the input noisy audio: the subscript t, f stands for time-frequency representation, that is the representation of the audio obtained through STFT. In fact a popular approach to speech enhancement is to optimize a mask which produces a spectrogram of clean speech when applied to noisy input audio: for this particular architecture a schematic representation is shown in Fig. 3.10. With reference to $\hat{M}_{t,f}$, the ground truth for the mask is computable as $M_{t,f} = Y_{t,f}/X_{t,f}$. Another very important aspect in this work is the realization of an *ad hoc* loss function: the weighted-SDR loss by building upon a previous work which attempts to optimize a standard quality measure, Source-to-Distortion Ratio (SDR) [60], but slightly modified. Mathematically the loss function takes the form:

$$loss_{wSDR} = (x, y, \hat{y}) := \alpha \, loss_{SDR}(y, \hat{y}) + (1 - \alpha) loss_{SDR}(z, \hat{z}) \tag{3.7}$$

with:

$$loss_{SDR}(y, \hat{y}) = -\frac{< y, \hat{y} >}{\|y\| \|\hat{y}\|} \tag{3.8}$$

and $\hat{z}$ and α equal respectively to $\hat{z} = x - \hat{y}$ and $\alpha = \frac{\|y\|^2}{\|y\|^2 + \|z\|^2}$.

The model was trained using two datasets, Diverse Environments Multichannel Acoustic Noise Database (DEMAND) [62] and the Voice Bank corpus [63], and obtained excellent results, measured through objective measures of the quality of the speech, shown in Tab. 3.1.

3.2.2 Waveform approaches

Approaches that exploit audio signals in the time-domain, i.e. waveform approaches, are gaining increasing interest from researchers. The main reason lies in the fact that by avoiding transformations, calculations and therefore computational power are saved. Furthermore, the use of audio in their raw form allows to keep the phase unchanged, which would not be possible with features such as MFCC for example.

One of the most recently explored techniques involves the use of GANs, or Generative Adversarial Networks used mainly in the field of computer vision to generate realistic images while as regards their application to the field of speech generation or audio enhancement they had not yet been used. The first work in this sense is discussed in [64]. Generative models basically consist of two distinct networks called respectively generator and discriminator: while the generator tries to fool the discriminator, the discriminator tries to keep from being fooled. The generator model takes a fixed-length random vector as input and generates a sample in the domain. The vector is drawn randomly from a Gaussian distribution, and is used to seed the generative process. After training, points in this multidimensional vector space will correspond to points in the problem domain, forming a compressed representation of the data distribution. This vector space is referred to as a latent space, or a vector space comprised of latent variables. Latent variables, or hidden variables, are those variables that are important for a domain but are not directly observable. In parallel the discriminator model takes an example from the domain as input (real or generated) and predicts a binary class label of real or fake (generated): while the real examples come from the training dataset, the generated ones are output by the generator model. After the training process, the discriminator model is discarded as we are interested in the generator. In essence, therefore, the generator learns to generate a certain type of data, starting from a random vector, never coming into contact with examples of the data it must represent, but only receiving feedback from the discriminator that communicates to the generator if the given output is it correct or not. A schematic representation of the architecture of a GAN is illustrated in Fig. 3.11. In the Speech Enhancement Generative Adversarial Network (SEGAN) the generator G takes in input the noisy input x together with the latent representation z, and its goal is to reproduce the clean audio $\hat{x}$.

$$\hat{x} = G(x)$$

The G network is structured similarly to an auto-encoder with an encoding part composed of convolutional layers with strides that compress the input proceeding to the center and then reach a compressed representation c, this is concatenated with the latent representation z and then subjected to the decoding process with a series of layers of deconvolution: schematically the generator G appears as in the illustration

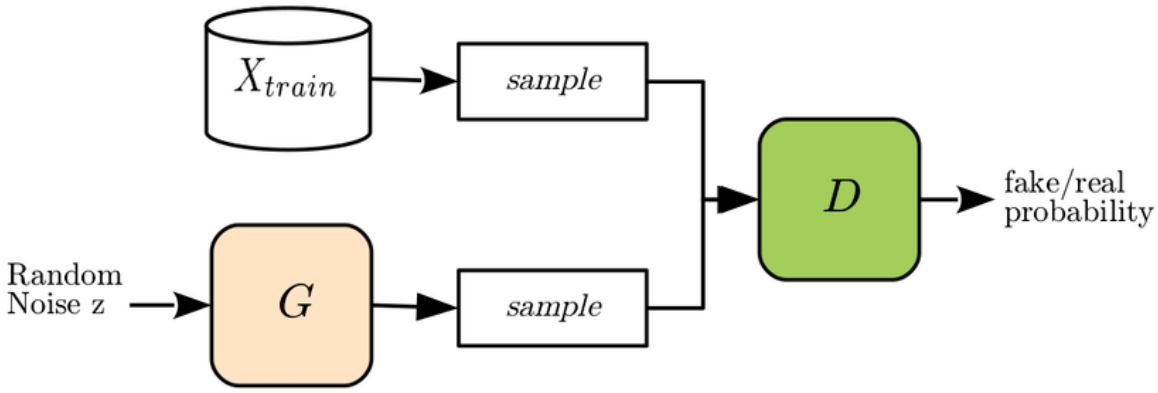

Figure 3.11: Generative Adversarial Network (GAN) schema. From [65]

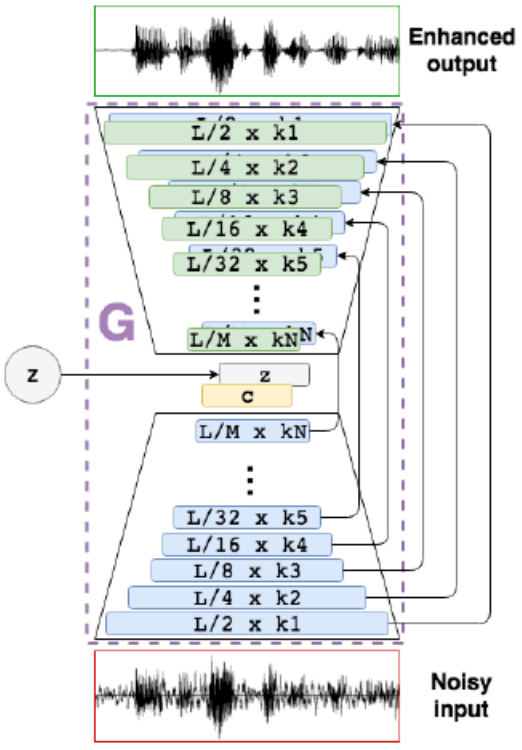

Figure 3.12: Encoder-decoder architecture for speech enhancement (G network). The arrows between encoder and decoder blocks denote skip connections. From [64].

Table 3.2: Quantitative evaluation results of noisy audio, Wiener filtering algorithm and SEGAN and DEMUCS methods. Higher score means better performance where bold text indicates highest score per evaluation measure.

	CSIG	**CBAK**	**COVL**	**PESQ**	**SSNR**
Noisy	3.35	2.44	2.63	1.97	1.68
Wiener	3.23	2.68	2.67	2.22	5.07
SEGAN	3.48	2.94	2.80	2.16	**7.73**
DEMUCS	**4.22**	**3.25**	**3.52**	**2.93**	-

in Fig. 3.12. Each compression convolution layer is connected to the corresponding deconvolution layer through skip connections, as in the case of the network presented in Sec. 3.2.1: these connections allow for better training behavior, as gradients can flow deeper into the network structure [66]. The output of this autoencoder is the cleaned audio, which is then subjected to the discriminator D. The network D follows the same one-dimensional convolutional structure as G encoder stage, and it fits to the conventional topology of a convolutional classification network. The loss function used is the one commonly used for GANs, that is a game of minimization and maximization that involves generator and discriminator whose purpose is to compete with each other, with some tricks. Tab. 3.2 shows the results of this approach comparing it with the Wiener filter and the noisy input: the datasets used for the evaluation are the same used for the model presented in Sec. 3.2.1.

Another very important approach is the one presented in [67]. Like the one presented in Sec. 3.2.1 the basic model consists of a U-Net, that is a convolutional autoencoder with the encoding layers connected to the decoding layers through skip connections. The main differences with respect to the approach already mentioned is that in this case the network is able to take audio input in raw form and to output the cleaned audio, and that a sequence modeling network is applied to the output of the encoding part, in particular a Long-Short Term Memory (LSTM) layer [68]: this particular type of architecture is called DEMUCS [69] and is graphically represented in Fig. 3.13. In this particular case, the encoder network E gets as input the raw wave form and outputs a latent representation $E(x) = z$; next, the sequence modeling R network takes the latent representation z as input and outputs a non-linear transformation of the same size, $R(z) = LSTM(z) + z$, denoted as $\hat{z}$. Finally, the decoder network D, takes as input $\hat{z}$ and outputs an estimation of clean signal $D(\hat{z}) = \hat{y}$. The L1 loss over the waveform together with a multiresolution STFT loss over the spectrogram magnitudes similarly to the one proposed in [70, 71] are used. In accordance with the models presented previously, this was also tested on the DEMAND dataset on the basis of the appropriate metrics to evaluate the performance of speech enhancement. The results of the model test measurement are shown in Tab. 3.2.

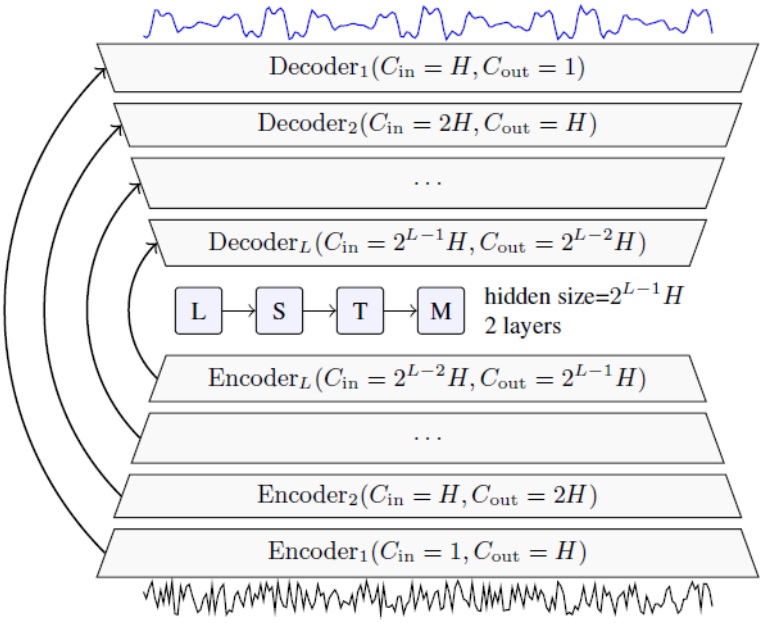

Figure 3.13: DEMUCS with the noisy speech as input on the bottom and the clean speech as output on the top. Arrows represents UNet skip connections. H controls the number of channels in the model and L its depth. From [67].

Table 3.3: Quantitative evaluation results of noisy audio, Wiener filtering algorithm and WaveNet method. Higher score means better performance where bold text indicates highest score per evaluation measure.

	CSIG	**CBAK**	**COVL**	**PESQ**	**SSNR**
Noisy	3.35	2.44	2.63	1.97	1.68
Wiener	3.23	2.68	2.67	2.22	5.07
WaveNet	**3.62**	**3.23**	**2.98**	-	-

The last approach presented, which is the one on which the thesis work is based, is represented by the work carried out in [72]. Like the other approaches presented in this section, it is composed of a model, called WaveNet, capable of handling audio in its raw form, but also has the peculiarity that it does not require any kind of preprocessing on the input audio. Another significant difference lies in the fact that the WaveNet is basically a Convolutional Neural Network, not conforming to the autoencoder structure that is present in all the approaches presented and in many others that represent the current state of the art. However, this model also uses skip connections to allow gradients to flow across the depth of the network. The technical details regarding the structure and particular implementations of convolution layers and loss functions will be covered in the next chapter. Like most of the models concerning speech enhancement, the network was tested in accordance with the other models presented on the DEMAND dataset and obtained the results shown in Tab. 3.3.

Chapter 4

Proposed Approach

After having discussed the fundamentals of audio signals and the theory concerning the basics of deep learning and the state of the art for the main task of this work, this chapter provides an overview of the whole proposed approach and of the corresponding technical implementation. This chapter is dedicated to presenting the thesis work from a technical point of view. For this reason, the datasets used will be presented in detail below, followed by a description of the implemented architectures with their characteristics and variants. Finally, the corresponding technical implementation, the work pipeline and the various attempts made will be described.

4.1 Datasets

In order to train the models used to carry out the speech enhancement task, first of all it was necessary to identify the datasets for the intended objective. The research was based on first of all numerous audio datasets and also with sufficient recording quality to have a good baseline.

For this purpose, four datasets have been selected: the Spoken Interaction with Interpretation in Switzerland (SIWIS) [73, 74], the Music, Speech, and Noise Corpus (MUSAN) [75], the well known VoxCeleb [76, 77, 78] and finally the DEMAND dataset [79]. The datasets will be presented below in their characteristics and composition.

4.1.1 SIWIS

The SIWIS database can be defined as a collection of speech data recorded by bilingual and trilingual speakers in a controlled environment, thus producing traces free of background noise. Inspired by the EMIME project [80] in which languages such as English, Japanese, Mandarin and Finnish were involved, for SIWIS the focus is on the three main official languages in Switzerland (French, German, Italian) and English. The recordings involve 36 accentless bilingual speakers (among which 14 trilingual

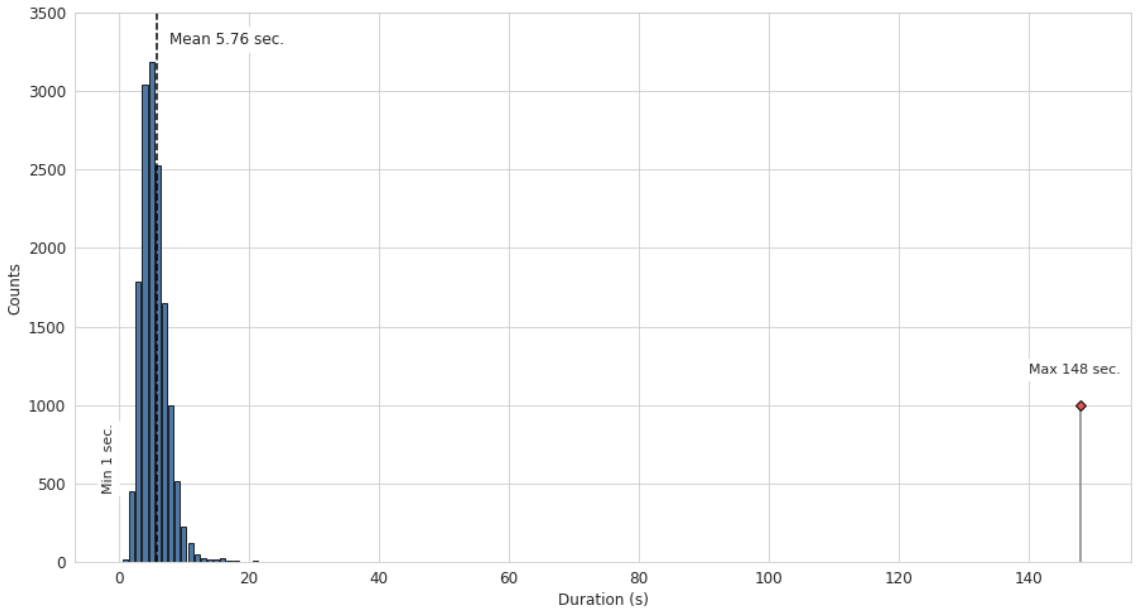

Figure 4.1: Duration distribution for SIWIS database audio recordings. Minimum, maximum and mean durations are annotated.

Table 4.1: Recording numbers and durations for SIWIS database. From [74].

Language	**Sessions**	**Prompts**	**Total duration**
French	31	5,332	512 min.
English	22	3,771	350 min.
German	17	2,903	266 min.
Italian	16	2,738	287 min.
Total	86	14,744	1,415 min. ∼23.6 hours

ones) and the reading material is mainly composed of news or parliamentary sentences. Besides, some sentences were repeated with some emphasis. Additionally, a 240-words text was read with some involvement. In detail, each set of 171 prompts for each language is divided in 5 parts: Europarliament statements, sentences from newspapers, semantically unpredictable sentences, the statements from the Europarliament but with some emphasis and finally some sentences read by "Le petit prince".

Summary statistics regarding the duration of the recordings are visible in Tab. 4.1, while as regards the distribution of the duration of the audio it is represented graphically in Fig. 4.1 where it is also possible to see which is the shortest audio duration (1 sec.), the longest one (∼ 150 secs.) and the overall average duration of the audio contained in the database (∼ 6 secs.).

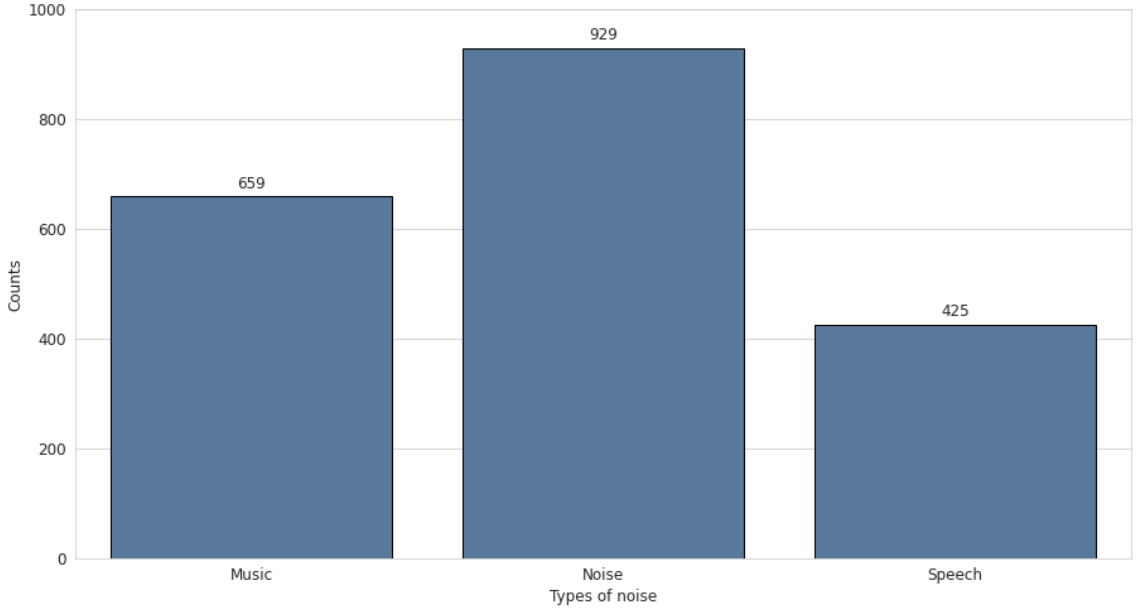

Figure 4.2: Distribution of the audio contained in the MUSAN database divided into Music, Noise and Speech.

4.1.2 MUSAN

The dataset consists of music from several genres, speech from twelve languages, and a wide assortment of technical and non-technical noises. In particular the corpus consists of approximately 109 hours of audio that is in the US Public Domain or under a Creative Commons license.

The first portion of the corpus consists of about 60 hours of speech. It contains 20 hours and 21 minutes of read speech from Librivox[1], all of which are in the Public Domain.

The music portion of the corpus consists of 42 hours and 31 minutes and is divided into Western art music and popular genres.

Finally the noise portion of the corpus contains 929 files of assorted noises, with a total duration of about 6 hours. These range from technical noises, such as DTMF tones, dialtones, fax machine noises, and more, as well as ambient sounds, such as car idling, thunder, wind, footsteps, paper rustling, rain, animal noises, etc: no recordings with intelligible speech are included.

In Fig. 4.2 it is possible to graphically view the quantitative subdivision of the audio tracks present in the MUSAN dataset divided into the subcategories music, noise and speech.

[1] `https://librivox.org/`. The purpose of the Librivox project is to make all books in the public domain available, narrated by real people and distributed for free, in audio format on the internet.

Table 4.2: Dataset statistics for both VoxCeleb1 and VoxCeleb2. Note that VoxCeleb2 is more than 5 times larger than VoxCeleb1. POI: Person of Interest. From [77].

Dataset	**VoxCeleb1**	**VoxCeleb2**
n. of POIs	1,215	6,122
n. of male POIs	690	3,761
n. of hours	352	2,442
n. of utterances	153,516	1,128,246
Avg. n. of utterances per POI	116	185
Avg. length of utterances (s)	8.2	7.8

Table 4.3: Train and test sets details for VoxCeleb 1 & 2. From `http://www.robots.ox.ac.uk/~vgg/data/voxceleb/vox1.html` and `http://www.robots.ox.ac.uk/~vgg/data/voxceleb/vox2.html`.

Dataset	**Portion**	**n. of speakers**	**n. of utterances**
VoxCeleb1	train	1,211	148,642
	test	40	4,874
VoxCeleb2	train	5,994	1,092,009
	test	118	3,6237

4.1.3 VoxCeleb

VoxCeleb contains over 1 million utterances for more than 7000 celebrities, extracted from videos uploaded to YouTube. The dataset is gender balanced and the speakers span a wide range of different ethnicities, accents, professions and ages. All the recorded audios are degraded with real world noise, consisting of background chatter, laughter, overlapping speech, room acoustics, and there is a range in the quality of recording equipment and channel noise.

The VoxCeleb dataset consists of two parts, not overlapping as regards the audio and the identities of the speakers, VoxCeleb1 and VoxCeleb2, made at different times. In detail, these two parts are composed as shown in Tab. 4.2. These subsections are then further divided into portions usually used as training and test sets. In particular, for very powerful models that require a lot of data to perform a training VoxCeleb2 is often used in its entirety, together with the training portion of VoxCeleb1, and only the test portion of the latter is used as a test set.

The details regarding the division into training set and test set of the two datasets can be seen in Tab. 4.3.

4.1.4 DEMAND

The DEMAND dataset is basically a clean and noisy parallel speech database designed to train and test speech enhancement methods that operate at 48kHz. For every clean audio there is also its counterpart with the addition of noise. This makes this dataset very convenient when it comes to training speech enhancement systems as it allows to start with ready-made data.

The peculiarity of this database is that its test portion is very often used as a reference in order to compare the performances of different models with the same starting data, using common metrics for the objective measurement of the improvement in audio quality.

A demonstration of this is present in Sec. 3.2: all the models presented, but also others omitted, which are part of the state of the art in the field of speech enhancement are compared with each other using this dataset.

4.2 Models

This section is dedicated to the in-depth presentation of the deep learning architectures implemented during the thesis work with the aim of fulfilling the task initially set, that is to improve the quality of audio signals by removing background noise so as to improve speaker recognition performances. For this reason, the architectures presented will be essentially two: first of all the model that deals with the denoising task and, subsequently, the one that has the task of recognizing the speaker.

4.2.1 Denoising Net

As mentioned in Sec. 3.2, the model chosen to perform the denoising task is inspired by the WaveNet [72, 81]. As already stated, this is substantially a very simple model, essentially composed of 1D convolutional layers and therefore capable of taking audio signals in their raw form as input. This aspect is very important since it means that the signal does not need pre-processing or particular transformations that could weigh on the efficiency of the denoising task or on general perfomances. In fact, it is necessary to think of the denoising task as a task performed by a device in real time. This therefore requires that the operation, in addition to being effective, must also be very efficient.

To fully understand the architecture of the WaveNet it is necessary, first of all, to start from the assumption that, originally, this model was not created to perform the task of speech enhancement but rather that of generating speech starting from text: this particular task in the field of audio signals is therefore called text-to-speech (TTS).

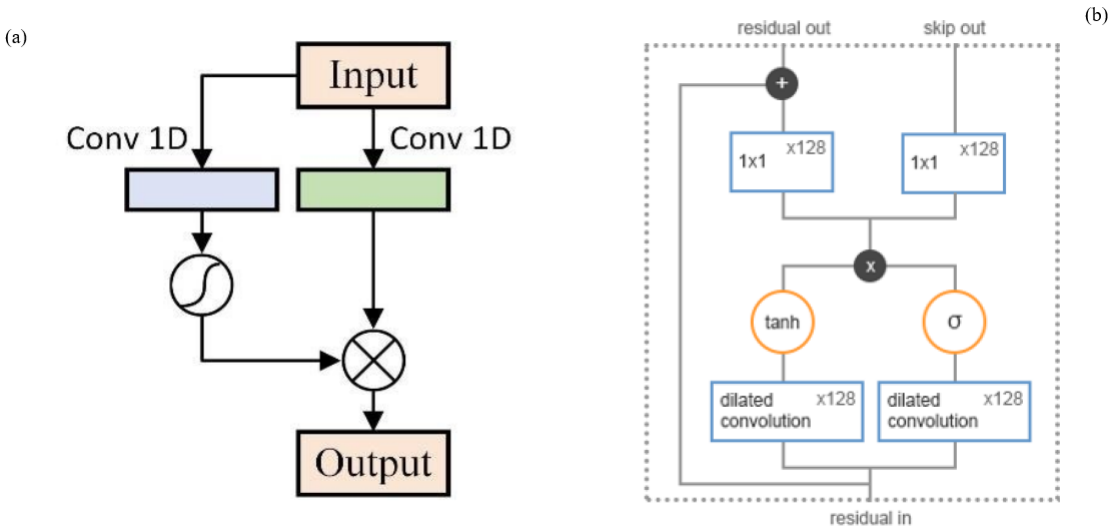

Figure 4.3: (a) An illustration of the GLU convolutional block. Two pathways of convolution operations are involved: the right pathway is the main pathway, and the pathway on the left is the additional pathway used to fulfil the gating mechanism. From [86]. (b) WaveNet's Residual Block layer. From [81].

As explained in [81], the generative model operate directly on the raw audio waveform. The joint probability of a waveform $x = \{x_1, \ldots, x_T\}$ (composed of sequential samples from time $t = 1$ to $t = T$) is factorised as a product of conditional probabilities as follows:

$$p(x) = \prod_{t=1}^{T} p(x_t \,|\, x_1, \ldots, x_{t-1}) \tag{4.1}$$

A model of this type is called *autoregressive* and it shapes the probability distribution of the next sample x_t given some fragment of previous samples. The next sample is produced by sampling from this distribution. An entire sequence of samples is produced by sequentially feeding previously generated samples back into the model[2]. This original WaveNet implementation turns out to be an audio domain adaptation of the PixelCNN architecture [82, 83], a specific generative model for images. Among the features inherited from PixelCNN are: causality[3], gated convolutional units [84], discrete softmax output distributions and the possibility of conditioning the model. Among the aspects that are original compared to the counterpart for the images there are: dilated convolutions [85] and non-linear quantization[4]. Some of the main features of the generative WaveNet are presented below.

[2]This procedure enforces time continuity in the resulting audio waveforms.

[3]An exhaustive explanation of what is meant by causal convolution is available at the link `https://theblog.github.io/post/convolution-in-autoregressive-neural-networks/`.

[4]With linear quantization every increment in the sampled value corresponds to a fixed size analogue increment. With non-linear quantization normally there is some sort of logarithmic encoding (e.g. μ-Law or A-law), so that the increment for small sample values is much smaller than the increment for large sample values.

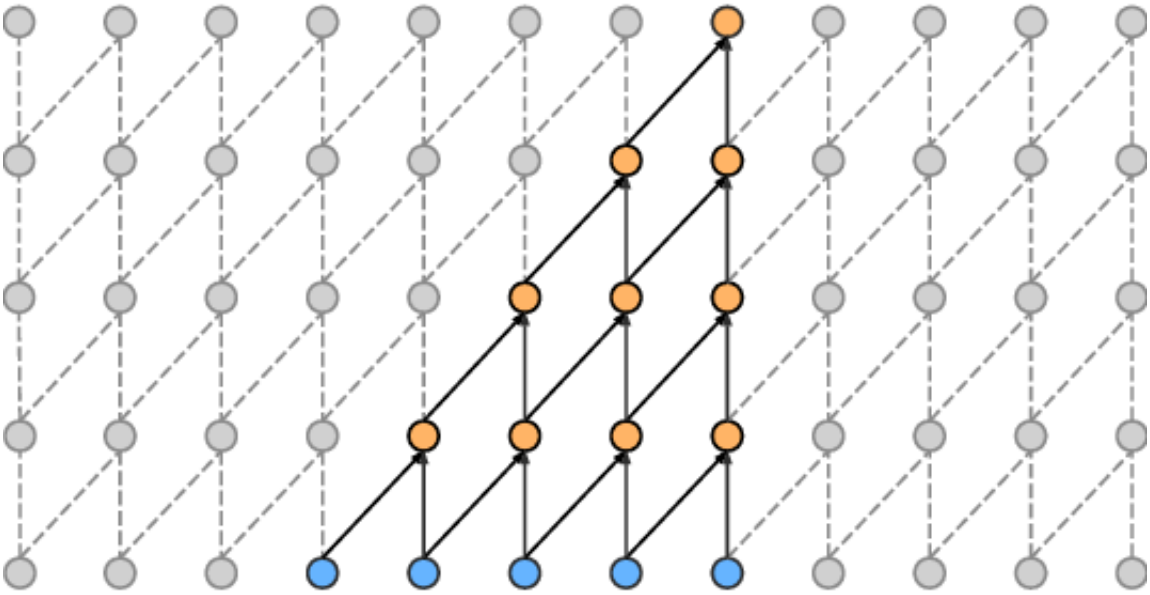

Figure 4.4: An example of causal convolution. The convolution output does not depend on future inputs. From [88].

The Gated Activation Unit (GAU) used for the WaveNet are the same used for the aforementioned PixelCNN and in general take inspiration from the LSTMs layers. It is possible to summarize their functioning through the equation:

$$z = \tanh(W_{f,k} * x) \odot \sigma(W_{g,k} * x) \tag{4.2}$$

where $*$ denotes a convolution operator, $\odot$ denotes an element-wise multiplication operator, $\sigma(\cdot)$ is a sigmoid function, k is the layer index, f and g denote filter and gate, respectively, and W is a learnable convolution filter. A graphical representation of the GAU is represented generically in Fig. 4.3a while a more precise example of the GAU used in the case of WaveNet is shown in Fig. 4.3b. Furthermore, as can always be seen in Fig. 4.3 residual skip connections are used throughout the network, to speed up convergence and enable training of much deeper models. This feature is essential to be able to effectively train such deep networks and avoid the vanishing gradient problem [87]. Moreover, they enable information at each layer to be propagated directly to the final layers. This allows the network to explicitly incorporate features extracted at several hierarchical levels into its final prediction.

The main feature of the generative WaveNet consists of causal convolutions. This particular type of convolution is necessary to prevent the model from violating the order in which the data is modeled. In fact the prediction $p(x_{t+1}|\, x_1, \ldots, x_t)$ emitted by the model at timestep t can not depend on any of the future timesteps $x_{t+1}, x_{t+2}, \ldots, x_T$. The advantage of this approach lies in the fact that it allows to maintain temporal coherence in data generation without having to resort to recurring connections, which

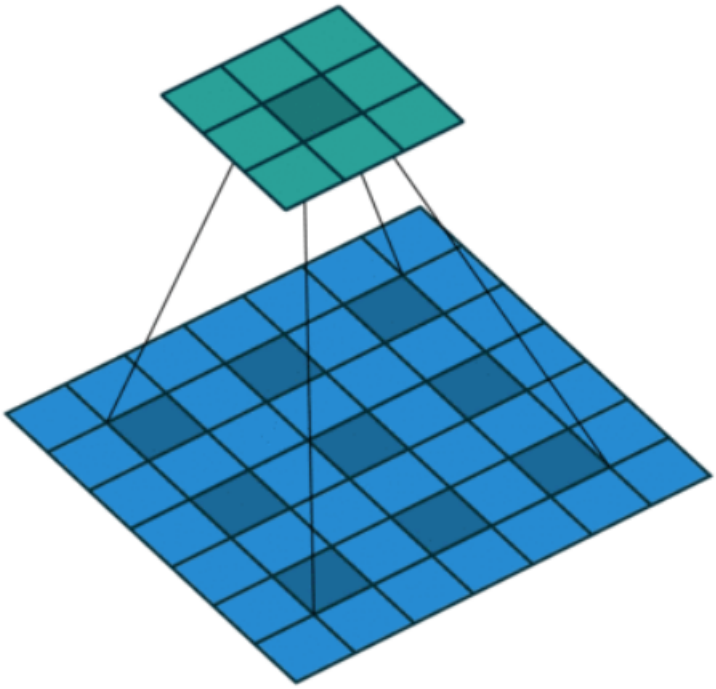

Figure 4.5: Graphical representation of a 2 dilated convolution on 2D input data. From [89].

are very expensive computationally, as is the case for RNNs type structures. To understand this concept graphically, Fig. 4.4 illustrates the principle of causal convolution.

One of the biggest problems of causal convolutions is that they require many layers, or large filters to increase the receptive field. For this reason, in the case of the generative WaveNet, common convolutions have not been used but dilated convolutions to increase the receptive field by orders of magnitude, without greatly increasing computational cost. A dilated convolution (or convolution with holes) is a convolution where the filter is applied over an area larger than its length by skipping input values with a certain step. It is equivalent to a convolution with a larger filter derived from the original filter by dilating it with zeros, but is significantly more efficient. The principle is similar to pooling or strided convolutions, but in this case the output has the same size as the input. As a special case, dilated convolution with dilation 1 yields the standard convolution. Intuitively, in the case of 2D input data, a dilated convolution looks like in Fig. 4.5. Characteristic feature of the WaveNet configuration is that the dilation is not fixed but it is doubled for every layer up to a limit and then repeated e.g.

$$1, 2, 4, \ldots, 512, 1, 2, 4, \ldots, 512, \ldots, 1, 2, 4, \ldots, 512$$

The implications of this configuration is two-fold. The former is that exponentially increasing the dilation factor results in exponential receptive field growth with depth [90], e.g. each 1, 2, 4, ..., 512 block has receptive field of size 1024, and can be seen as

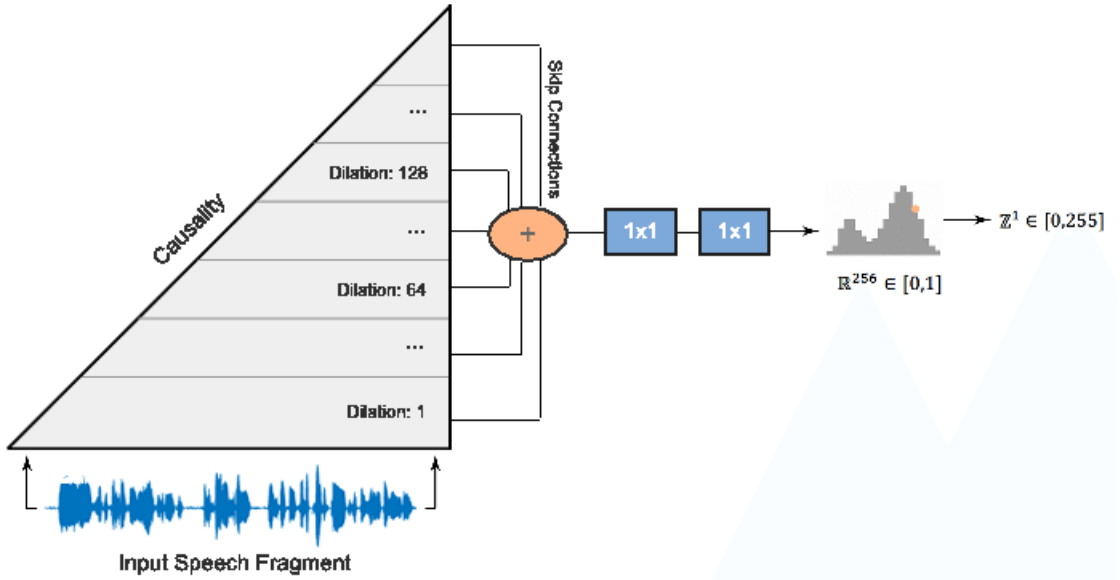

Figure 4.6: Overview of Wavenet. From [72].

a more efficient and discriminative (non-linear) counterpart of a 1×1024 convolution, while the latter involves that stacking these blocks further increases the model capacity and the receptive field size.

A very important final aspect regarding the generative WaveNet concerns the initial data processing. In fact using a discrete softmax output distribution, it is necessary to perform a more coarse 8-bit quantization to make the task computationally tractable, contrary to what it would take to work with 16-bit audio, which is the classic format in which raw audio is presented. In the latter case, the softmax layer would need to output 65,536 probabilities per timestep to model all possible values. Using μ-law companding transformation to data instead, they are quantized to 256 possible values:

$$f(x_t) = \text{sign}(x_t)\frac{\ln(1+\mu|x_t|)}{\ln(1+\mu)} \tag{4.3}$$

As mentioned, the WaveNet architecture was born as a generative model to perform the TTS task and in the light of what has been explained up to now, it can be schematically summarized as in Fig. 4.6.

In [72] instead it is reported how the architecture just presented, applying some modifications, can be adapted to the speech denoising/enhancement task. In fact, speech denoising, while having a lot in common with speech synthesis, has many unique characteristics. Hence the need to adapt the starting architecture to the new task.

First element of substantial difference between the generative WaveNet and the one

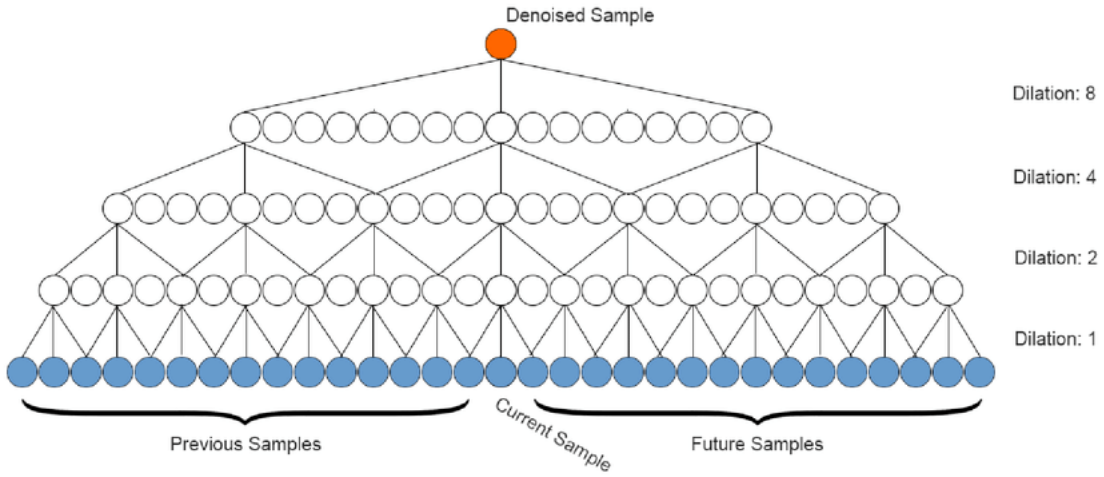

Figure 4.7: Non-causal, dilated convolutions with exponentially increasing dilation factor. From [72].

designed for denoising is that for the latter the convolutions are not causal (visually they appear as in Fig. 4.7). Unlike speech synthesis, in fact, as far as denoising is concerned, future information is usually available: even in the case of real-time applications, the model has access to valuable information on subsequent samples compared to the sample of interest, thanks to the small latency granted to the model to fulfill the task.

Moreover, instead of applying asymmetrical convolutions (length = 2), it was chosen to stretch them up to a length equal to 3 so as to be able to perform a symmetrical padding. In this way, the sample of interest selected by the scrolling of the convolution can benefit from a homogeneous context for both past and future samples.

The ability to predict raw audio, without the need for any kind of preprocessing is the strong point of this architecture. This is made possible by the fact that the WaveNet for speech denoising makes predictions in real values, as opposed to the generative one which returned a discrete softmax output. Also with regard to μ-law quantization, this would prove to be inefficient as it would end up amplifying the background noise. As the output is not explicitly modeling a probability distribution, but rather the output itself, the model can be defined as *discriminative* rather than *generative* and *autoregressive*.

The last structural difference consists in the kernel size of the two final convolutional layers: while in the generative model, being of a regressive nature, the samples had to be submitted to the network to predict subsequent samples, the filters of the final layers were both 1×1. In the case of denoising WaveNet, however, the kernel size of the final layers is 3×1. Graphically, to make a comparison with the generative WaveNet represented in Fig. 4.6, the variant for speech denoising appears as in Fig. 4.8.

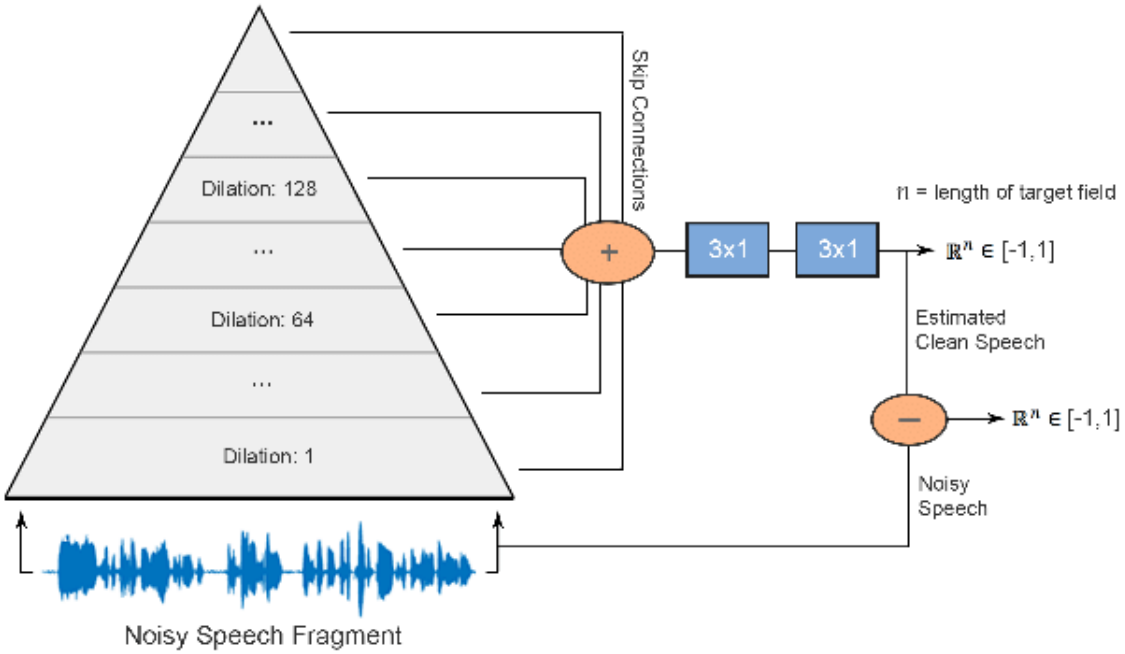

Figure 4.8: Overview of the speech-denoising Wavenet. From [72].

Peculiar elements used in the original work, but not included in this thesis work, are a conditioning of the model based on the speaker identity[5] and the use in the training phase of audio tracks containing only noise[6] to improve the performance of the model.

The last point worth pointing out is the loss functions used to train the model. For the original work, a two-term function was used, called Energy conserving loss. This loss function takes into account not only how close the processed audio is to the natively noise-free audio, but also how close the estimated noise is to the original noise. Mathematically this loss function can be expressed as follows:

$$\mathcal{L}(\hat{s}_t) = |s_t - \hat{s}_t| + |b_t - \hat{b}_t| \tag{4.4}$$

with $\hat{s}_t$ representing the enhanced audio, s_t the clean audio, b_t the noise summed to the clean audio in order to make the noisy audio and finally $\hat{b}_t$ the estimated noise. In this way, the network predicts both components of the signal and, since the second output is produced by a parameterless operation on the speech estimate, the loss produced by this output is directly representative of the models ability to perform the speech

[5]The condition value is the bias term in every convolution operation. In addition, an auxiliary code (all zeros) has been added denoting any speaker identity so that the trained model can be used for unknown speakers. This conditioning procedure can also be interpreted as a data augmentation strategy.

[6]Training samples contain only background-noise was also employed after observing that the model had difficulties producing silence.

enhancement task since $\hat{b}_t = m_t - \hat{s}_t$ with $m_t = s_t + b_t$. From this relation it can in fact be deduced that $\hat{m}_t = \hat{s}_t + \hat{b}_t = \hat{s}_t + (m_t - \hat{s}_t) = m_t \rightarrow \hat{m}_t = m_t$. Precisely for this reason, using this method to obtain the estimated noise leads the loss to conserve the total energy of the audio $E_{\hat{m}_t} = E_{m_t}$. As can be seen from Eq. 4.4, the two terms that make up the loss function are basically two simple L1 losses[7].

This section represents the presentation of the architecture whose purpose is to perform the denoising task of the input audio. Details regarding the technical implementation of the aforementioned network are available in Sec. 4.3.

4.2.2 ResNet

The second component of the pipeline considered during the thesis work is a further architecture, whose task is to establish the identity of the speaker in an audio signal. In particular, this network is trained with the task of identifying the person who is speaking in an audio track but, once trained, it can be used to verify, starting from two distinct audio tracks, if the person who is speaking in these two audio tracks is the same. Therefore, using more specific terms, while the first task described corresponds to that of speaker recognition, the second, more generic, is called speaker verification. More exhaustive explanations regarding the two tasks have already been reported in Sec. 2.3.2.

The architecture typically used to perform the classification task[8], whatever the nature of the input, consists in taking as input a certain representation of it, be it an image, an audio signal or its transformation, and proceed to process it until it reaches a dense and compressed representation: this representation is called embedding [91, 92, 93]. While in the past these embeddings were trained using classification losses, lately there is a tendency to use softmax classifiers. The reference work used to cover this part of the thesis is presented in [94].

The main focus of this work is not the architecture itself[9], but rather to verify which is the best loss function among a series of functions selected to train the same architecture under the same conditions. Since the model used in this thesis derives directly from [94], and has been recovered in a pre-trained form (i.e. already subjected to a training procedure), a very limited focus on the particularity of the loss functions used will be made while greater importance will be given on model building. In fact,

[7] L1 Loss Function is used to minimize the error which is the sum of the all the absolute differences between the true value and the predicted value. Mathematically it can be expressed as L1 loss = $\sum_{i=1}^{n} |y_{\text{true}} - y_{\text{predicted}}|$.

[8] Speaker recognition, since it is essentially a question of classifying who is speaking can be considered a special case of the classification task.

[9] As mentioned, in fact, for the speaker recognition task, at an architectural level, the NNs in literature have more or less all the same structure. What usually varies are the embedding and pooling mechanisms, the loss functions and the dataset used to perform the training in addition to the preprocessing applied to the input data.

since the network was conceived with the intention of operating on a test set of speakers that it has never seen during the training phase (speaker verification), this poses the problem in the field of open set setting and makes it a metric learning problem in which entries must be mapped to a representative embedding space.

Regarding the main focus of the work taken as a reference for the use of this neural network, the loss functions examined are seven:

- **Softmax**: a softmax function followed by a multi-class cross-entropy loss;
- **AM-Softmax** [95, 96]: softmax loss but normalizing the weights and the input vectors, in this way the posterior probability only relies on cosine of angle between them;
- **AAM-Softmax** [97]: equivalent to AM-Softmax except that there is additive angular margin penalty;
- **Triplet** [98]: triplet loss minimises the distance between an anchor and a positive (same identity), and maximises the distance between an anchor and a negative (different identity);
- **GE2E** [99]: every utterance in the batch except the query itself is used to form centroids i.e. the centroid that is of the same class as the query is computed from one fewer utterance than centroids of other classes;
- **Prototypical** [100]: each mini-batch contains a support set S and a query set Q. Squared Euclidean distance is used as the distance metric. During training, each query example is classified against N speakers based on a softmax over distances to each speaker prototype;
- **Angular Prototypical**: this is the loss function that has obtained the best result in the work taken as a reference and is therefore the one used to train the pre-trained model that was used for this thesis work. Its formulation, shown in Eq. 4.5, is the same as that of the Prototypical loss but differs in the way $S_{j,k}$ is calculated:

$$L_P = -\frac{1}{N}\sum_{j=1}^{N}\log\frac{e^{S_{j,j}}}{\sum_{k=1}^{N}e^{S_{j,k}}} \tag{4.5}$$

  It is also worth pointing out that this function is an original realization by the authors of [94].

The experimental setup of this work also involved the use of different architectures compared in terms of performances for the speaker recognition task. Among these architectures there are:

- **VGG-M-40** [101];

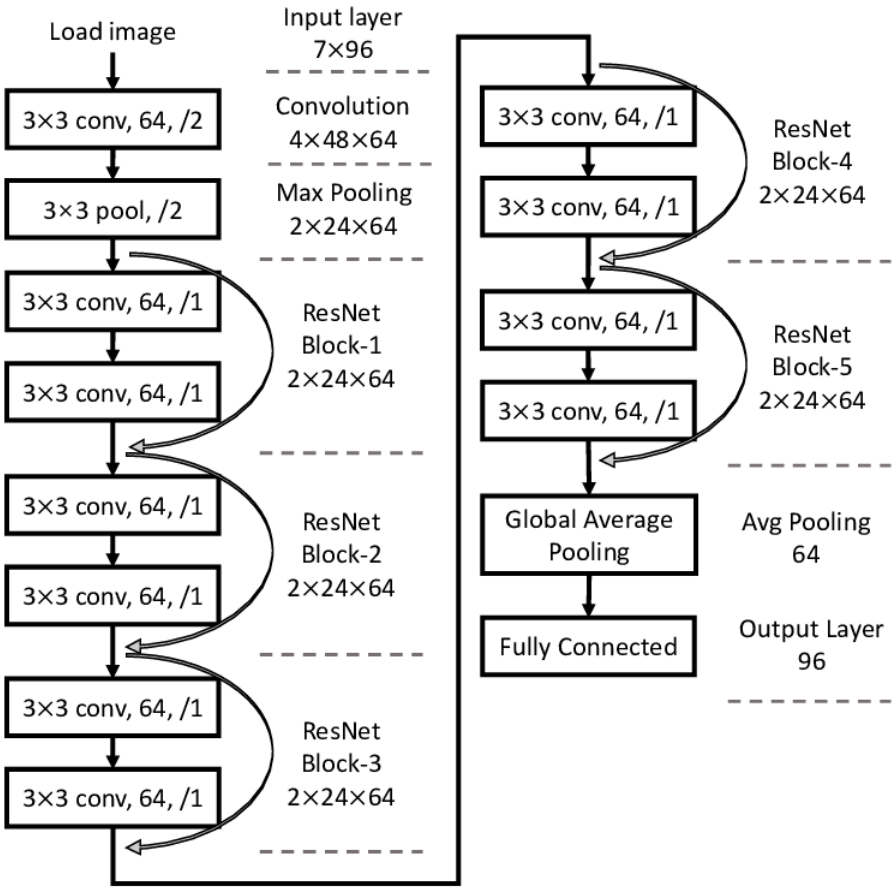

Figure 4.9: The structure of ResNet-12. From [104].

- **Thin ResNet-34** [102, 103];
- **Fast ResNet-34** [94].

among which the Thin ResNet-34 achieved the best results in terms of accuracy in the classification of the speakers. For this reason, the pre-trained model found online[10] is precisely a Thin ResNet-34 network trained using an Angular Prototypical as a loss function: this architecture will be presented below to fully understand the structure and processes which the network input is submitted.

ResNet architecture is a very widespread NN in the field of image recognition. Only recently, in fact, its capabilities have been explored for tasks such as speaker recognition, applied to transformations of audio signals that bring them into the time-frequency domain, taking the data from a one-dimensional to two-dimensional format.

Going in chronological order, the first ResNet, contract name which stands for Residual Network, was composed of 152 convolutional layers. The only mechanism applicable to avoid the vanishing gradient problem in the presence of such a deep architecture have turned out to be the skip connections, designed and built precisely

[10]The pre-trained Thin ResNet-34 model and the implementation instructions are given at the link `https://github.com/clovaai/voxceleb_trainer`.

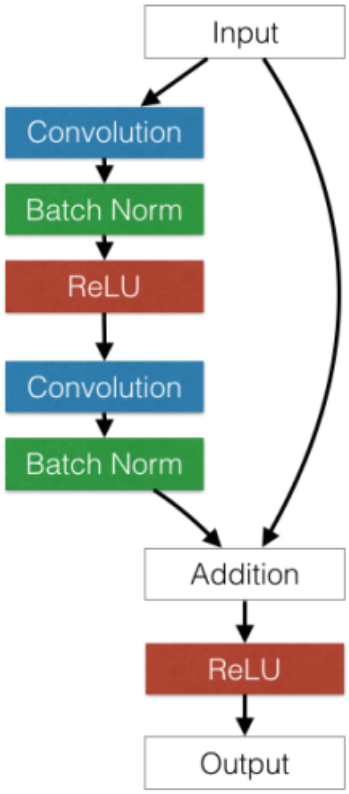

Figure 4.10: Example of ResNet Residual block.

for the invention of this particular type of architecture. With time and the progress in the field of DL of a certain current of thought aimed at optimizing the NNs in terms of computational heaviness, very smaller versions of the ResNet have been created, such as the ResNet-50 and the aforementioned ResNet-34. For example, in Fig. 4.9 a ResNet-12 is schematically represented. This image has a double utility: on the one hand it allows to realize what the general structure implemented in a ResNet architecture is, while on the other it is the representation of how this network, starting from extreme dimensions, over time and depending on the task, has been heavily revised to find a better balance between performance and computational resources.

As it is possible to observe in Fig. 4.9, the network is substantially formed by a set of blocks, delimited by the points where the skip connections take place, called *residual blocks*. It is precisely the existence of these blocks that helps to avoid the possibility of a too strong reduction or an explosion of the gradient occurring, since during the back-propagation the gradient in addition to flowing along the main structure also flows through the skip connections, thus skipping the blocks. It is worth noticing that Fig. 4.10 shows an example of residual block: these blocks are often composed of different convolutional layers, followed by other operations such as batch normalization [105] or pooling (or both) and finally the activation function.

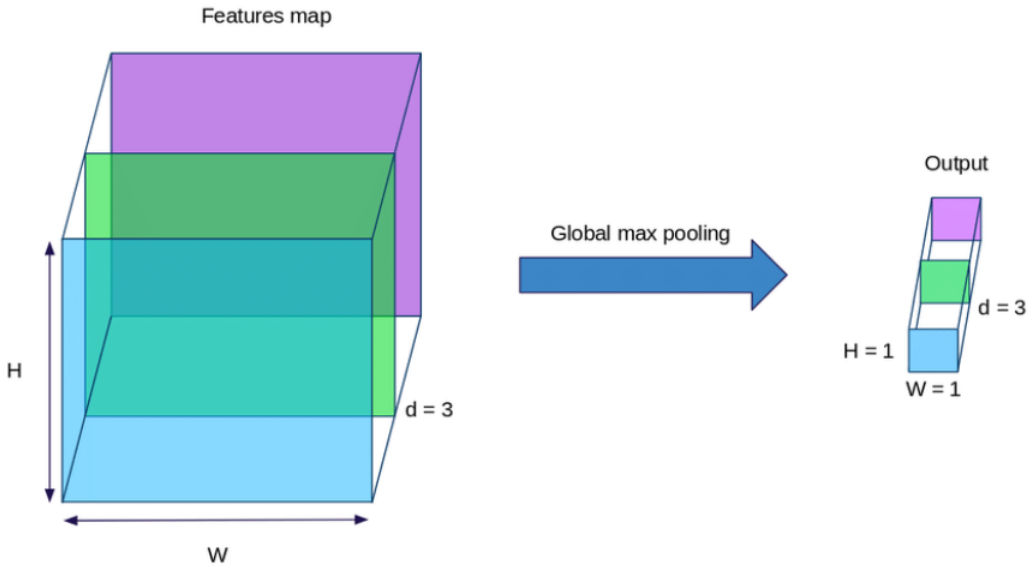

Figure 4.11: Scheme of the global max pooling mechanism. Global max pooling perform an operation that take the maximum per features map and produce an 1D vector with known size (number of features map). From [106].

The second distinctive element of ResNet networks is the final pooling mechanism. As previously said, the architecture proceeds to compress the initial input by extracting increasingly complex features as the input reaches the deeper layers of the network. In order to extract very complex features, convolutional layers must in fact proceed to apply many filters, arriving at very complex multi-dimensional representations. The pooling mechanism, in this sense called global pooling, "flattens" the multi-dimensional representation. This step is essential in order to achieve a one-dimensional representation. It is also sometimes used in models as an alternative to using a fully connected layer to transition from feature maps to an output prediction for the model. In fact, in order to proceed with the classification task, the generic ResNet architecture is composed of several fully connected layers following the global pooling layer, which are then necessary to achieve the correct dimensionality based on the classification. As in the case of classic pooling layers, global pooling is usually distinguished in global average pooling and global max pooling. To better understand the global pooling mechanism, Fig. 4.11 graphically illustrates the global max pooling process.

In the case of the architecture built in [94], the encoding layer is slightly more complex than those usually used: it is in fact a Self-Attemptive Pooling (SAP) layer [102]. The introduction of SAP layer arises from the necessity to pay attention to the frames that are important to the classification[11], and aggregate those informative frames to form an utterance level representation. This SAP layer implementation is

[11] In fact not all frame of features contribute equally to the utterance level representation.

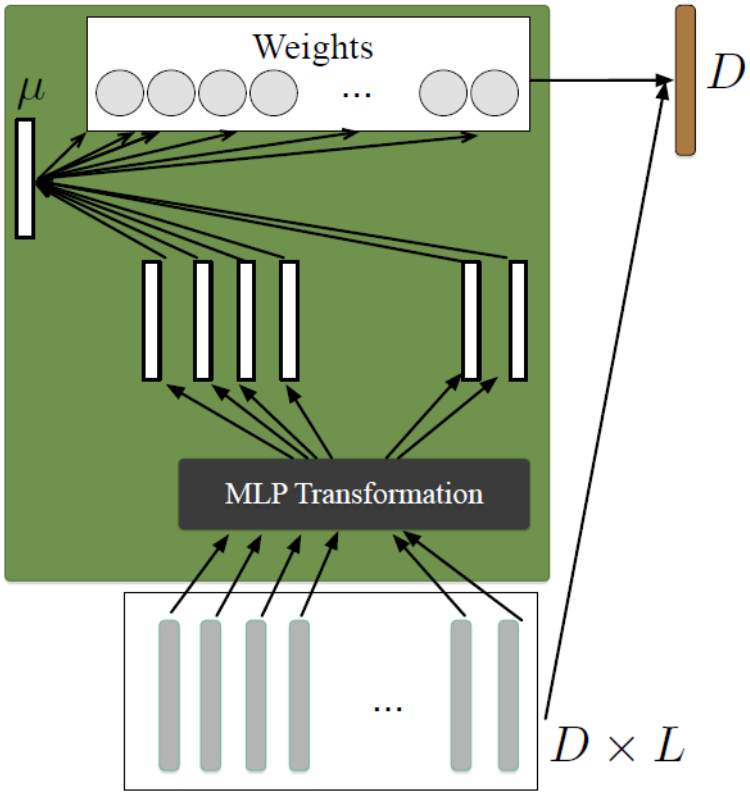

Figure 4.12: Self-Attemptive Pooling (SAP) layer mechanism representation. D denotes the feature coefficients dimension, and L denotes the temporal duration length. From [102].

similar to [107, 108, 109]. The first step consists in submitting the utterance level feature maps $\{x_1, x_2, \ldots, x_L\}$ to a multi-layer perceptron to obtain the hidden representations $\{h_1, h_2, \ldots, h_L\}$. In the case of a multi-layer perceptron with a single layer the process for obtaining the hidden representation is explained by the equation:

$$h_t = \tanh(W x_t + b) \tag{4.6}$$

At this point it is necessary to measure the importance of each frame. To do this, importance is measured as the similarity of h_t with a trainable context vector μ[12] (it is randomly initialized and jointly learned during the training process). Subsequently, a normalized weight of importance w_t is obtained through a softmax function.

$$w_t = \frac{e^{(h_t^T u)}}{\sum_{t=1}^{T} e^{(h_t^T u)}} \tag{4.7}$$

Finally, the utterance level representation e can be computed as a weighted sum of

[12]It can be interpreted as a high level representation of a fixed query "what is the informative frame over the whole frames".

the frame level CNN feature maps based on the learned weights.

$$e = \sum_{t=1}^{T} w_t x_t \tag{4.8}$$

In Fig. 4.12 it is possible to observe a schematic representation of the functioning of the SAP layer used.

4.3 Technical implementation

After having extensively discussed the structure of both architectures implemented (WaveNet and ResNet), it is now worthwhile to focus on the implementation details put in place during the thesis work. The results achieved will not be discussed in this section but will be extensively presented in Sec. 5.

All the technical implementation work was done using Python[13] as the programming language. As an implementation framework for neural architectures instead, PyTorch [110] was used. The reason why this particular framework was chosen rather than others, such as Tensorflow [111] or Keras [112] for example, is that it is widely used in academia for the analysis of audio signals or in computer vision. Furthermore, its data structures (tensors) and its high compatibility with training via GPU allow to effectively optimize the computational capacity of the machine used. Finally PyTorch, unlike other frameworks or APIs, leaves a lot of customization freedom in the construction of DL architectures and in the learning mechanism: this feature makes it very flexible and understandable thanks to its "python like" grammar.

4.3.1 Data processing

The datasets used have been extensively described in Sec. 4.1. The objective of this section is instead to explain the purpose that each dataset covers within the work pipeline and the manipulation processes to which they have been subjected. In fact, despite not having been subjected to "heavy" pre-processing operations, it was necessary to slightly manipulate the data to adapt them to the task of the thesis work.

First of all, since not all data sources originally provided the same sampling rate for the audio tracks, these have been resampled to 16kHz. Furthermore, a characteristic of all data sources is that the audios were already presented in a mono-channel format, so no intervention was necessary from this point of view. Finally, among the preliminary operations, it was arbitrarily chosen to immediately exclude all audio tracks with a duration strictly less than four seconds. Within the various datasets there were in fact

[13] Python Software Foundation, `https://www.python.org/`.

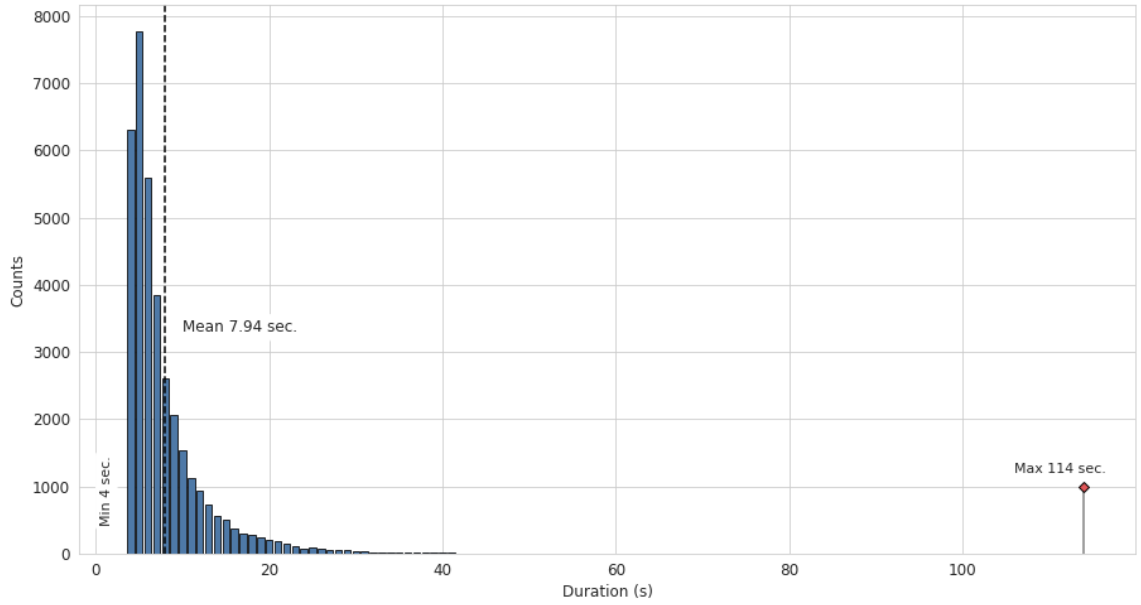

Figure 4.13: Duration distribution for the portion of VoxCeleb2 test set used as training and test set for the realized system. Minimum, maximum and mean durations are annotated.

audio tracks even of a duration of just under one second but it was decided to remove them and thus obtain more significant audio signals.

Usually, when working with speech audios, preprocessing techniques are applied to pre-emphasize[14] [113] the traits of the signal in which the speech is contained. Voice Activity Detection (VAD)[15] techniques [114] are also often applied. In the case of this thesis work, in accordance with the aim of avoiding preprocessing operations on the incoming audio as much as possible, these techniques have not been applied.

To begin with, it must first be specified that for the actual training of the overall system (the serial combination of the WaveNet and the pre-trained ResNet), the SIWIS dataset and a portion of the VoxCeleb2 test-set (about 1/3) were essentially used, together with the MUSAN dataset. These three datasets represent two clean audio datasets and a noise source. The selection of the first two essentially took place in order to have an extremely clean audio source i.e. the SIWIS dataset to which noises of different nature can be added, and a more "dirty" audio source, as is the case of the VoxCeleb2 dataset which as explained in Sec. 4.1.3 is essentially composed of videos collected online containing interviews with famous people.

[14] Pre-emphasis refers to a process designed to increase (within a frequency band) the magnitude of higher frequencies with respect to the magnitude of lower frequencies (it was noted that higher frequencies are more important for signal disambiguation than lower frequencies) in order to improve the overall SNR: this process takes place by applying a calibrated filter to the signal.

[15] VAD refers to the task of determining whether a signal contains speech or not. It is thus a binary decision. There are several techniques with which to apply this type of process: energy thresholding, analyze different properties of the signal by extracting features, real machine learning techniques.

Table 4.4: Train test split for SIWIS and VoxCeleb2 portion used for trainig the whole system.

Dataset	**Train set split**	**Test set split**
SIWIS	90 %	10 %
VoxCeleb2	70 %	30 %

In Fig. 4.13 the distribution of the duration of the audio signals in the portion of the test set of VoxCeleb2 used to build the training and the test set to train the system can be observed. The distribution shown refers to the data after excluding audios with a duration of less than four seconds.

At this point, before proceeding with the subsequent operations, both datasets have been divided into training and test sets according to the proportions shown in Tab. 4.4, using the stratified sampling technique[16], in this case based on the speakers. In this way, in both portions all the speakers are present with a proportional quantity of audio recordings for each of them.

The mechanism of adding noise to the audios that make up the different datasets took place in two distinct ways. As for the SIWIS dataset, since it is a rather small database, for each audio track was chosen to randomly select between the different types of noise present in the MUSAN dataset (three distinct types, namely music, speech and noise: see Sec. 4.1.2) and add them individually. In this way, from each audio originally present in the SIWIS dataset three have been obtained, each characterized by a different type of background noise randomly extracted from those available. This choice was dictated not only by the need to implement a pseudo data augmentation mechanism but also by the need to maximize the variety of background noises present since the initial purpose is to train a system that is robust with respect to the noises present in urban environments in everyday life.

As for the VoxCeleb2 portion, on the other hand, this augmentation mechanism was not necessary, since it is already a rather large database. For this reason, a randomly chosen noise from the three types listed above was added to each audio, in order to obtain the corresponding "dirty" audio from each clean audio with a $1:1$ ratio.

As for the intensity of the noise added to the different audio signals, this was chosen randomly too. In fact, for each audio it was decided to proceed by selecting the audio track of noise to be added, and to add it by randomly selecting among four distinct SNRs values (see Sec. 2.3.4). For both the training and test samples these were synthetically mixed with one of the following SNRs: 2.5, 7.5, 12.5 and 17.5 dB. After adding the background noise to the data, their distribution in the train and test

[16]Stratification is the process of dividing members of the population into homogeneous subgroups before sampling.

Table 4.5: Audio distribution before oversampling.

Dataset	Portion	n. of samples	avg. samples per speaker
SIWIS	train	7,344	~ 333
	test	816	~ 37
VoxCeleb2	train	8,448	~ 16
	test	2,113	~ 4
SIWIS + VoxCeleb2	train	15,792	~ 30
	test	2,929	~ 6

Figure 4.14: The KALDI toolkit and PyTorch framework logos.

portions is shown in Tab. 4.5.

From a technical point of view, in order to optimize the use of the computational resources, the processed audios have been saved, instead of adding them to the background noise in a dynamic way. Furthermore, the processed audio has been saved in pairs with the corresponding audio free of background noise, so as to create $\{X, Y\}$ pairs of inputs and ground truth, already in the form of tensors. To save these pairs of tensors, it was decided to use a data structure of the KALDI toolkit[17] [115], namely the ark, scp structure. The terms ark and scp represent the terms archive and script file respectively. In this sense, during the data processing, the pairs of tensors were saved in a compressed ark file so that the amount of memory used was minimized. The scp file, on the other hand, essentially corresponds to a text file, in which in each line there is an identifier of the audio pair (i.e. the audio added to the noise and the noiseless counterpart) and the path that leads to the ark file in which that pair of is contained. In this way, what is loaded into memory when defining the PyTorch Dataset[18] class are essentially lists of strings and not directly the tensors, which are instead dynamically imported from the ark file when the Data Loader creates the batches. By exploiting this mechanism, the use of memory is therefore optimized. In addition, two further

[17] All the project documentation is available at `https://kaldi-asr.org/doc/`.

[18] The explaination of what are Datasets and Data Loader classes in PyTorch grammar is available at `https://pytorch.org/tutorials/beginner/data_loading_tutorial.html` with a small tutorial for beginners.

Table 4.6: Audios distribution after oversampling.

Dataset	Portion	n. of samples	n. of samples per speaker
SIWIS	train	7,344	247
	test	816	247
VoxCeleb2	train	9,027	74
	test	2,287	74
SIWIS + VoxCeleb2	train	25,334	247
	test	5,482	247

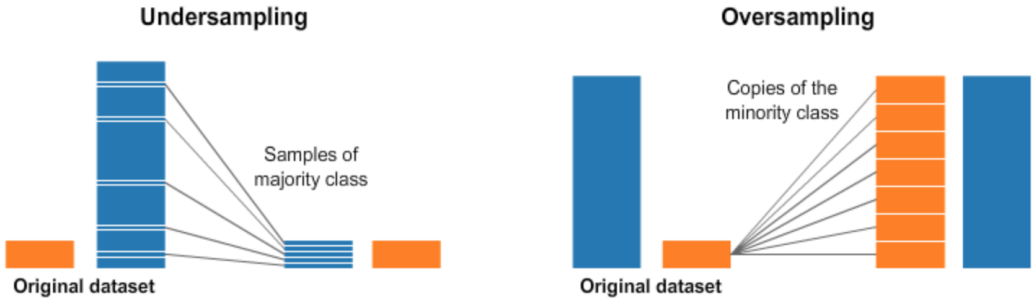

Figure 4.15: Schematic representation of the operation of the Undersampling (left) and Oversampling (right) techniques.

precautions were taken during the construction of the Dataset class.

First of all, to preserve the variety of speakers involved in the various datasets, it was decided to rebalance the number of audios per speaker, calculating the maximum number of audios for each speaker at the time of defining the data loader. Subsequently, for each speaker, it was decided to proceed by replicating the audios several times in order to ultimately obtain that all the speakers had the same number of audio tracks (with a difference of $\sim 1/2$ tracks). Using technical language, this type of procedure is called *oversampling*[19]. The concepts of oversampling and its opposite, undersampling, are represented graphically in Fig. 4.15 while the data distribution after the oversampling operation regarding train and test set is shown in Tab. 4.6. At first glance, this type of intervention might seem counterproductive: the audio signals were in fact replicated by creating real clones that could lead the model to overfit the training data. For this reason, the second precaution was applied.

As it can be observed in fact in Fig. 4.1 for the SIWIS dataset and in Fig. 4.13 for VoxCeleb2, the duration of the audio signals is very variable, moreover, as mentioned before, all audios with a duration strictly less than four seconds have been excluded regardless of the training set and the test set. Since the target length of the audios

[19]Oversampling (and its counterpart, that is, undersampling) is defined as a statistical technique used to adjust the class distribution in a dataset.

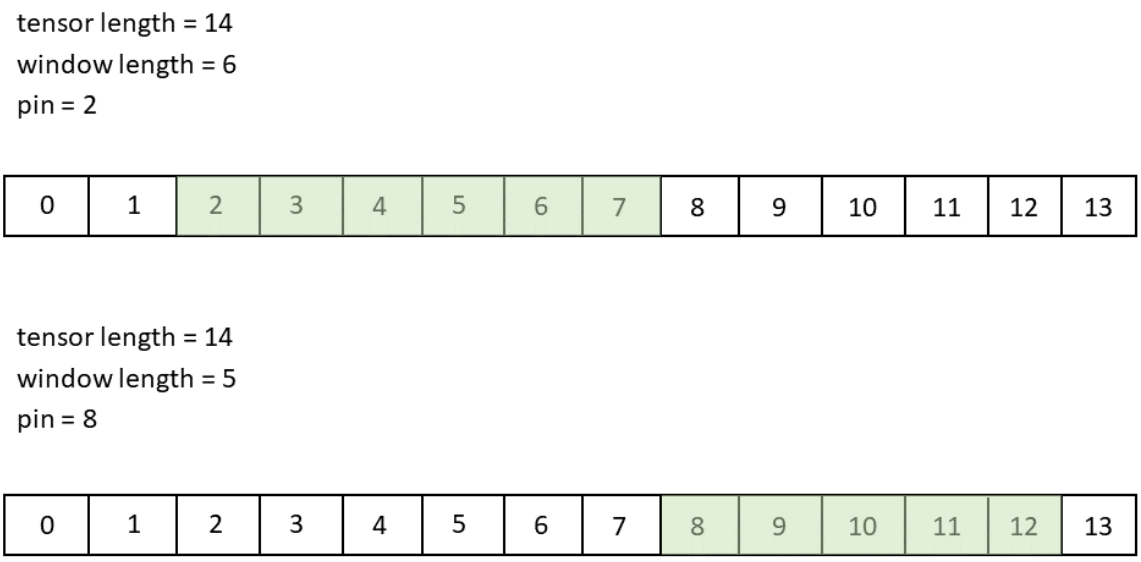

Figure 4.16: Examples of windowing techniques. The pin value is randomly chosen between 0 and tensor_length − window_length at each call of the batch by the data loader.

used for training is one second, a sliding window has been created, so that at each call of the data this window selects a random second from each audio. The windowing mechanism is intuitively presented as shown in Fig. 4.16. With this solution, an almost unlimited augmentation mechanism is created. In fact, at each training epoch, the audios presented to the network are never exactly the same, they are part of the same audio track, but at different instants: the only constant value is represented by the duration which, as mentioned, has been chosen to be one second. Therefore, considering the chosen sampling rate of 16 kHz, each tensor that makes up the training and test set will have a dimensionality 1×16000, that is 1 channel and 16000 samples.

Regarding the other datasets, however, they have been used for different purposes. The DEMAND dataset, illustrated in Sec. 4.1.4, was used to evaluate the performance in terms of pure denoising, in accordance with what happens in the state of the art (see Sec. 3.2) regarding speech enhancement models. As for VoxCeleb1 and the training portion of VoxCeleb2, these play different roles. While the test portion of VoxCeleb1 is used as an evaluation set of the pipeline performance as regards the speaker verification task, the train portions of both VoxCeleb1 and VoxCeleb2 were used by the authors of [94] to train the SV model. For this reason it was decided not to use them in any of the training phases of the overall system, in order to avoid the results being influenced by the fact that the pre-trained model had already seen this particular portion of the data. Furthermore, since the independence of the speakers is guaranteed between the train and test portions of VoxCeleb1 and 2, in this way the system does not risk being influenced even by the identities of the speakers, authors of the audios used for

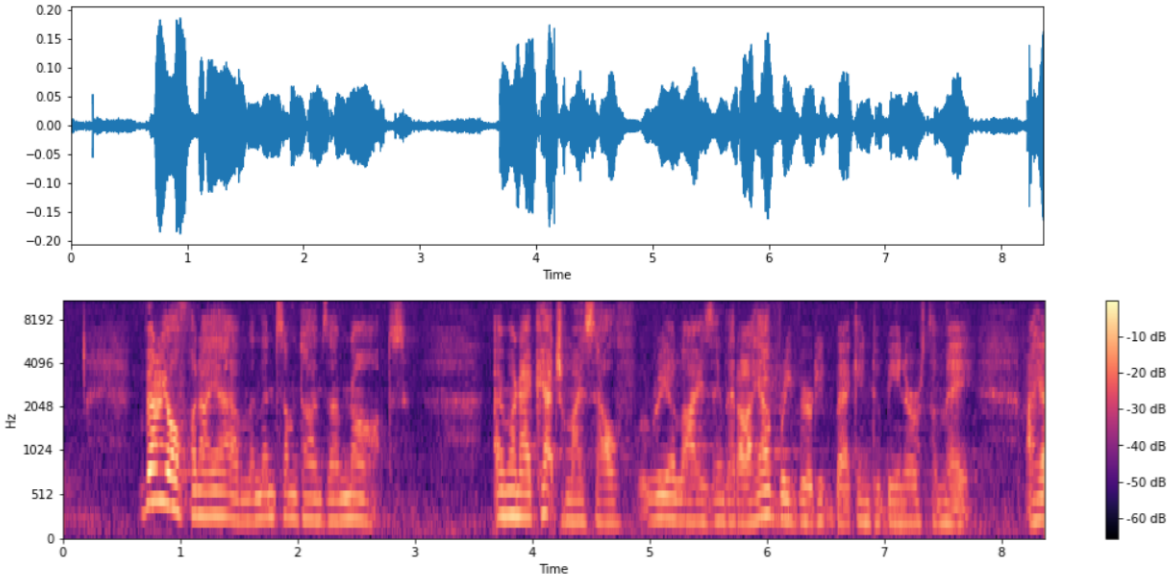

Figure 4.17: Representation of a speech audio (above) with the corresponding Mel Spectrogram (below) whose scale has been converted from power to dB.

training.

As pointed out in Sec. 4.2.1, the data subjected to the WaveNet architecture does not require special transformations or processing operations. This however is not the case with ResNet. In fact, since this is a network designed for 2D inputs, as widely emphasized in Sec.4.2.2, the input needs a transformation that allows to represent some useful features to characterize the voice of the speaker in 2D form. In this regard, many features that can be extracted from speech audios in Sec. 2.2 have been presented. In the case of the implemented pre-trained model, the transformation used is the Mel spectrogram (for details see Sec. 2.2.3). In particular in Fig. 4.17 it is possible to observe an example of eight-second audio paired with the corresponding Mel spectrogram, to which a transformation on the chromatic scale from power to dB was subsequently applied (thus obtaining a sort of logarithmic scale for the color).

4.3.2 Experimental setup

As previously mentioned, the system is made up of two distinct NNs: a WaveNet in charge of carrying out the denoising task and a ResNet, which fulfills the speaker recognition task. These networks manipulate one-dimensional and two-dimensional inputs, respectively. Despite this, the particular way in which the ResNet was written (always using the PyTorch classes), allows to provide a one-dimensional input, since the transformation of the latter takes place directly within the network. This feature allows to place the two models one in line with the other without particular precautions

in terms of implementation through code. In ML, a practice that comes very close to that used is called *stacking*[20] [116].

Regarding the WaveNet, it was decided to implement it using the configuration proposed by the authors of [72], although it is rather heavy computationally. The network thus proposed by the authors is made up of a total of 30 GAU convolutional blocks. The dilation factor in each block increases with a power of 2 in a range that goes from a minimum of 1 to a maximum of 512 (i.e. 1, 2, . . . , 256, 512). This pattern is repeated 3 times, ideally constituting 3 stacks. These blocks are characterized by the constant use of 128 filters each, a kernel size 3×1 and the use of dropouts[21] with $p = 0.05$.

The 30 GAU convolutional blocks are preceded by an initial convolutional layer that projects the one-dimensional input in 128 channels with a standard, undilated convolution, with a filter size 3×1, to align with the number of filters of each residual layer. The skip connections consist of convolutions with a 1×1 kernel size, always with 128 channels. A ReLU[22] [117] activation function is then applied to the final sum of all skip connections.

At the end of the block of 30 GAUs divided into 3 stacks, there are still two non-dilated convolutional layers, separated by a ReLU function, always with kernel size 3×1, which project at 2048 and 256 channels respectively. Finally the network ends with an output layer that projects the feature map from 256 to 1 channel, using a 1×1 filter, in order to obtain a single channel signal again.

With this setup the network overall settles on about 6.3 million parameters. Being a convolutional network, the training was performed on the GPU [118] to speed up the computing operations. As anticipated, the input has a 1×16000 dimensionality and also the output audio preserves the same size. To avoid memory overload and to allow training and evaluation even on a single GPU, it was decided to use a batch size of 8, in order to create tensors that can be managed by a single NVIDIA P100 with a 16 Gb memory (also available using free services such as Gooble Colaboratory[23] [119]). The loss function used is a composite function, created by the authors of [72], called Energy conserving loss (see Sec. 4.2.1). The last relevant element regarding the training setup is the optimizer: it is in fact a simple Adam optimizer [120]. This optimizer, however,

[20] Stacking is a way to ensemble multiple classifications or regression model. Stacked generalization is an ensemble method where a new model learns how to best combine the predictions from multiple existing models.

[21] Use the dropout means that during training randomly zeroes some of the elements of the input tensor with probability p using samples from a Bernoulli distribution.

[22] The Rectified Linear Unit activation function (ReLU) is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of NNs because a model that uses it is easier to train and often achieves better performance.

[23] The service presentation demo is available at `https://colab.research.google.com/notebooks/intro.ipynb`.

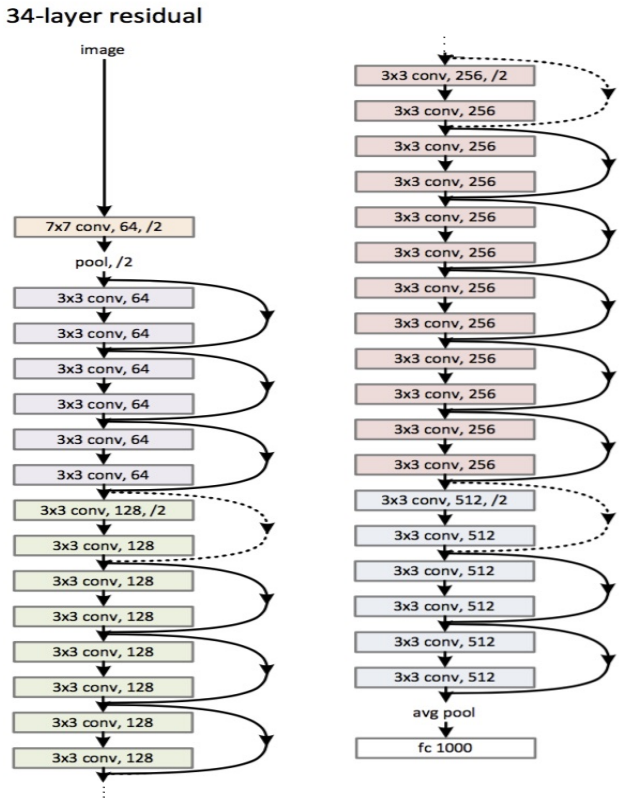

Figure 4.18: Representation of the ResNet-34 architecture.

is supported by a learning rate scheduler[24], so as to update the value of the learning rate at each training epoch following an exponential approach. The choice to use a scheduler was dictated by the need to dynamically adapt the learning rate of the optimizer in order to facilitate the learning of the neural network as much as possible.

With reference to Sec. 4.3.1, it has been specified that three distinct datasets have been created: SIWIS, VoxCeleb2 and SIWIS+VoxCeleb2. The choice to create these three distinct datasets is due to the desire to create three distinct WaveNet models, each one trained on one of these datasets. This choice is motivated by the desire to test the impact on training due to the nature of the starting data. As already mentioned, SIWIS represents an example of a very clean dataset, being recorded under controlled conditions, while VoxCeleb2 is a case of a dataset that is already "dirty" at the start, to which the background noise was obviously added. The last option then, the union between the two datasets, aims to verify the hypothetical positive impact on training due to such an important variety of audio of different nature. All the implementation details regarding the three models are the same: the only variant is the training and validation dataset.

[24]Learning rate schedules seek to adjust the learning rate during training by reducing it according to a pre-defined schedule. Common learning rate schedules include time-based decay, step decay and exponential decay.

Instead, as regards the pre-trained ResNet model as anticipated, it is in detail the ResNet-34 (of which it is possible to graphically observe an example of structure in Fig. 4.18), in a lighter version, called Thin ResNet-34. In practice, the structure does not change compared to the original model, but rather the quantity of filters applied by each 2D convolutional layer. In particular, the Thin ResNet-34 compared to the normal counterpart, has a quarter of the filters: for this reason, while the minimum and maximum number of filters in the original architecture are respectively 64 and 512, as regards the Thin version they are 16 and 128. Reducing the number of filters plays a fundamental role in lightening the network, making it computationally lighter for the machine on which it is trained. While the ResNet-34 has ∼22 million parameters, the lightest version comes in at just 1.4 million.

Other features already mentioned concern the global pooling layer, which in the case of the Thin ResNet-34 used is a SAP layer, and the final dimensionality of the fully connected layer which determines the size of the output, which the authors of the architecture have identified in 512. Given this dimensionality, specified by the last fully connected layer, the network, therefore, regardless of the incoming audio, will output a 1×512 vector. Ideally, in a multi-dimensional space, this vector should be positioned close to other vectors generated from audio recorded by the same speaker. In this sense the network is able to perform both SR and SV tasks. With an additional fully connected layer of dimensionality equal to the number of identities contained in the training set, it is able to establish who is the speaker in the incoming audio (this is the mechanism applied by the authors for training the network, so that this specialize in extracting features that can identify the speaker). In evaluation mode, on the other hand, assuming that the identities of the speakers are unknown (and therefore these are not present in the original training set), remaining at the embedding layer it is possible to calculate the distance between two compressed representations to verify if the speaker in two different audios are the same.

As already mentioned in Sec. 4.3.1, the system is trained on the entire train portion of VoxCeleb2, while its performance is evaluated on the test portion of VoxCeleb1 (for details see Sec. 4.1.3). The pre-trained model used was trained for $\sim$ 500 epochs using an Adam optimizer with an initial learning rate of 0.001 and programmed to decrease by 5% every 10 epochs. As anticipated in Sec. 4.2.2, the loss function used for training is the Angular Prototypical, calculated on normalized tensors. It is also worth noting that even in the ResNet case no VAD techniques were applied and that no data augmentation mechanisms were implemented.

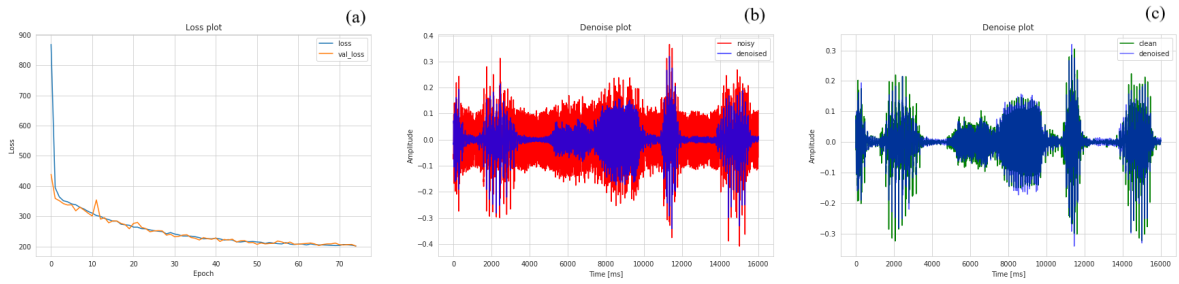

Figure 4.19: Plot composed summary of the performances printed at each epoch. It is possible to observe the trend of the loss functions in training and validation (a), the comparison between enhanced audio and noisy audio (blue and red respectively) in (b), while in (c) it is possible to observe the comparison between enhanced audio and ground truth (respectively blue and green). The audio used is a section of a second belonging to an audio chosen randomly from the test set.

4.3.3 Pipeline details

As previously underlined several times, the final goal of the thesis work is to create a single system composed of the two models presented. The main purpose is to implement a denoising system that is however useful for subsequent tasks, such as the recognition of the speaker. For this reason it was decided to create this pipeline composed of both models organized in such a way as to constitute a single end-to-end system during the training phase. This type of structure allows models to communicate among them so that they can provide each other with feedback relating to the performance of the overall system.

The first element of interest, given its dominant position within the pipeline, is the WaveNet. This architecture represents the main element, as it is the one actually implemented from scratch. To begin, it should be noted that, being a speech enhancement model, it was not possible to use a particular measure as an indicator of the quality of the network performance during training. This role usually for classification models is covered for example by measures such as accuracy[25] or F-score[26]. For what concerns the training of the WaveNet, on the other hand, the trend of the loss function (Energy conserving loss) was observed, both in training and in validation. It was therefore decided to proceed by saving the model whenever both showed an improvement compared to previous epochs. For this reason, none of the three WaveNet models has been trained using a precise number of epochs but, on the contrary, it has been chosen to proceed with the training until the loss has reached a substantial plateau[27] area.

[25]Informally, accuracy is the fraction of predictions the model got right. Formally, accuracy has the following definition: $\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$.

[26]The F-score is a way of combining the precision and recall of the model, and it is defined as the harmonic mean of the models precision and recall: $\text{F-score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$.

[27]In ML it is said that the loss function has reached a plateau zone when, despite the training continues, the loss does not continue to fall but rather remains substantially constant.

An example of monitoring the loss functions in training and validation is represented graphically in Fig. 4.19. With this type of representation it is also possible to verify the functioning of the WaveNet for the speech enhancement task.

Fortunately, the training part relating to the Thin ResNet-34 was not necessary as it was possible to use a pre-trained model[28]. This element is very important because, taking a training like that practiced by the authors of the original work, considering the means available, would have required a lot of time and much more powerful machines.

After a first training phase of the different WaveNet models created, and considering that the ResNet has been implemented starting from a pre-trained model, the next phase involved the union of the two models. In this sense, since the objective of the further training phase is to improve the performances of the WaveNet, this operation could be considered as a sort of fine-tuning[29] of the network. The procedure basically consisted in placing the ResNet in the queue of the WaveNet, "freezing the weights". This practice essentially consists in excluding the weights that are not intended to be updated through training from the back-propagation mechanism, not making them visible to the optimizer. In addition to the advantage that the weights are not updated (not risking therefore to compromise the initial performances), a certain acceleration is also obtained with respect to plotting the operations, since for all these parameters the gradients are not calculated and consequently there is a saving in terms of memory and calculation time. The complete procedure consists of several stages:

- First, the data batch is prepared as in the WaveNet training phase and submitted to the denoise network;
- after the denoising operation the loss function (Energy conserving loss) of the network output is calculated with respect to the same audio but without noise;
- at this point both the WaveNet output and the corresponding noiseless audio are subjected to the normalization operation. This procedure is necessary to keep ResNet inputs compatible with the training procedure it has received;
- once normalized, the output of the WaveNet and the ground truth are subjected to a transformation, in particular a Mel spectrogram;
- the Mel spectrograms are subjected to the ResNet architecture which returns for each one an embedding of dimensionality 1×512;
- a comparison between the two through a Mean Squared Error (MSE) loss (or L2 loss[30]) function is made;

[28]The complete repository containing the code and the pre-trained model of the Thin ResNet-34 is available at `https://github.com/clovaai/voxceleb_trainer`.

[29]The term fine-tuning usually means taking weights of a trained NN and use it as initialization for a new model being trained on data from the same domain (often e.g. images).

[30]L2 Loss Function is used to minimize the error which is the sum of all the squared differences between the true value and the predicted value i.e. L2 loss = $\sum_{i=1}^{n}(y_{\text{true}} - y_{\text{predicted}})^2$.

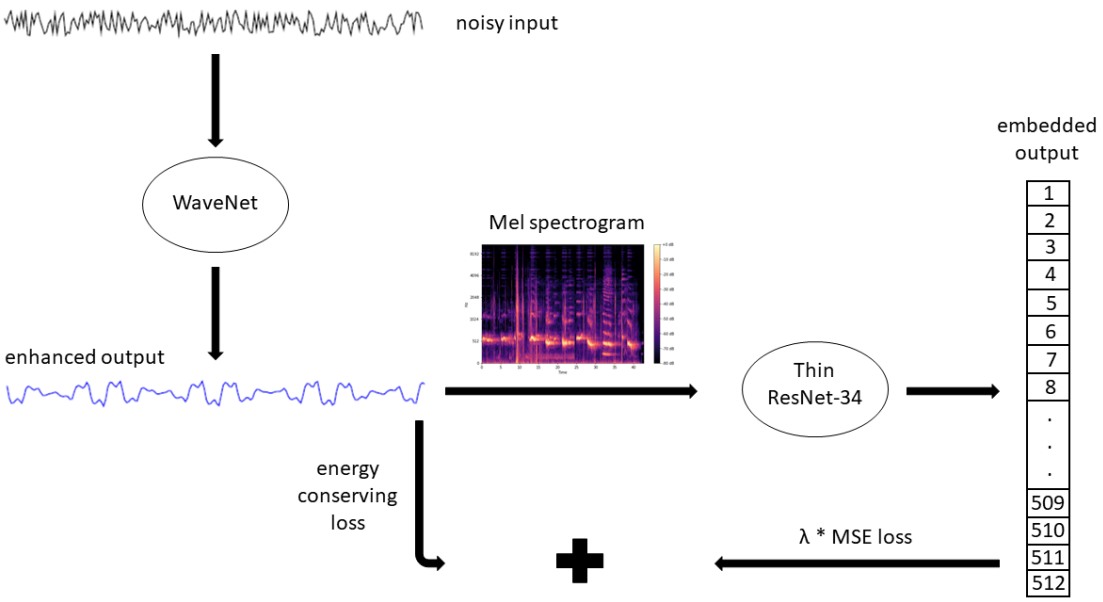

Figure 4.20: Summary of the fine-tuning system for the WaveNet network obtained by combining this architecture with the pre-trained Thin ResNet-34.

Table 4.7: Summary of the number of epochs needed to reach convergence for all three models. The number distinguishes between training and fine-tuning.

Model	**n. of epochs training**	**n. of epochs fine-tuning**
SIWIS	75	83
VoxCeleb	92	66
SIWIS+VoxCeleb	87	82

- after numerous attempts to combine the two calculated loss functions, it was decided to carry out a simple sum between the functions but multiplying the MSE loss by a factor λ. This operation has proved necessary since the MSE loss is calculated on the outputs obtained from normalized inputs and therefore different in terms of orders of magnitude compared to those used to calculate the Energy conserving loss;
- finally, once the overall loss function has been obtained, the back-propagation operation is carried out, addressed exclusively to the WaveNet parameters.

Also for this fine-tuning phase it was decided to proceed by monitoring the trend of the loss in training and validation using the achievement of a substantial plateau as a stop criterion. Provided they are not used as a stopping criterion, as regards the actual number of epochs used in the training and fine-tuning phase for each model, these are shown in Tab. 4.7. The saving criterion has remained unchanged: also in this case, for each improvement of the loss function in training and validation, a checkpoint

has been created to be able to resume the procedure later. Similar to WaveNet's first training phase, an Adam optimizer with a learning rate scheduler was also used for its fine-tuning phase. The one saved from the last training period was not chosen for reuse, but another one was initialized. With reference to the overall loss function, it can be mathematically summarized as

$$\mathcal{L}(\hat{s}_t) = (|s_t - \hat{s}_t| + |b_t - \hat{b}_t|) + \lambda * (z_t - \hat{z}_t)^2 \tag{4.9}$$

where the first term contains the specification of the Energy conserving loss (see Sec. 4.2.1), λ represents the multiplication factor of the second term to adjust the order of magnitude (in the case of the experiments carried out, $\lambda = 1000$ was chosen), and the second term is the MSE loss between the audio enhanced $\hat{z}_t$ and the ground truth z_t both normalized and passed through the ResNet.

Chapter 5

Performance Evaluation

At this point, all the theoretical elements necessary to understand the work carried out and the work pipeline in its entirety were presented. The objective of this chapter is to present all the results obtained in a quantitative and measurable way. To better understand the methods applied and the visualizations that will be proposed, however, it is first of all necessary to briefly introduce the applied measures from a theoretical point of view. Subsequently, the results obtained will be presented and discussed, represented by appropriate visualizations. Finally, a brief comment on the results will be given.

5.1 Performance measurement methods

In order to fully understand why certain measures have been chosen, it is necessary to explain how they are calculated and, in particular, what they represent. First of all it is important to distinguish between two types of measures used: objective measures of audio quality, used in the context of speech enhancement (see Sec. 3.2), and in this case used for measuring the denoising aspect of the pipeline, and classification quality measures, therefore more focused on the performance of the speaker verification task. Both types will be presented in the following sections.

5.1.1 Measures of audio quality

When it comes to audio quality measures, it is first necessary to distinguish between subjective and objective measures [121]. The purpose of these measures is to quantitatively measure the quality and intelligibility of the audio: they are particularly useful for comparing the same audio before and after being subjected to an enhancement process. The distinction between quality and intelligibility is not accidental, in fact they are two distinct concepts. While the former is a subjective measure which reflects the way the signal is perceived by listeners, the latter is the measure of the effective-

Table 5.1: Scale of degradation mean opinion score. From [121].

Rating	Quality	Impairment
5	Excellent	Imperceptible
4	Good	Perceptible-not annoying
3	Fair	Slightly annoying
2	Poor	Annoying
1	Unsatisfactory	Very annoying

ness of speech and is usually expressed as a percentage of message that is understood correctly.

Regarding subjective distortion measures, these are based on the opinion of a listener or a group of listeners who rate the performance of a system or quality of a signal in accordance with an opinion scale. The subjects provide their opinion on the quality of each signal using the predefined listening-quality scale. Typically, the scale used to evaluate the quality of the audio corresponds to that represented in Tab. 5.1. These measures are time-consuming and costly to obtain, requiring a set of discriminating listeners. These characteristics, in the context of a thesis work, therefore preclude the possibility of using this set of measures as a reference, even if it would mean using the most accurate assessment of the performance since the degree of perceptual quality and intelligibility are determined by the human auditory system.

The objective measures of audio quality, on the other hand, are very different. These measures are based on a mathematical comparison of the original and processed (enhanced) speech signals. The purpose with which they were made is to reproduce, as faithfully as possible, what would be the results obtained through subjective measurements. Unlike subjective measures, however, these do not require people to evaluate the quality of the audio so their use involves great savings in terms of costs and time. As already mentioned in Sec. 3.2, as regards the evaluation of the performances of speech enhancement algorithms, in the state of the art the measures used are essentially five: signal distortion (SIG), background noise distortion (BAK), overall quality (OVL), Perceptual Evaluation of Speech Quality (PESQ) and Segmental signal-to-noise ratio (SSNR) [122]. These measurements are composed and each focus on some particular aspects of the audio signal being evaluated. They will be explained in more detail below[1].

The SIG, followed by the BAK and the OVL measures normally are among the subjective measures (an explanation of the value assumed by these measures is visible in Tab. 5.2) and actually constitute a single measurement that analyzes the audio signal on several fronts. Referring to [32] it is possible to identify the mathemati-

[1]To implement all the measures in Python a package is already available on Github at the link

Table 5.2: Description of the SIG and BAK scales used in the subjective listening tests. From [122].

Rating	SIG Description	BAK Description
5	Very natural, no degradation	Not noticeable
4	Fairly natural, little degradation	Somewhat noticeable
3	Somewhat natural, somewhat degraded	Noticeable but not intrusive
2	Fairly unnatural, fairly degraded	Somewhat intrusive
1	Very unnatural, very degraded	Very intrusive

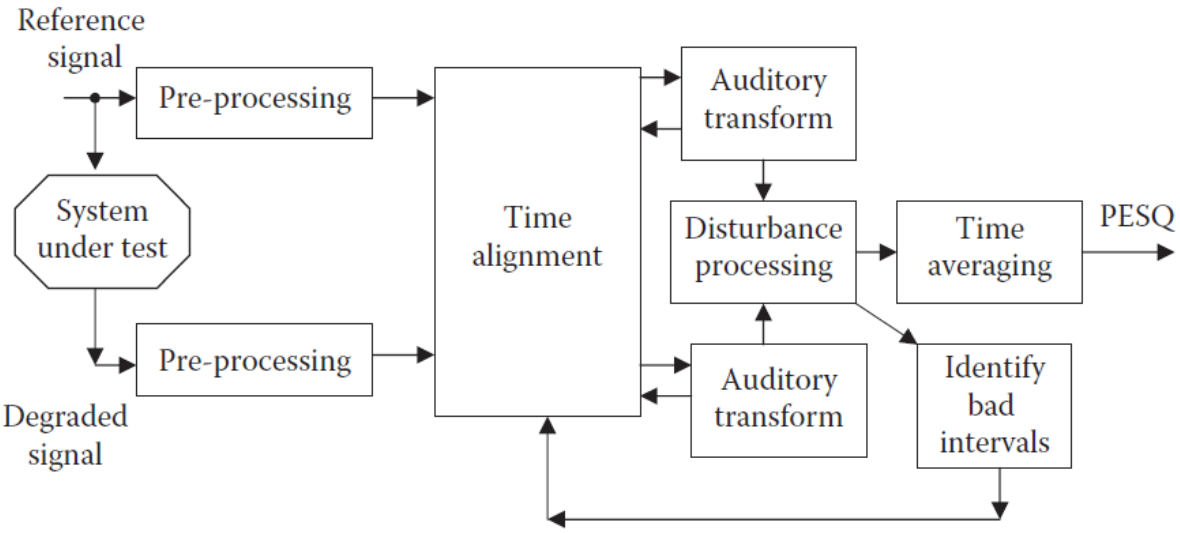

Figure 5.1: Block diagram of the PESQ measure computation. From [32].

cal formalization of these composite measurements, obtained by combining different measurements:

$$\text{CSIG} = -0.261 + 1.562 \cdot \text{IS} - 0.02 \cdot \text{PESQ} \tag{5.1}$$

$$\text{CBAK} = 1.893 + 0.007 \cdot \text{IS} + 0.8 \cdot \text{PESQ} - 0.468 \cdot \text{CEP} + 0.291 \cdot \text{LLR} - 0.008 \cdot \text{WSS} \tag{5.2}$$

$$\text{COVL} = -0.736 - 0.012 \cdot \text{IS} + 1.50 \cdot \text{PESQ} \tag{5.3}$$

where LLR, IS and CEP are respectively the log likelihood ratio measure, the Itakura-Saito, the cepstrum distance measure [123] while WSS represents the weighted spectral slope distance [124]. The PESQ measure [125] aims to reproduce as similarly as possible what would be a subjective evaluation in terms of Mean Opinion Score (MOS). The procedure necessary for its calculation is very complex but can be summarized through the diagram shown in Fig. 5.1. Mathematically, its final passage can be calculated as:

$$\text{PESQ} = a_0 - a_1 D_{ind} - a_2 A_{ind} \tag{5.4}$$

`https://github.com/schmiph2/pysepm` was used.

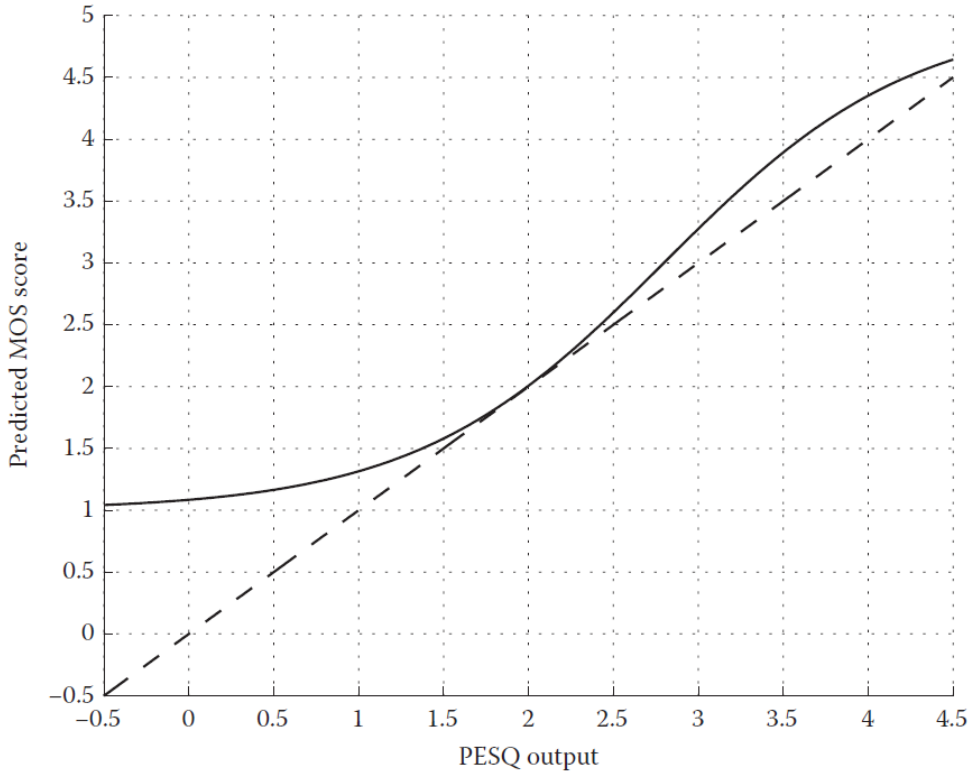

Figure 5.2: Output mapping functions to MOS scores as used for the narrowband PESQ (dashed lines) and wideband PESQ (solid lines) implementations. From [32].

with D_{ind} representing the average disturbance value and A_{ind} the average asymmetrical disturbance value. The coefficients a_0, a_1 and a_2 are estimated using a function of the predicted MOS scores following the relationship shown in Fig. 5.2. The last measure, the SSNR is very simply equivalent to the geometric mean of SNRs across all frames of the speech signal. Mathematically, to calculate the SSNR the equation is equivalent to:

$$\text{SSNR} = \frac{10}{M}\sum_{m=0}^{M-1}\log_{10}\frac{\sum_{n=Nm}^{Nm+N-1}x^2(n)}{\sum_{n=Nm}^{Nm+N-1}(x(n)-\hat{x}(n))^2} \tag{5.5}$$

with $x(n)$ representing the original (clean) signal, $\hat{x}(n)$ the enhanced signal, N the frame length (usually equal to 15-20 milliseconds) and M the number of frames in the signal. One potential problem with the estimation of SSNR is that the signal energy during intervals of silence in the speech signal: for this reason very often modified versions are used that take into account the VAD to exclude silent fragments of the original audio from the calculation.

All these measures are characterized by a precise range of values. As for the first four (SIG, BAK, OVL and PESQ) this range is equal to $[1, 5]$, where the higher the value, the better the enhancement process was. The range of values obtainable from

SSNR, on the other hand, is between $[0, +\infty]$.

5.1.2 Measures for the quality of the classification

In addition to the measures used to quantify the effectiveness of the audio background noise removal task, it was also necessary to measure the effectiveness of the speaker identity verification system. As already mentioned in Sec. 4.2.2, only during the training phase the ResNet network had the identities of the speakers, so only in this phase it was possible to use measures such as accuracy. Concerning the evaluation phase, both in training and subsequently to verify the model's performance on the basis of different parameters, a different measure was used, namely the Equal Error Rate (EER).

The EER is often used for the validation of biometric systems, as it can be calculated, unlike the accuracy, even without having the predictions of the labels. Assuming that, in the specific case, at the time of the evaluation there is a ground truth in the format $[0, 1]$, to indicate whether the identities of the speakers of the two audios are the same person. Concerning the prediction of the system instead, it consists of a distance measure calculated between the features of the two audio extracted from ResNet. In particular, the distance measure used between the two 1×512 vectors is the *pairwise distance*, which mathematically can be represented as

$$||x||_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} \tag{5.6}$$

where p represents the norm degree, normally placed equal to 2.

In order to deepen the EER it is first necessary to introduce two further concepts: the False Reject Rate (FRR) and the False Accept Rate (FAR). While the former represents the rate of false negatives (*type I error*), the latter represents the exact opposite, which is the proportion of false positives (*type II error*). In many areas, such as security, a false positive is considered more serious than a false negative, while in others, such as the medical field, the opposite is true. Given these premises, the EER (also called Crossover Error Rate) can be defined as the point where FRR and FAR are equal [126]. As the sensitivity of a biometric system increases, FRRs will rise and FARs will drop. Conversely, as the sensitivity is lowered, FRRs will drop and FARs will rise. Fig. 5.3 shows a graph depicting the FAR versus the FRR. The EER is the intersection of both lines of the graph.

Since, as already pointed out, the predictions do not consist of labels but rather in scores based on a distance measure, to obtain the FRR and the FAR necessary for the calculation of the EER, it is essential to use an additional measure, namely the Receiver Operating Characteristic (ROC) curve[2]. Very briefly, the ROC curve is created by

[2]Scikit-learn provides a very usefull built-in ROC curve function. `https://scikit-learn.org/s`

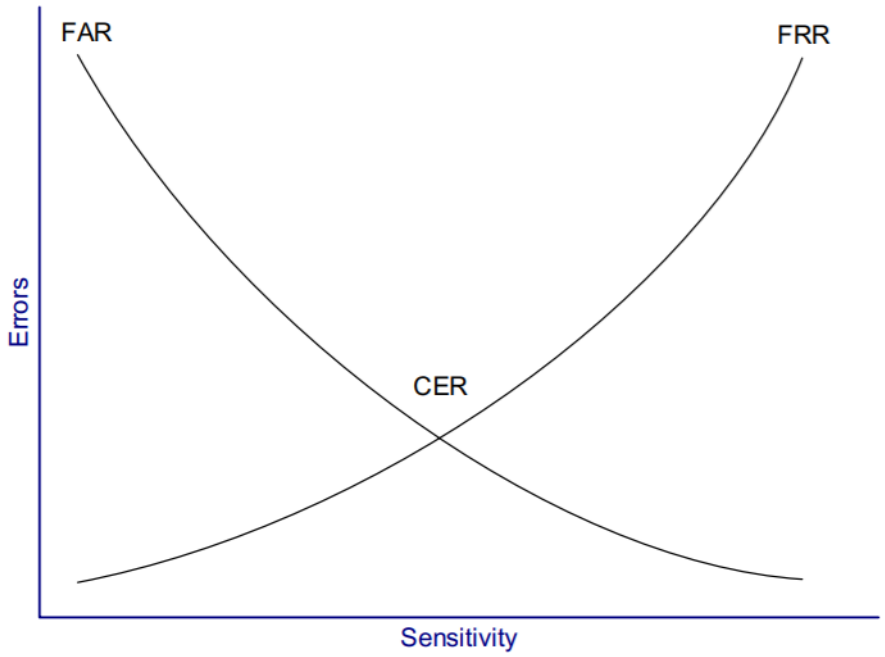

Figure 5.3: Crossover Error Rate. From [126].

plotting the value of the True Positive Rate (TPR) (fraction of true positives) versus the False Positive Rate (FPR) (fraction of false positives) that another term for the FAR at various threshold settings. The ROC curve is therefore the true positive rate as a function of the false positive rate. Visually the ROC curve is represented as in Fig. 5.4.

Operationally, to calculate the EER then first of all proceed to calculate the ROC curve to obtain in this way the FPR (or FAR) and TPR. Mathematically these measures are obtainable as

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{5.7}$$

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{5.8}$$

with FP, TN, TP and FN respectively equal to False Positive, True Negative, True Positive and False Negative. Later it is possible to obtain the False Negative Rate (FNR) (or FRR) directly simply as

$$\mathrm{FNR} = 1 - \mathrm{TPR} \tag{5.9}$$

At this point, for each threshold, the difference in absolute value between FNR and FPR is calculated, thus identifying which is the smallest difference. Finally, in correspondence with what has been identified as the smallest difference, the corresponding

table/modules/generated/sklearn.metrics.roc_curve.html.

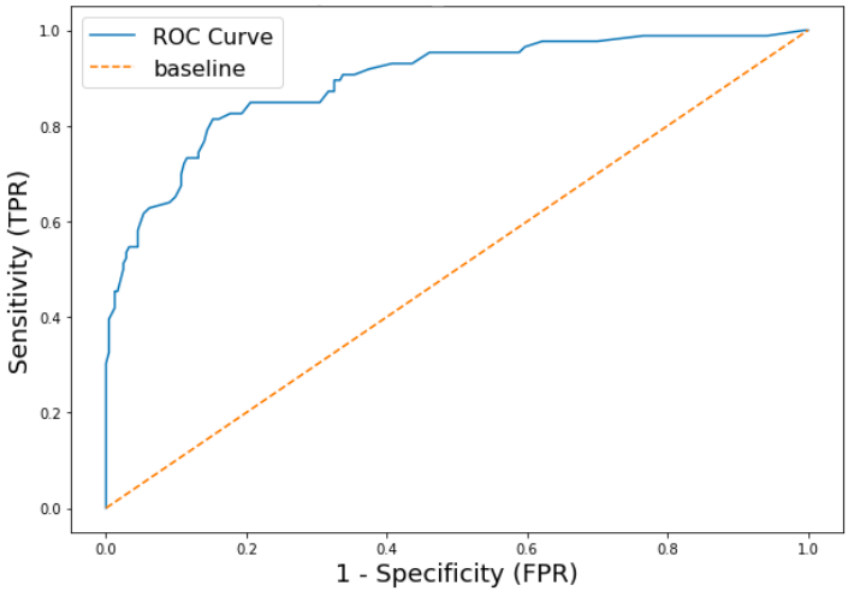

Figure 5.4: ROC curve example. From `https://towardsdatascience.com/roc-curve-in-machine-learning-fea29b14d133`.

values of FPR and FNR are taken: the maximum value between the two will be the one used as EER.

5.2 Results

At this point, after introducing the measurements used to evaluate the different elements of the pipeline from a methodical point of view, it is possible to present all the results achieved. It is important to remember, as mentioned in Sec. 4.3.2, that there are three models created, one for each dataset (SIWIS, VoxCeleb and SIWIS+VoxCeleb). Furthermore, all three models were evaluated at the end of the first training procedure and then again after the fine-tuning procedure.

The first result on which attention must be paid regards the performance of each WaveNet model created, in terms of EER, for each SNR used during the construction phase of the datasets with additive background noise. In this sense, Fig. 5.5 is particularly explanatory. For each model, the performances achieved in the recognition task are represented after the audio enriched with background noise has been subjected to the denoise network. The result is segmented based on the SNR used to add noise to the original audio. Pure audio performance with background noise is also visible. As additional information, the average value of EER reached is also graphically represented for each SNR.

The second set of results concerns the difference in terms of performance again for the speaker verification task but comparing the performances before and after

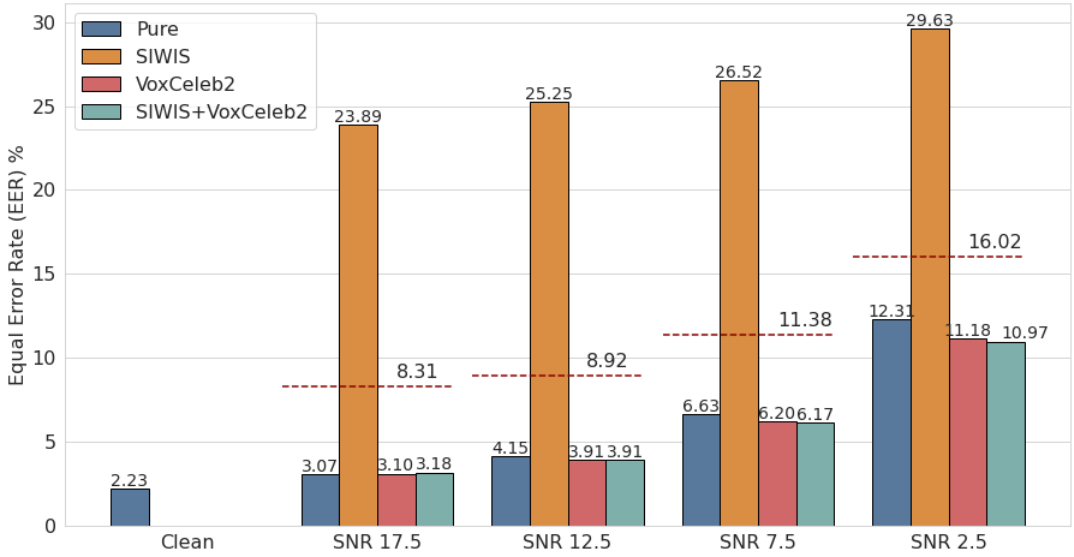

Figure 5.5: Results expressed in terms of EER% achieved by ResNet-54 for the speaker verification task after the audio enriched with background noise has been subjected to the denoise procedure by the three distinct WaveNet models. Lower is better. The test result for non-denoised audio is also visible.

the fine-tuning phase of the networks. For this purpose, the visualizations created have been designed to give as clearly as possible the idea of the comparison between the result obtained using the audio with the background noise, the improved audio from the network after training and finally the improved audio after the fine-tuning procedure. As for the fine-tuning procedure, this was described in detail in Sec. 4.3.3. For each model, i.e. those trained on SIWIS, VoxCeleb2 and SIWIS+VoxCeleb2, a separate plot was created, respectively represented in Fig. 5.6, 5.7 and 5.8, to make it easier to compare the same models after the training procedure and after that of fine-tuning, always offering a comparison with the baseline determined by the performance obtained by ResNet on audio not purified from background noise.

Another very interesting element from the point of view of exploring the results is the mixture. By mixing is meant the operation of adding together the denoised audio and the same audio not subjected to denoising, both multiplied by an appropriate coefficient. Mathematically this operation looks like

$$z = \alpha * \hat{x} + (1 - \alpha) * x \tag{5.10}$$

with α representing a coefficient with values between 0 and 1, $\hat{x}$ indicating enhanced audio and x representing audio with background noise. In fact, in the literature it is quite common, as regards the speech enhancement task, to recombine the enhanced audio with the starting audio in a certain proportion to obtain a final audio that is better than the two from which it originated. Based on these considerations, the

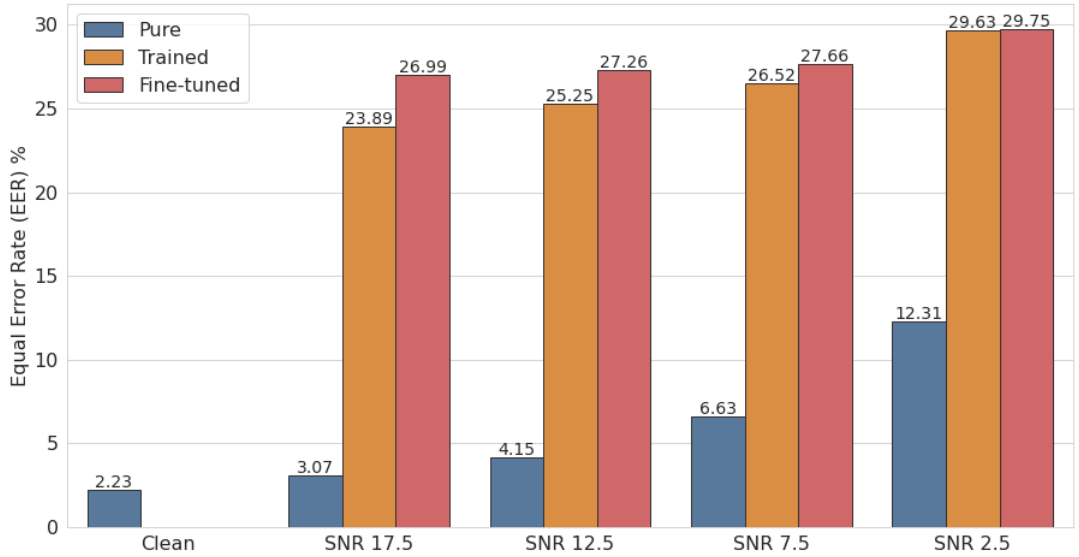

Figure 5.6: Results achieved by the model trained on the SIWIS dataset. Difference between the performance in speaker verification obtained from audio with background noise, those subjected to the denoising procedure from the model after the training phase, and those improved by the model after the fine-tuning phase. Lower is better.

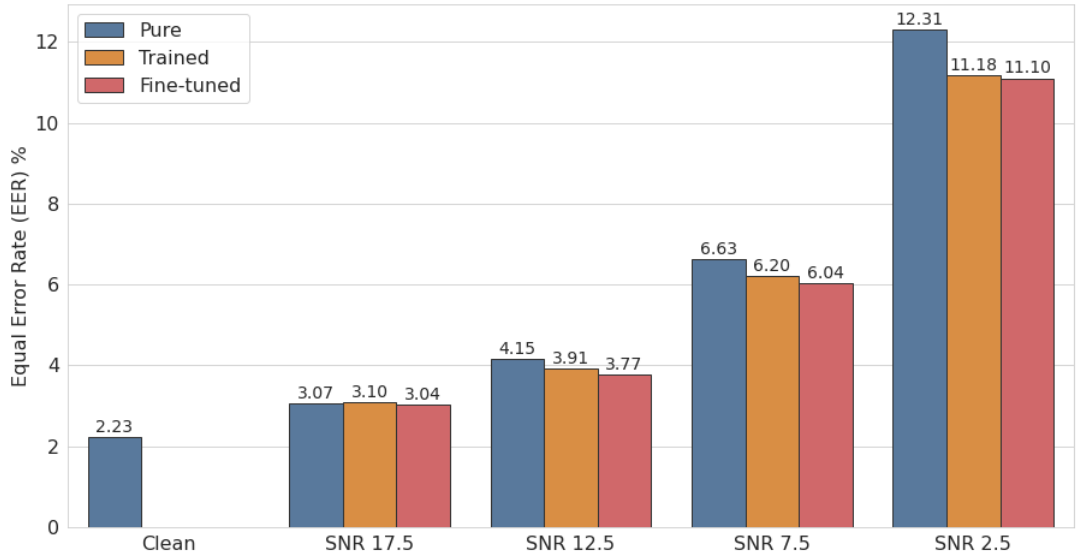

Figure 5.7: Results achieved by the model trained on the VoxCeleb2 dataset. Difference between the performance in speaker verification obtained from audio with background noise, those subjected to the denoising procedure from the model after the training phase, and those improved by the model after the fine-tuning phase. Lower is better.

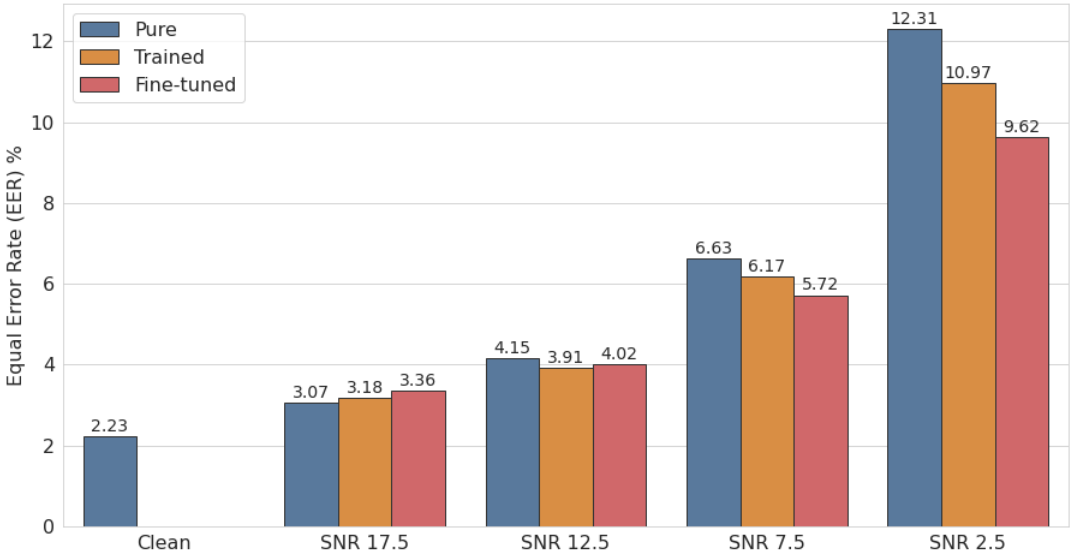

Figure 5.8: Results achieved by the model trained on the SIWIS+VoxCeleb2 dataset. Difference between the performance in speaker verification obtained from audio with background noise, those subjected to the denoising procedure from the model after the training phase, and those improved by the model after the fine-tuning phase. Lower is better.

impact of a blending approach on ResNet performance in terms of EER was measured. This type of measurement was carried out both for the models after the first training phase and after the fine-tuning phase to verify the hypothetical impact of the latter on the change of balance in the mixing. The results regarding the former are shown in Fig. 5.9 while for the latter they are shown in Fig. 5.10.

The last section of results produced concerns the sheer ability to remove background noise from audio using the different WaveNet models. As anticipated in Sec. 3.2, the measures used to verify this task are substantially five (fully explained in Sec. 5.1.1). The three visualizations realized respectively represent the performances for the denoise task of one of the WaveNet models. Therefore, observing Fig. 5.11, 5.12 and 5.13 in order, it is possible to observe the performances of the SIWIS, VoxCeleb and SIWIS+VoxCeleb models respectively. The individual representations are composed of a first plot representing a radar plot with the values obtained for the PESQ, BAK, SIG and OVL measurements, flanked by a second plot representing the performances in terms of SSNR. Also in this case, the performances were checked both before and after the fine-tuning phase in order to verify the impact of this procedure overall.

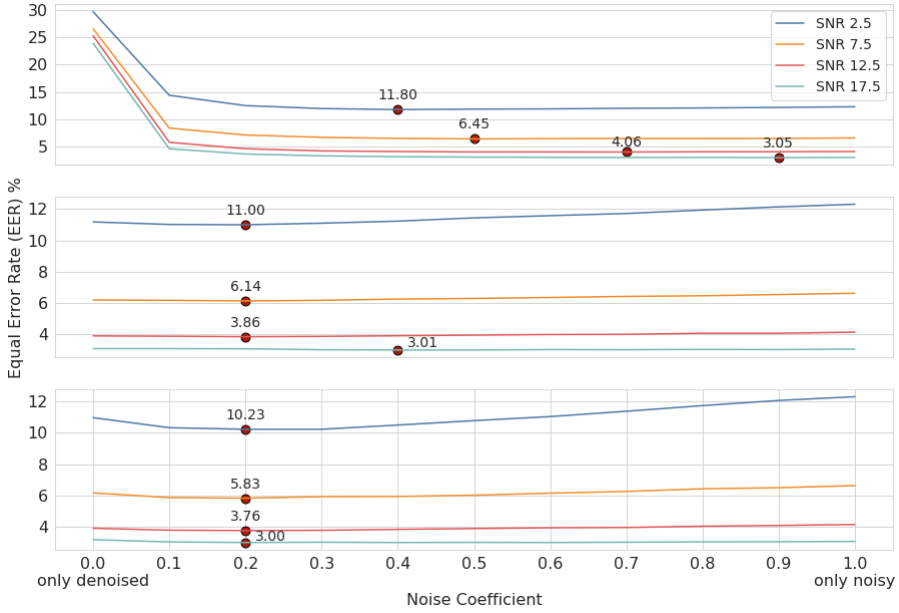

Figure 5.9: Results achieved by the models after the training phase. Representation of mixing for alpha values ranging from 0 to 1 with a granularity of 0.1. Each box represents a model. Starting from the top SIWIS, VoxCeleb2 and finally SIWIS+VoxCeleb2. Each line within the different boxes represents a different SNR. The minimum point for each SNR is marked with a red dot.

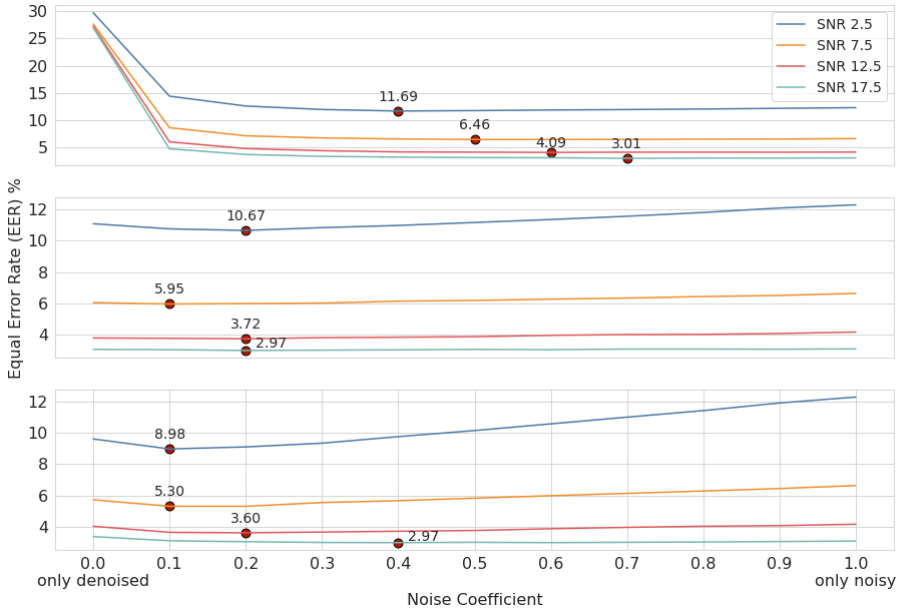

Figure 5.10: Results achieved by the models after the fine-tuning phase. Representation of mixing for alpha values ranging from 0 to 1 with a granularity of 0.1. Each box represents a model. Starting from the top SIWIS, VoxCeleb2 and finally SIWIS+VoxCeleb2. Each line within the different boxes represents a different SNR. The minimum point for each SNR is marked with a red dot.

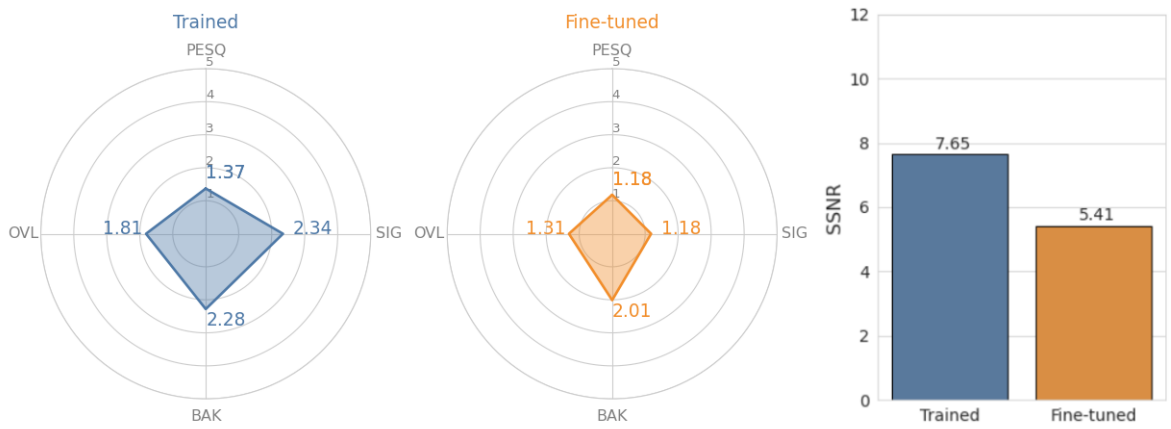

Figure 5.11: Denoising performance for the SIWIS model. Each measurement is calculated for both before and after the fine-tuning phase. For every measure higher is better.

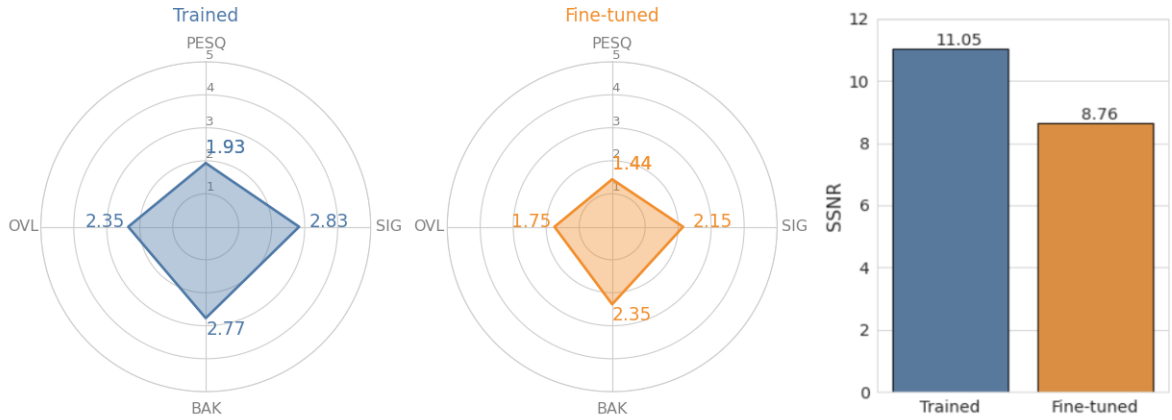

Figure 5.12: Denoising performance for the VoxCeleb model. Each measurement is calculated for both before and after the fine-tuning phase. For every measure higher is better.

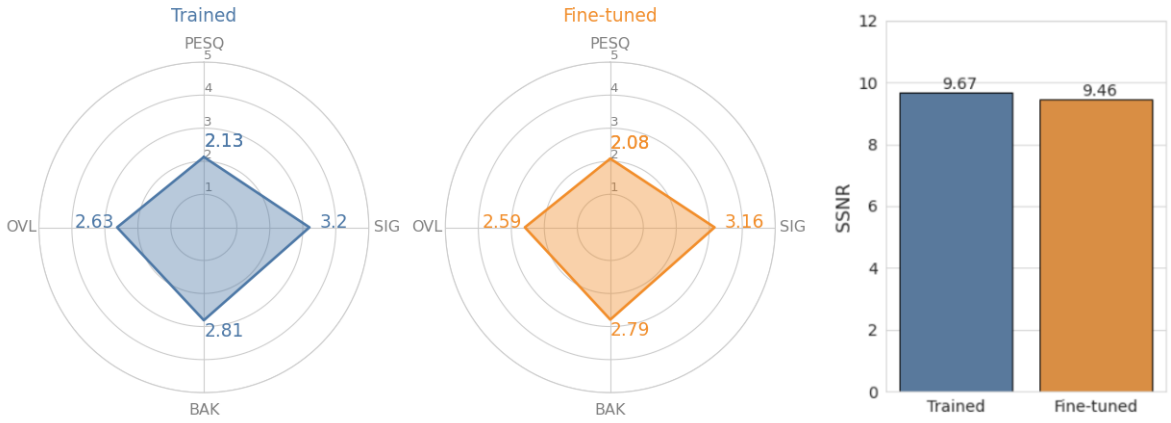

Figure 5.13: Denoising performance for the SIWIS+VoxCeleb model. Each measurement is calculated for both before and after the fine-tuning phase. For every measure higher is better.

5.3 Discussion

Now that all the results have been graphically represented, it is certainly useful to comment one by one the approaches used and the implications that these results entail.

The first result concerns the performance in terms of EER achieved by the individual models for each SNR. This result is represented in Fig. 5.5. First of all it is possible to observe that the lower the SNR the greater the error obtained in the SV task. Such a result is perfectly in line with expectations, it appears physiological that the greater the prevalence of noise within the audio signal, the more difficult it becomes to be able to recognize the identity of the speaker. An unexpected element, however, concerns the level of performance reached by the different models. In fact, it is clear that the model trained exclusively on the SIWIS dataset obtains extremely negative performances, with the same methodology adopted during the training phase. The responsibility for this negative result could be attributed to the nature of the starting dataset. As widely explained in Sec. 4.1.1, in fact, the SIWIS dataset represents an extremely difficult baseline to reach, as the basic audio is recorded under controlled conditions and is therefore extremely clean. This therefore implies that, using the model on the dataset used to evaluate the classification of the ResNet (i.e. the VoxCeleb1 test set, as anticipated in Sec. 4.3.2) it can cause an excessive removal of background noise. Such a suppression of background noise could lead the model to also remove very characteristic features of the speaker's vocal timbre, drastically worsening the recognition performance. On the other hand, observing the other models it is possible to identify two very distinct behaviors. The first concerns the general performance. Indeed, it would appear that the model trained on the SIWIS+VoxCeleb2 dataset is the one that achieves the best results. This could be due to the fact that the union of two such different datasets allows the model to generalize more during the training phase. Another particular behavior that can be observed is that both of these models would appear to perform better when the noise is louder. This can be taken for granted: the louder the noise, the easier it is to identify and remove it. However, the fact remains that the removal operation could compromise the features useful for recognition, but in this case does not happen and, indeed, the louder the noise, the better the results obtained.

The second result that is worth examining is the direct comparison, for each model, of the performance in terms of EER before and after the fine-tuning procedure. In particular, the reference graphical representations for this result are those in Fig. 5.6 for the SIWIS dataset, and in Fig. 5.7 and 5.8 for the VoxCeleb and SIWIS+VoxCeleb datasets. In accordance with what already happened in the training phase, for the SIWIS dataset, the application of the fine-tuning procedure only further worsens per-

formance. The same principle applies to the other two models, i.e. the patterns shown after the preliminary training phase are repeated slightly more markedly after the fine-tuning phase. The model trained on the VoxCeleb2 dataset is improved with respect to any SNR, even if very little in some cases. The one trained on the SIWIS+VoxCeleb2 dataset, on the other hand, clearly improves with regard to the lower SNRs, while it would seem to begin to worsen for the higher ones. In this case therefore the presence of SIWIS begins to have a certain influence.

The next result examined concerns the combination of the enhanced audio with the starting noisy audio. For this type of result there are two visualizations realized, in order to distinguish between the models before (Fig. 5.9) and after (Fig. 5.10) the fine-tuning procedure. As is possible to see in the aforementioned figures, for all models, both before and after the fine-tuning procedure, for each SNR, the performance tends to improve by implementing a certain mixing between the enhanced audio and the noisy audio. The results obtained before the fine-tuning by applying the mixing procedure (shown in Fig. 5.9) are rather encouraging. Most of the absolute minimum points are obtained with very low mixing percentages, except for the SIWIS model. In any case, on average, the improvement obtained is not particularly incisive but still represents a good starting point. It is however interesting to note that, in no case, the enhanced audio turns out to be the best, but it is instead the mixing, even very light with the noisy audio, to obtain the best result. As for the results obtained after the fine-tuning, attention should be paid to Fig. 5.10. In proportion, the greatest improvement is achieved by the SIWIS model. In fact, it seems that the audio extraneously degraded by the network greatly benefits from the mixing procedure, managing to obtain better performances than pure noisy audio. This improvement reaches its peak with values of α around 0.5, thus implying a strong restructuring of the starting audio. However, it is worth noting that the big improvement already occurs starting from a coefficient $\alpha = 0.1$. As for the other two models, it is clear how these also gain performance from audio mixing, but to a lesser extent than the first. For these models, the optimal α value is decidedly lower, around 0.2. The gain from the point of view of the EER is undoubted even if not extremely significant. At first glance, it would also seem that the higher the SNR, the higher the α value must be to achieve the greatest performance improvement. However, the fact remains that this type of operation, or mixing, is in effect at no cost, and in any case produces positive effects in obviously improving the characteristic features of the voice timbre of the speakers.

The last set of evaluations taken into consideration concerns the evaluation of the denoising system from the point of view of pure noise removal. As already explained in Sec. 5.1.1, this type of measurement quantifies how much the audio quality has improved from the point of view of intelligibility. However, an increase in this feature does not necessarily correspond to an improvement in the performance obtained for

the SV task. Also for this reason it was necessary to build the overall system as illustrated in Sec. 4.3.3, precisely to be able to produce a denoising that had a positive impact on the classification performance of the speaker. This set of measurements are shown graphically in Fig. 5.11, 5.12 and 5.13, representing respectively the SIWIS, VoxCeleb and SIWIS+VoxCeleb models. As is possible to see at a glance, the best model based on all measurements (except for the SSNR) is the one trained on the SIWIS+VoxCeleb dataset. Also with regard to this set of measures, the model trained on SIWIS is clearly the worst. Another element of interest is that, for all three models, the performance after the fine-tuning procedure deteriorates compared to all measurements. This particular behavior could be due to the fact that, by placing the ResNet in the queue of the WaveNet, the latter actually mitigated its behavior in terms of noise removal, to favor a greater conservation of the features (which is exactly the type of result that is was trying to get).

Therefore, in general it can be said that the fine-tuning procedure has partially satisfied the expected results. In fact, it is clear how the performance in speaker verification improves after applying this phase. Another noteworthy and already widely underlined element is the dataset that was the best for training a WaveNet model. The SIWIS+VoxCeleb dataset would seem to represent a good balance as regards the type of audio: VoxCeleb contains basic audio that already contains a small portion of noise, while SIWIS represents a rather demanding baseline to achieve.

Chapter 6

Conclusions

To summarize, the thesis work was divided into several phases. First of all it was necessary to identify a suitable source of audio data, and carry out its processing (see Sec. 4.3.1). Subsequently, a neural architecture was created with the aim of carrying out the denoising task (see Sec. 4.2.1) and another with the aim of identifying the speaker (see Sec. 4.2.2). The initial approach was that of a first training of the speech enhancement model. Afterwards, as established in the main objective set at the beginning of the thesis work, after the creation of the denoising network and its initial training an end-to-end system was created consisting of the two networks in order to verify the possible benefit of the first network from a fine-tuning procedure structured in this way. The purpose of this is to verify that the denoising network can be useful as it is used as a starting point for a subsequent task, such as that of identifying the speaker. The training procedure in its entirety is presented in Sec. 4.3.3.

In conclusion, it can be said that the results obtained are quite satisfactory. In fact, considering what is explained in Sec. 5.2, and the considerations expressed in Sec. 5.3, it can be said that the initial objective has been partially achieved. The system created, trained in an end-to-end form, showed actual improvements in the speaker verification task, thanks to the strategy adopted.

The large amount of technical limitations encountered during the realization of the thesis work must also be taken into consideration. These could be considered a first form of necessary improvement. The scarcity of resources in fact precluded the possibility of carrying out a whole series of tests, which with the means available would have required a considerable consumption of time. From this point of view it is therefore correct to think that the modesty of the results is partly attributable to these limitations encountered. However, despite the modest entity of the results found, it is still possible to observe a certain behavior in the results obtained. It can be imagined that there is further space for improvement if the pipeline is developed. Furthermore, by breaking down the barrier of technological limits encountered (attributable to the

particular historical moment) there is the possibility that a more efficient solution can be developed with the same performance.

Considering all this it is therefore possible to define a whole series of possible future developments for this work. First of all, it would be desirable to try to modify the treatment of the data administered to the denoise network. From this point of view, actually working using a context aware model could prove to be a winning strategy for improving performance as regards denoising. Another element of deep interest could be to develop a generative solution for the speech enhancement task. Starting from the aforementioned GAN models it would be possible to create a CycleGAN inspired by the Pix2Pix model for images with the aim of improving the audio signal instead of pictures. Another explorable space could also be represented by the classification of the type of noise characterizing the audio, so as to predict a more precise response of the model based on the nature of the disturbance signal present.

Among the elements concerning the applications as a future work it could be interesting to verify the action of denoising on signals other than audio, such as electrocardiograms. Quantifying the possible positive impact on a whole range of one-dimensional signals would be absolutely interesting from the point of view of possible applications.

References

[1] Gamal Bohouta and Veton Këpuska. "Next-Generation of Virtual Personal Assistants (Microsoft Cortana, Apple Siri, Amazon Alexa and Google Home)". In: (Jan. 2018).

[2] Nilu Singh, R.A. Khan, and Raj Shree. "Applications of Speaker Recognition". In: *Procedia Engineering* 38 (2012). INTERNATIONAL CONFERENCE ON MODELLING OPTIMIZATION AND COMPUTING, pp. 3122–3126.

[3] G. R. Doddington. "Speaker recognitionIdentifying people by their voices". In: *Proceedings of the IEEE* 73.11 (1985), pp. 1651–1664.

[4] P. Dhanalakshmi, S. Palanivel, and V. Ramalingam. "Classification of audio signals using SVM and RBFNN". In: *Expert Systems with Applications* 36.3, Part 2 (2009), pp. 6069–6075.

[5] Nilu Singh, Alka Agrawal, and R. A. Khan. "Automatic Speaker Recognition: Current Approaches and Progress in Last Six Decades". In: *Global Journal of Enterprise Information System* 9.3 (2017), pp. 45–52.

[6] A. Narayanan and D. Wang. "Investigation of Speech Separation as a Front-End for Noise Robust Speech Recognition". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.4 (2014), pp. 826–835.

[7] Robert C. Maher. *Principles of Forensic Audio Analysis*. 1st ed. Modern Acoustics and Signal Processing. Springer International Publishing, 2018.

[8] J G Proakis; D G Manolakis. *Digital signal processing : principles, algorithms, and applications*. 4. ed. Prentice Hall international editions. Prentice-Hall, 2006.

[9] Federico Avanzini and Giovanni De Poli. *Algorithms for Sound and Music Computing*. Jan. 2012.

[10] Heleen Luts. "Diagnosis of hearing loss in newborns : clinical application of auditory steady-state responses". In: (Aug. 2020).

[11] Alan C. Bovik. *The Essential Guide to Image Processing*. Academic Press, 2009.

[12] David Dorran. "Digital Signal Processing Foundations". In: *School of Electrical and Electronic Engineering at ARROW@DIT* (2015).

[13] T. J. Deeming. “Fourier analysis with unequally-spaced data”. In: *Astrophysics and Space Science* 36 (1975), pp. 137–158.

[14] Steven Smith. *Digital Signal Processing: A Practical Guide for Engineers and Scientists*. Book and CD ROM. Newnes, 2002.

[15] Mittal Darji. “Audio Signal Processing: A Review of Audio Signal Classification Features”. In: *International Journal of Scientific Research in Computer Science, Engineering and Information Technology* 2 (May 2017), pp. 227–230.

[16] M. McKinney and J. Breebaart. “Features for audio and music classification”. In: (2003).

[17] Garima Sharma, Kartikeyan Umapathy, and Sridhar Krishnan. “Trends in audio signal feature extraction methods”. In: *Applied Acoustics* 158 (2020), p. 107020.

[18] F. J. Harris. “On the use of windows for harmonic analysis with the discrete Fourier transform”. In: *Proceedings of the IEEE* 66.1 (1978), pp. 51–83.

[19] Christopher Bingham, M Godfrey, and J Tukey. “Modern techniques of power spectrum estimation”. In: *IEEE Transactions on audio and electroacoustics* 15.2 (1967), pp. 56–66.

[20] N. Ahmed, T. Natarajan, and K. R. Rao. “Discrete Cosine Transform”. In: *IEEE Transactions on Computers* C-23.1 (1974), pp. 90–93.

[21] Zbigniew Engel, Maciej Kaczyski, and Wiesaw Wszoek. “A Vibroacoustic Model of Selected Human Larynx Diseases”. In: *International journal of occupational safety and ergonomics : JOSE* 13 (Feb. 2007), pp. 367–79.

[22] N. Dalal and B. Triggs. “Histograms of oriented gradients for human detection”. In: 1 (2005), 886–893 vol. 1.

[23] Tony Lindeberg. “Scale Invariant Feature Transform”. In: vol. 7. May 2012. DOI: 10.4249/scholarpedia.10491.

[24] Sergey Shuvaev, Hamza Giaffar, and Alexei Koulakov. “Representations of Sound in Deep Learning of Audio Features from Music”. In: (Dec. 2017).

[25] I. Martin-Morato, M. Cobos, and F. J. Ferri. “On the Robustness of Deep Features for Audio Event Classification in Adverse Environments”. In: (2018), pp. 562–566.

[26] Fred J. Damerau (Editors) Nitin Indurkhya. *Handbook of Natural Language Processing, Second Edition (Chapman & Hall CRC Machine Learning & Pattern Recognition Series)*. Chapman & Hall/CRC Machine Learning & Pattern Recognition Series”, Taylor and Francis Group, LLC, 2010.

[27] Lawrence Rabiner and B Juang. “An introduction to hidden Markov models”. In: *ieee assp magazine* 3.1 (1986), pp. 4–16.

[28] Homayoon Beigi. “Speaker Recognition: Advancements and Challenges”. In: Nov. 2012, pp. 3–29.

[29] J. P. Campbell. “Speaker recognition: a tutorial”. In: *Proceedings of the IEEE* 85.9 (1997), pp. 1437–1462.

[30] Douglas A Reynolds. “An overview of automatic speaker recognition technology”. In: 4 (2002), pp. IV–4072.

[31] A. Belouchrani et al. “A blind source separation technique using second-order statistics”. In: *IEEE Transactions on Signal Processing* 45.2 (1997), pp. 434–444.

[32] Philipos C. Loizou. *Speech Enhancement : Theory and Practice*. 2nd ed. CRC Press, 2013.

[33] Alireza Dibazar, Shrikanth Narayanan, and Theodore Berger. “Feature analysis for automatic detection of pathological speech”. In: *Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings* 1 (Feb. 2002), 182–183 vol.1.

[34] Kaitlin Lansford and Julie Liss. “Vowel Acoustics in Dysarthria: Speech Disorder Diagnosis and Classification”. In: *Journal of speech, language, and hearing research : JSLHR* 57 (Feb. 2014), pp. 57–67.

[35] Dong Yu Li Deng. *Deep Learning: Methods and Applications*. Foundations and Trendső in Signal Processing 7.3-4. Now Publishers, 2014.

[36] Tom M. Mitchell. *Machine Learning*. 1st ed. McGraw-Hill series in computer science. McGraw-Hill, 1997.

[37] Haohan Wang and Bhiksha Raj. *On the Origin of Deep Learning*. 2017. arXiv: 1702.07800 [cs.LG].

[38] Robert Hecht-Nielsen. “Theory of the backpropagation neural network”. In: *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[39] Terrence J. Sejnowski. *The Deep Learning Revolution: Machine Intelligence Meets Human Intelligence*. 1st ed. MIT Press, 2018.

[40] Aaron Courville Ian Goodfellow Yoshua Bengio. *Deep Learning*. The MIT Press, 2016.

[41] Alex Sherstinsky. “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network”. In: *Physica D: Nonlinear Phenomena* 404 (Mar. 2020).

[42] S. Albawi, T. A. Mohammed, and S. Al-Zawi. “Understanding of a convolutional neural network”. In: (2017), pp. 1–6.

[43] Michael A. Nielsen. “Neural Networks and Deep Learning”. In: Determination Press, 2015.

[44] Joseph Yacim and Douw Boshoff. “Impact of Artificial Neural Networks Training Algorithms on Accurate Prediction of Property Values”. In: *Journal of Real Estate Research* 40 (Nov. 2018), pp. 375–418.

[45] Mehmet Hacibeyoglu. “Human Gender Prediction on Facial Mobil Images using Convolutional Neural Networks”. In: *International Journal of Intelligent Systems and Applications in Engineering* 3 (Sept. 2018), pp. 203–208.

[46] Geoffrey I. Webb. “Overfitting”. In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 744–744.

[47] Alla Eddine Guissous. “Skin Lesion Classification Using Deep Neural Network”. In: (Nov. 2019).

[48] E.R. Davies. *Computer Vision: Principles, Algorithms, Applications, Learning*. 5th ed. Academic Press, 2017.

[49] Christian Hennig and Mahmut Kutlukaya. “Some thoughts about the design of loss functions”. In: *REVSTAT Statistical Journal Volume* 5 (Apr. 2007), pp. 19–39.

[50] Geoffrey E Hinton. “How neural networks learn from experience”. In: *Scientific American* 267.3 (1992), pp. 144–151.

[51] Shashi Sathyanarayana. “A Gentle Introduction to Backpropagation”. In: *Numeric Insight, Inc Whitepaper* (July 2014).

[52] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *CoRR* abs/1609.04747 (2016).

[53] Ian Goodfellow et al. “Generative Adversarial Nets”. In: *ArXiv* (June 2014).

[54] Hyeong-Seok Choi et al. “Phase-aware Speech Enhancement with Deep Complex U-Net”. In: (Mar. 2019).

[55] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *LNCS* 9351 (Oct. 2015), pp. 234–241.

[56] Dor Bank, Noam Koenigstein, and Raja Giryes. *Autoencoders*. 2020. arXiv: 2003.05991 [cs.LG].

[57] Michal Drozdzal et al. “The importance of skip connections in biomedical image segmentation”. In: (2016), pp. 179–187.

[58] Thanh Nguyen et al. “Automatic phase aberration compensation for digital holographic microscopy based on deep learning background detection”. In: *Optics Express* 25 (June 2017), pp. 15043–15057.

[59] M. Safizadeh, A. Lakis, and Marc Thomas. “USING SHORT-TIME FOURIER TRANSFORM IN MACHINERY DIAGNOSIS”. In: 3 (Jan. 2005).

[60] Shrikant Venkataramani, Jonah Casebeer, and Paris Smaragdis. “End-to-end Source Separation with Adaptive Front-Ends”. In: (2017). arXiv: 1705.02514.

[61] H.-L. Kuo. “A More Generalized Wiener Filtering Technique”. In: *IFAC Proceedings Volumes* 13.11 (1980). 3rd IFAC/IFIP/IFORS Conference on System Approach for Development., Rabat, Morocco, 24-27 November, pp. 539–542.

[62] Joachim Thiemann, Nobutaka Ito, and Emmanuel Vincent. “The diverse environments multi-channel acoustic noise database: A database of multichannel environmental noise recordings”. In: *The Journal of the Acoustical Society of America* 133.5 (2013), pp. 3591–3591.

[63] C. Veaux, J. Yamagishi, and S. King. “The voice bank corpus: Design, collection and data analysis of a large regional accent speech database”. In: (2013), pp. 1–4.

[64] Santiago Pascual, Antonio Bonafonte, and Joan Serrà. “SEGAN: Speech Enhancement Generative Adversarial Network”. In: (2017). arXiv: 1703.09452 [cs.LG].

[65] Jamie Hayes et al. “LOGAN: Evaluating Privacy Leakage of Generative Models Using Generative Adversarial Networks”. In: (May 2017).

[66] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: (2015). arXiv: 1512.03385 [cs.CV].

[67] Alexandre Defossez, Gabriel Synnaeve, and Yossi Adi. “Real Time Speech Enhancement in the Waveform Domain”. In: (2020). arXiv: 2006.12847 [eess.AS].

[68] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-term Memory”. In: *Neural computation* 9 (Dec. 1997), pp. 1735–80.

[69] Alexandre Défossez et al. “Music Source Separation in the Waveform Domain”. In: (2019). arXiv: 1911.13254 [cs.SD].

[70] Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. “Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram”. In: (2019). arXiv: 1910.11480 [eess.AS].

[71] Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. “Probability density distillation with generative adversarial networks for high-quality parallel waveform generation”. In: (2019). arXiv: 1904.04472 [eess.AS].

[72] Dario Rethage, Jordi Pons, and Xavier Serra. “A Wavenet for Speech Denoising”. In: (2017). arXiv: 1706.07162 [cs.SD].

[73] Pierre-Edouard Honnet et al. “The SIWIS French Speech Synthesis Database Design and recording of a high quality French database for speech synthesis”. In: (Jan. 2017).

[74] Jean-Philippe Goldman et al. “The SIWIS database: a multilingual speech database with acted emphasis”. In: (May 2016).

[75] David Snyder, Guoguo Chen, and Daniel Povey. “MUSAN: A Music, Speech, and Noise Corpus”. In: (2015).

[76] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. “VoxCeleb: a large-scale speaker identification dataset”. In: (2017).

[77] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. “VoxCeleb2: Deep Speaker Recognition”. In: (2018).

[78] Arsha Nagrani et al. “VoxCeleb: Large-scale Speaker Verification in the Wild”. In: *Computer Speech & Language* 60 (Oct. 2019), p. 101027.

[79] Cassia Valentini-Botinhao. “Noisy speech database for training speech enhancement algorithms and TTS models”. In: (2017).

[80] Mikko Kurimo et al. “Personalising Speech-To-Speech Translation in the EMIME Project.” In: (Jan. 2010), pp. 48–53.

[81] Aaron van den Oord et al. “WaveNet: A Generative Model for Raw Audio”. In: (2016).

[82] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. “Pixel Recurrent Neural Networks”. In: (2016). arXiv: 1601.06759 [cs.CV].

[83] Aaron van den Oord et al. “Conditional Image Generation with PixelCNN Decoders”. In: (2016). arXiv: 1606.05328 [cs.CV].

[84] Avinash Madasu and Vijjini Anvesh Rao. “Gated Convolutional Neural Networks for Domain Adaptation”. In: (2019). arXiv: 1905.06906 [cs.CL].

[85] Zhengyang Wang and Shuiwang Ji. “Smoothed Dilated Convolutions for Improved Dense Prediction”. In: (2018). arXiv: 1808.08931 [cs.CV].

[86] Kunjin Chen et al. “Convolutional Sequence to Sequence Non-intrusive Load Monitoring”. In: *The Journal of Engineering* 2018 (Sept. 2018).

[87] Sepp Hochreiter. “The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6 (Apr. 1998), pp. 107–116.

[88] Ke Tan and DeLiang Wang. “A Convolutional Recurrent Neural Network for Real-Time Speech Enhancement”. In: (June 2018).

[89] Zhendong Zhang, X. Wang, and C. Jung. “DCSR: Dilated Convolutions for Single Image Super-Resolution”. In: *IEEE Transactions on Image Processing* 28 (2019), pp. 1625–1635.

[90] Fisher Yu and Vladlen Koltun. “Multi-Scale Context Aggregation by Dilated Convolutions”. In: (2015). arXiv: `1511.07122 [cs.CV]`.

[91] Mahdi Hajibabaei and Dengxin Dai. “Unified Hypersphere Embedding for Speaker Recognition”. In: (2018). arXiv: `1807.08312 [eess.AS]`.

[92] Chao Li et al. “Deep speaker: an end-to-end neural speaker embedding system”. In: *arXiv preprint arXiv:1705.02304* 650 (2017).

[93] Koji Okabe, Takafumi Koshinaka, and Koichi Shinoda. “Attentive statistics pooling for deep speaker embedding”. In: *arXiv preprint arXiv:1803.10963* (2018).

[94] Joon Son Chung et al. “In defence of metric learning for speaker recognition”. In: (2020). arXiv: `2003.11982 [eess.AS]`.

[95] Feng Wang et al. “Additive Margin Softmax for Face Verification”. In: *IEEE Signal Processing Letters* 25.7 (July 2018), pp. 926–930.

[96] Hao Wang et al. “CosFace: Large Margin Cosine Loss for Deep Face Recognition”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (June 2018).

[97] Jiankang Deng et al. “ArcFace: Additive Angular Margin Loss for Deep Face Recognition”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019).

[98] Florian Schroff, Dmitry Kalenichenko, and James Philbin. “FaceNet: A unified embedding for face recognition and clustering”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2015).

[99] Li Wan et al. “Generalized End-to-End Loss for Speaker Verification”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (Apr. 2018).

[100] Jake Snell, Kevin Swersky, and Richard S. Zemel. “Prototypical Networks for Few-shot Learning”. In: (2017). arXiv: `1703.05175 [cs.LG]`.

[101] Ken Chatfield et al. “Return of the Devil in the Details: Delving Deep into Convolutional Nets”. In: *Proceedings of the British Machine Vision Conference 2014* (2014).

[102] Weicheng Cai, Jinkun Chen, and Ming Li. “Exploring the Encoding Layer and Loss Function in End-to-End Speaker and Language Recognition System”. In: *Odyssey 2018 The Speaker and Language Recognition Workshop* (June 2018).

[103] Joon Son Chung, Jaesung Huh, and Seongkyu Mun. “Delving into VoxCeleb: Environment Invariant Speaker Recognition”. In: *Odyssey 2020 The Speaker and Language Recognition Workshop* (Nov. 2020).

[104] Hyungeun Choi, Seunghyoung Ryu, and Hongseok Kim. “Short-Term Load Forecasting based on ResNet and LSTM”. In: (Oct. 2018), pp. 1–6.

[105] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: (2015). arXiv: 1502.03167 [cs.LG].

[106] Anthony Brunel et al. “A CNN adapted to time series for the classification of Supernovae”. In: (Jan. 2019).

[107] Zichao Yang et al. “Hierarchical Attention Networks for Document Classification”. In: (Jan. 2016), pp. 1480–1489. DOI: 10.18653/v1/N16-1174.

[108] Gautam Bhattacharya, Md Jahangir Alam, and Patrick Kenny. “Deep Speaker Embeddings for Short-Duration Speaker Verification”. In: (Aug. 2017), pp. 1517–1521. DOI: 10.21437/Interspeech.2017-1575.

[109] F A Rezaur Rahman Chowdhury et al. “Attention-Based Models for Text-Dependent Speaker Verification”. In: (2017). arXiv: 1710.10470 [eess.AS].

[110] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: (2019). arXiv: 1912.01703 [cs.LG].

[111] Martín Abadi et al. “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems”. In: (2016). arXiv: 1603.04467 [cs.DC].

[112] François Chollet et al. *Keras*. https://keras.io. 2015.

[113] Erfan Loweimi et al. “On the importance of pre-emphasis and window shape in phase-based speech recognition”. In: *International Conference on Nonlinear Speech Processing*. Springer. 2013, pp. 160–167.

[114] Man-Wai Mak and Hon-Bill Yu. “A study of voice activity detection techniques for NIST speaker recognition evaluations”. In: *Computer Speech & Language* 28.1 (2014), pp. 295–313.

[115] Daniel Povey et al. “The Kaldi speech recognition toolkit”. In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding* (Jan. 2011).

[116] Bohdan Pavlyshenko. “Using Stacking Approaches for Machine Learning Models”. In: Aug. 2018, pp. 255–258. DOI: 10.1109/DSMP.2018.8478522.

[117] Raman Arora et al. “Understanding deep neural networks with rectified linear units”. In: *arXiv preprint arXiv:1611.01491* (2016).

[118] Daniel Salles Chevitarese, Dilza Szwarcman, and Marley Vellasco. “Speeding Up the Training of Neural Networks with CUDA Technology”. In: *Artificial Intelligence and Soft Computing*. Ed. by Leszek Rutkowski et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 30–38. ISBN: 978-3-642-29347-4.

[119] T. Carneiro et al. “Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications”. In: *IEEE Access* 6 (2018), pp. 61677–61685.

[120] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[121] P. Krishnamoorthy. “An Overview of Subjective and Objective Quality Measures for Noisy Speech Enhancement Algorithms”. In: *IETE Technical Review* 28.4 (2011), pp. 292–301. DOI: 10.4103/0256-4602.83550. eprint: https://www.tandfonline.com/doi/pdf/10.4103/0256-4602.83550. URL: https://www.tandfonline.com/doi/abs/10.4103/0256-4602.83550.

[122] Y. Hu and P. C. Loizou. “Evaluation of Objective Quality Measures for Speech Enhancement”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 16.1 (2008), pp. 229–238.

[123] H. Levitt. “Digital Hearing Aids : Past , Present , and Future”. In: 2005.

[124] Jesper Boldt. “Binary masking & speech intelligibility”. PhD thesis. Ph. D. thesis, Aalborg Universitet, 2011.

[125] Nicoleta Roman and John Woodruff. “Intelligibility of reverberant noisy speech with ideal binary masking”. In: *The Journal of the Acoustical Society of America* 130.4 (2011), pp. 2153–2161.

[126] Joshua Feldman Eric Conrad Seth Misenar. *Eleventh Hour CISSP. Study Guide*. 3rd ed. Syngress, 2016.