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

Deep Learning for Audio Super Resolution

Relatore: Prof. Simone Bianco
Correlatore: Dott. Marco Buzzelli

Relazione della prova finale di:
Giorgio Bini
Matricola 838674

Anno Accademico 2019-2020

Abstract

Audio Super-Resolution is the problem of predicting the missing high-frequency content of a given signal from its low frequencies. Several recent studies have shown that Deep Learning algorithms are able to achieve remarkable results by modeling audio Super-Resolution as a regression task. A large variety of approaches have been proposed in literature, including convolutional and recurrent architectures to capture both local and long-term dependencies between audio frames. Furthermore, some research show that significant improvements may be achieved by processing the input signal not only in the time, but also in the frequency domain by exploiting the Fourier transform operations as an integral part of the neural network configuration. This thesis project aims not only to deal with the study of these approaches, but also to combine them in a principled way in order to explore a novel model architecture.

List of Abbreviations

A/D	Analog-to-Digital
AI	Artificial Intelligence
BiLSTM	Bidirectional LSTM
BWE	Bandwidth Extension
CNN	Convolutional Neural Network
dB	Decibel
DFT	Discrete Fourier Transform
DL	Deep Learning
DNN	Deep Neural Network
DSP	Digital Signal Processing
DTFS	Discrete-Time Fourier Series
DTFT	Discrete-Time Fourier Transform
FFT	Fast Fourier Transform
FiLM	Feature-Wise Linear Modulation
GAN	Generative Adversarial Networks
GRU	Gated Recurrent Unit
HR	High-Resolution
Hz	Hertz
IDFT	Inverse Discrete Fourier Transform

LR	Low-Resolution
LSD	Log-Spectral Distance
LSTM	Long Short-Term Memory
LTI	Linear Time-Invariant
ML	Machine Learning
MSE	Mean-Square Error
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SNR	Signal to Noise Ratio
SR	Super-Resolution
STFT	Short-Time Fourier Transform
STT	Speech-to-Text
TFiLM	Temporal Feature-Wise Linear Modulation
TFNet	Time-Frequency Network
VCTK	Voice Cloning Toolkit Corpus

List of Figures

List of Tables

Contents

Chapter 1

Introduction

In the last decades, the rapid development and the wide diffusion of digital circuits has made it possible to store, transmit and process any type of data. As a natural consequence of this technological progress, the scientific community has shown a growing interest in designing software tools capable of solving increasingly complex tasks. This is true for many areas of science that have been revolutionized by the advent of digital technologies, such as the signal processing. The lower cost of computational power and the possibility to design very sophisticated algorithms, have favoured the spread of digital in many signal processing applications in which analog circuits were once used. Indeed, it is often easier to operate precise mathematical operations on signals through computer software rather than in analog form. Moreover, the way of how signals are processed digitally has changed considerably over the years. Recently, deep learning-based approaches have demonstrated state of-the-art results in many challenging applications for both one-dimensional (such as audio) and two-dimensional signals (for instance, images). Some of the best-known techniques for DL modeling will be examined in the context of a specific problem, such as audio super-resolution.

Audio SR refers to the task of reconstructing HR audio data from LR, down-sampled input sequences containing only a small fraction (15-50%) of the original samples. Several factors have determined the choice of this topic. First, this task is useful for a wide variety of applications in many sectors, ranging from telecommunications to many other domains. For example, super resolution techniques may be applied with regard to the data compression problem. This task consists of two main phases: encoding and decoding. First, an encoder object is used to compress audio signals with a downsample-based approach. In the second phase, a decoder is utilized in order to upsample data with a super resolution algorithm. This procedure is highly useful to save storage space and can

be also used to reduce data transmission time in online applications.
Another interesting use case can be found in the use of this model for the audio restoration task. The objective of this problem consists in reconstructing high quality signals from audio files recorded with low fidelity equipment. In this sense, audio super resolution may also come in help in forensics analysis by improving clarity of recordings. In addition, companies may also consider the application of this algorithm to save on the use of expensive signal acquisition devices. More and more often, software technologies can address physical limitations of hardware components or improve their performance through effective data processing.
Other possible uses could be concerned with the integration of this model with other AI applications. For example, one might think of the audio super-resolution algorithm as a useful tool that may come in help for the speech-to-text conversion task, in which another DL model could be facilitated in the learning process through the use of more precise and informative input data. An analogous example can concern the opposite problem, i.e. text-to-speech, where a SR system has the potential to improve the perceived quality of an artificial audio file created by another deep learning model.
Audio super resolution can be useful for any kind of audio signal. However, for this thesis project, it is decided to refer univocally to data concerning the speech domain, which is of particular interest for many of the previously listed applications.

1.1 Problem formulation

The best way to avoid introducing any potential ambiguity is to define formally the problem, both from a notational and terminological point of view.
Super-resolution refers to the task of reconstructing HR data from LR observations. If we denote with Y the high-dimensional signal and with X the LR input, we could see the audio SR model as a mapping function f_θ, parametrized by θ, with the following form:

$$f_\theta : X \to Y$$

The above definition can be extended to include any kind of signal, such as audio or images. Without losing generality, it is possible to define a signal as a physical quantity that varies with time, space, or any other independent variable [15]. However, since the object of study is limited to the audio signals domain, the focus is restricted to that type of data.

Audio signals can be further classified into various categories: continuous (i.e., analog) versus discrete, continuous-valued versus discrete-valued. For this study we will refer specifically to digital audio signals, i.e. discrete-valued measurements of continuous-time quantities obtained by a sampling process.

The preceding formulation directly leads to a specific definition of the elements $x \in X$ and $y \in Y$. It is possible to define $x = (x_{1/R_1}, x_{2/R_1}, \ldots, x_{R_1 S/R_1})$ as a one-dimensional vector sampled at rate R_1, where S is the duration of the signal (in seconds). It follows that number of samples is defined as $T_1 = SR_1$. The target sequence $y = (y_{1/R_2}, y_{2/R_2}, \ldots, y_{R_2 S/R_2})$ has a higher sampling rate $R_2 > R_1$, and that makes higher the temporal dimension too, i.e. $T_2 > T_1$, where $T_2 = SR_2$. We can define $r = \frac{R_2}{R_1}$ to denote the upsampling ratio of the two signals, which in this work is equal to $r = 4$. The main idea of this problem formulation is that $\hat{y} = f_\theta(x)$, where $\hat{y} = (\hat{y}_{1/R_2}, \hat{y}_{2/R_2}, \ldots, \hat{y}_{R_2 S/R_2})$ is the reconstruction of the HR signal. To obtain the target estimation, a supervised learning model is trained on a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$ composed by LR, HR pairs. The model f_θ, that in this case is a DL algorithm, must predict the target observation y from its down-sampled version x, or more generally predict the conditional probability distribution $p(y|x)$.

As for terminology, it is important to mention that the problem of audio super-resolution is studied under the name of BWE [10] by the audio processing community, so these two terms are used as synonyms in the remainder of the thesis. Furthermore, since the data used refer to speech, it would also be correct to speak about *speech bandwith extension* [45]. However, the adoption of a more generic terminology is preferred, also because the techniques used in this project can theoretically be extended to other categories of audio signals, such as music or environmental sounds. In general, it is fair to say that this application can be extended for all signals in which the high-frequency part is considered highly dependent on its corresponding low-frequency counterpart, as in the case of speech.

Another term that is often used in this thesis is the one of *system*. A system may be defined as a device that performs an operation on a signal. The concept of system includes not only physical devices, but also software realizations of operations on a signal [15]. Systems can be classified differently depending on their properties. For example, linear systems perform only linear operations on the input signal. On the contrary, if these operations are nonlinear, such as the processing of a neural network, the system is said to be nonlinear.

1.2 Related work

The problem of audio super resolution has been widely studied in the past few decades. Existing data-driven approaches have lead to significant improvements in the context of SR problems with a large variety of methods. Even if non-parametric models have shown good results [11], most methods in the literature are characterized by the use of parametric models, such as DL, Gaussian Mixture Models [6], [9], [18] or Hidden Markov Models [12], [16].

Another level of categorization between existing approaches is the following: domain-dependent and domain-agnostic. The former aim to restore high-frequency information from low sample rate audio by the extraction of handcrafted features and designing ad-hoc learning strategies (see e.g., [18]). More recently, the algorithms used to model audio allows the direct use of raw data as input. This is another advantage of DL models: they are often effective even without a manual feature extraction. Furthermore, many of these DL algorithms are fully domain-agnostic, i.e. their use can be extended for different type of task, such as Text Classification (see e.g. [46]).

Deep learning-based approaches for audio SR have changed considerably over the last few years. Traditional Deep Neural Network (DNN)-based techniques [28] have been superseded by CNN models [39]. The latest improvements are associated with the use of RNN [46] and Generative Adversarial Networks (GAN) [48]. The former can take into account the sequential nature of the audio data, the latter is particularly suitable for generative tasks. Although GANs yield remarkable results in many digital signal processing problems, they suffer from instabilities during the training phase [43]. Because of this intrinsic difficulty, this family of models is not included in the rest of the study.

It is also important to mention that a satisfactory result for this task has been obtained by the use of auto-regressive models, such as WaveNet [45].

Results of recent papers indicate that there is not a clear winner between all these different methods. This is also due to the fact that, when comparing models, many aspects must be considered, such as computational time (on which a possible application in real-time depends). However, there are some standard evaluation metrics currently used in SR problems that can be used for comparison. All the aspects related to model evaluation are explained in detail in the body of the work.

1.3 Research Objectives

The study of recent literature papers has advanced the idea of combining different methods that performed well in other studies. Indeed, the key thrust of this thesis is on the implementation of a novel model architecture inspired by some of the state-of-the-art techniques. The two studies from which most of the proposed methods in this work derives are the following:

- **Time-frequency networks for audio super-resolution** [42]: This work introduces TFNet, a deep neural network that consists of two branches with similar architectures; a time domain branch and a frequency domain branch, which explicitly models the reconstruction's spectral magnitude. Each domain is modeled jointly using design patterns from AudioUNet [39], a fully convolutional model consisting of encoder and decoder blocks.

- **Temporal FiLM** [46]: Birnbaum, Kuleshov *et al.* propose TFiLM, an innovative neural network component that uses a RNN to alter the activations of a convolutional model. This approach aims to capture long-range information in sequential data processing by the adoption of recurrent layers that can expand the receptive field of CNNs.

The proposed architecture is intended to directly exploit the best of both works, in order to explore a novel model configuration. The key idea is to replace the AudioUNet components used by Lim, Yeh *et al.* with the TFiLM layers cited above. The model maintains the branching structure that allows to process audio in both time and frequency domain. A detailed exposition of the most interesting features is covered in detail in Chapter 3.

1.4 Thesis Organization

This work is organised into five chapters, including the present one. A brief overview of the thesis structure is provided below.

- **Chapter 2** treats the main theoretical aspects that need to be covered in addressing the audio SR problem by using DL algorithms to process data. These concepts include basic elements of Digital Signal Processing (DSP), ML and DL.

- **Chapter 3** is devoted entirely to the characterization and analysis of the proposed model. All the main features are highlighted in detail to provide a precise and comprehensible description.

- **Chapter 4** describes the experiments performed and the results obtained. A particular focus is given to description of the dataset as well as the evaluation metrics used.
- **Chapter 5** we conclude the thesis and discuss possible future directions to advance the field.

Chapter 2

Preliminary Concepts of Signal Processing and Deep Learning

This chapter briefly summarises the theoretical aspects which lie at the basis of this work. These concepts concern both DSP and DL fields. On one hand, it is necessary to know the pertinent elements that characterize the reference domain, i.e. digital audio signals. On the other hand, it is important to introduce the main aspects of deep learning models, since they are the main computational systems used for processing audio signals in this project.
In order to avoid any unintended confusion or ambiguity, first it is important to say that in this chapter we use a different notation, with respect to the one adopted in the rest of the work, for defining some quantities of interest. This choice is motivated by the fact that notation is determined by the context, which in this chapter is purely theoretical, while in the others is applied to a very specific domain.

2.1 Digital Signal Processing

The main objective of this paragraph is to present the theoretical fundamentals related to DSP. Since the focus is on the audio domain, all the examples deal mainly with one-dimensional digital signals.
Before dealing with DSP, we shall briefly introduce a unit of measure which is of fundamental importance for this domain: the Decibel.
Formally, dBis a logarithmic measure that evaluates the ratio between two homogeneous quantities (two powers, two energies, etc...). The formula for expressing a ratio Q in dB

is as follows:

$$Q(dB) = 10 \log_{10}(Q) \tag{2.1}$$

where Q is a ratio between two homogeneous quantities (which share the same unit of measure).

2.1.1 Classification of Signals

Signals can be categorized according to various criteria: *multi-channel* vs *single-channel*, *real-valued* vs *complex-valued*, *one-dimensional* vs *multi-dimensional*, *continuous-valued* vs *discrete-valued*. It is important to define such a categorization because some techniques can only be applied to specific families of signals. The objective of this paragraph is to outline in a comprehensible and exhaustive way the main families of signals.

As mentioned in Chapter 1, it is possible to define a signal x as a function of one or more independent variables. Formally, we can consider $x : \mathcal{A} \to \mathcal{B}$ where $\mathcal{A}$ and $\mathcal{B}$ are subsets of real vector spaces of dimension $\mathbb{R}^m$. The values of the domain $\mathcal{A}$ usually refer to spatial and/or temporal coordinates, while the codomain $\mathcal{B}$ denotes values assumed by physical quantities such as pressure, currents. For example, audio signals such as speech and music concern variations of air pressure over time.

According to [15], we can consider different families of signals, depending on whether or not their domain and codomain vector spaces are real: a first distinction can be made between "continuous-time" and "discrete-time" signals.

Continuous-time ones are defined for every value of time, so they take on values over the interval $(-\infty, \infty)$. The functions $x_1(t) = \cos \pi t, x_2(t) = e^{-|t|}\ -\infty < t < \infty$ are good examples of this class of signals.

On the other hand, we have functions which are defined only at discrete instants of time, i.e. discrete-time signals. They may arise by a sampling process that takes place over *analog* signals at discrete time instants. The main factors that characterize the sampling operation are discussed in more detail in 2.1.2. In many practical applications, discrete-time signals are obtained by a periodically sampling of analog signals, such that time instants are equidistant. The signal $x\left(t_n\right) = e^{-|t_n|}, n = 0, \pm 1, \pm 2, \ldots$ is an example of a discrete-time signal. For the rest of the work, we use the index n to emphasize the discrete-time nature of a signal as $x(n)$ instead of $x(t)$.

Furthermore, values in the codomain $\mathcal{B}$ can be either continuous or discrete. In particular, if a signal takes on all possible values on a certain range, it is said to be a continuous-valued signal. Alternatively, if the signal is obtained by quantizing its values to a finite set of discrete values, it is said to be a discrete-valued signal.

In summary, it is possible to distinguish the following classes of signals:

- $\mathcal{A} \in \mathbb{R}$, $\mathcal{B} \in \mathbb{R}$: continuous-time and continuous-valued signals, also known as *analog* signals;
- $\mathcal{A} \in \mathbb{R}$, $\mathcal{B} \in \mathbb{Z}$: *quantized analog* signals, i.e. continuous-time and discrete-valued signals;
- $\mathcal{A} \in \mathbb{Z}$, $\mathcal{B} \in \mathbb{R}$: discrete-time and continuous-valued signals, also known as *sampled* signals;
- $\mathcal{A} \in \mathbb{Z}$, $\mathcal{B} \in \mathbb{Z}$: *digital* signals, i.e. discrete-time signals having a set of discrete values. In this work we only deal with digital signals since they are the only ones that can be processed by a computer.

It is important to mention that, for simplicity of notation, in the above definitions we refer to the discrete set of values with the integer symbol $\mathbb{Z}$. A graphical example of this categorization is provided in Figure 2.1.

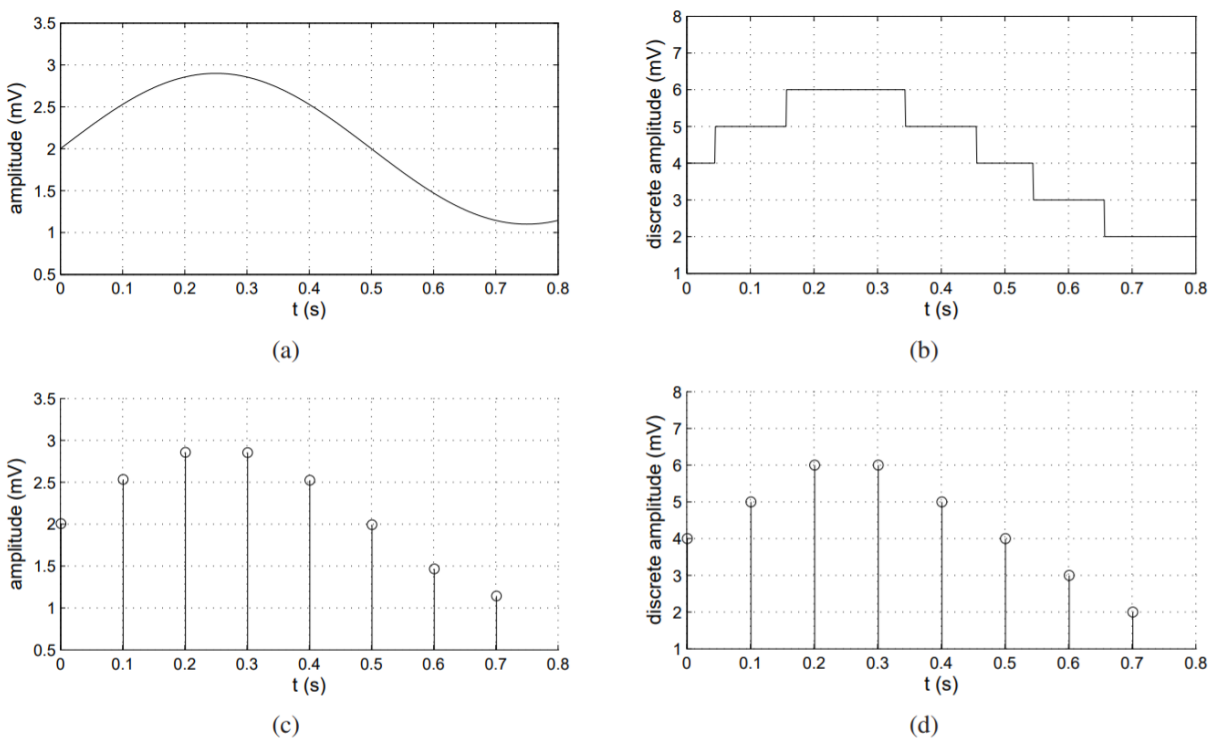

Figure 2.1: (a) *Analog*, (b) *quantized-analog*, (c) *sampled*, and (d) *digital* signals. From [14].

Furthermore, we can distinguish *one-dimensional* and *multi-dimensional* signals depending on the vector space dimension of $\mathcal{A}$. In particular, a signal is called one-dimensional if its value is a function of a single independent variable. An example of this class of signals is the audio, which is function of time. On the other hand, if the signal is a function of M independent variables, the signal is called a M-dimensional one. For example, images are function of two spatial coordinates.
Another distinction can be made between *single-channel* and *multi-channel* signals. Examples of the former category are mono recordings or black and white images, while stereo audio clips and RGB digital images are examples of the latter. In this thesis project we work mainly on single-channel, one-dimensional signals.

2.1.2 From Analog to Digital

As previously motivated, this work deal mainly with digital signals. That means they are a sequence of data bits, obtained through a process that allows to convert signals from their original analog form to a digital one. According to [15], this procedure is called Analog-to-Digital (A/D) conversion, and the corresponding devices are called A/D converters.
It is possible to divide this process into three main steps:

1. *Sampling.* This is the conversion of a continuous-time signal into a discrete-time one.

2. *Quantization.* It is basically an approximation process that consists in converting a continuous-valued signal into a discrete-valued one.

3. *Coding.* In this process, each discrete value is represented by a b-bit binary sequence.

This section aims to deepen these three steps, since they are fundamental in plenty of DSP applications. The main reference for this explanation is [15].
As previously mentioned, sampling is the first step of the A/D conversion. This can be done in many ways, but in many practical applications a discrete-time signal $x(n)$ is obtained by a *periodic* or *uniform* sampling of analog signals. If we denote with $x_a(t), \quad -\infty < n < \infty$ the analog signal, then the discrete-time signal can be calculated as follows:

$$x(n) = x_a(nT), \quad -\infty < n < \infty \tag{2.2}$$

where T are the seconds between successive samples. This time interval T is called the *sampling period* or *sample interval* and its reciprocal $1/T = F_s$ is called the *sampling rate* (samples per second) or the *sampling frequency*, that is commonly measured in Hz (cycles per second). Equation 2.2 establishes a linear relationship between the time variables t and n, that can be expressed as:

$$t = nT = \frac{n}{F_s} \tag{2.3}$$

A graphical example of the periodic sampling is given in Figure 2.2.

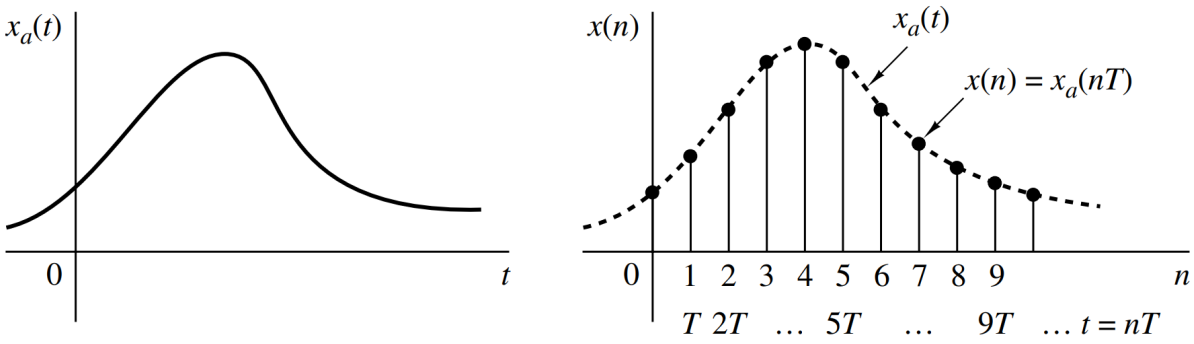

Figure 2.2: Periodic sampling of an analog signal. From [15].

At this point, it is important to point out how the sampling rate F_s is selected in real DSP applications. The acquisition of analog signals is typically driven by some prior knowledge about the characteristics of the signal to be sampled. In particular, this information concern the frequency content of general class of signals (e.g., the class of speech signals, the class of video signals, etc.), and they are generally available. For example, we know that human voice frequencies are, for the most part, below 3000 Hz. By knowing this information, appropriate sampling rate can be selected to avoid the problem commonly called *aliasing*.

The latter occurs when the sampling rate is not sufficiently high to capture the higher frequencies of the original signal. The most common criterion to determine the sampling rate necessary to convert analog data to digital is given by the *sampling theorem*, which was introduced by Nyquist (1928) and later popularized by Shannon (1949). This popular theorem states that it is possible to completely recover an analog signal $x_a(t)$ from its sample values if:

$$F_s > 2 \times F_{max} \tag{2.4}$$

where F_{max} is the largest frequency component in the analog signal. It is important to note that minimal sampling frequency that allows perfect reconstruction of a bandlimited signal from its samples, corresponds to $2 \times F_{max}$. This sampling frequency is known as *Nyquist rate*.

Thus, if the sampling rate is chosen according to the sampling theorem it is possible to avoid the problem of aliasing. A graphical example of the aliasing effect is given in Figure 2.3.

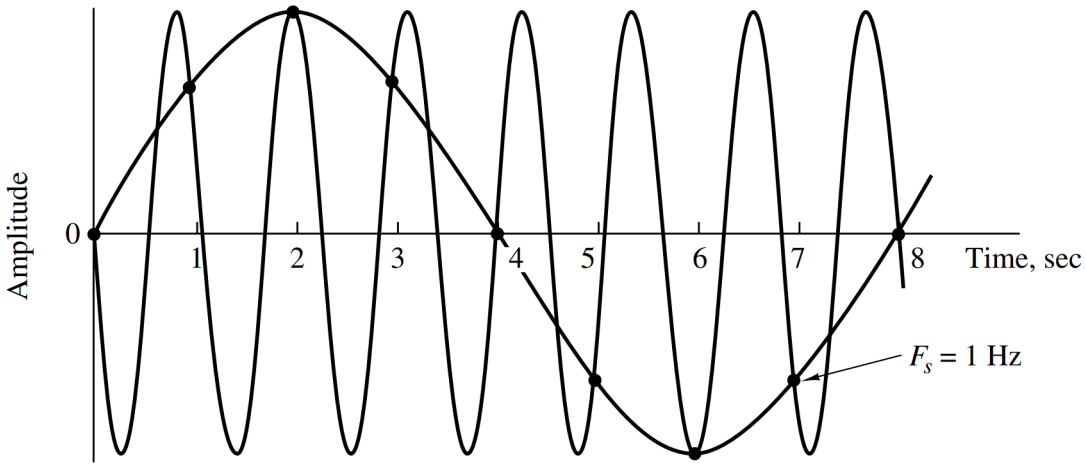

Figure 2.3: Illustration of aliasing; the sampling frequency $F_s = 1$ Hz is not sufficiently high to unambiguously reconstruct the original sinusoid. From [15].

Usually, after the sampling process, the quantization is performed. Briefly, quantization is the irreversible operation of converting a discrete-time continuous-amplitude signal into a digital one by expressing each sample value as a finite (instead of an infinite) number of digits. This results in signal distortion: it introduces the so called *quantization error* which measures the difference between the quantized values and the actual sample ones. Formally, let $x_q(n)$ denote the sequence of quantized samples at the output of the quantizer $Q[x(n)]$. Hence

$$x_q(n) = Q[x(n)] \tag{2.5}$$

Then it is possible to define the quantization noise $e_q(n)$ as follows:

$$e_q(n) = x_q(n) - x(n) \tag{2.6}$$

At this point, we can take an example to explain both the sampling and quantization

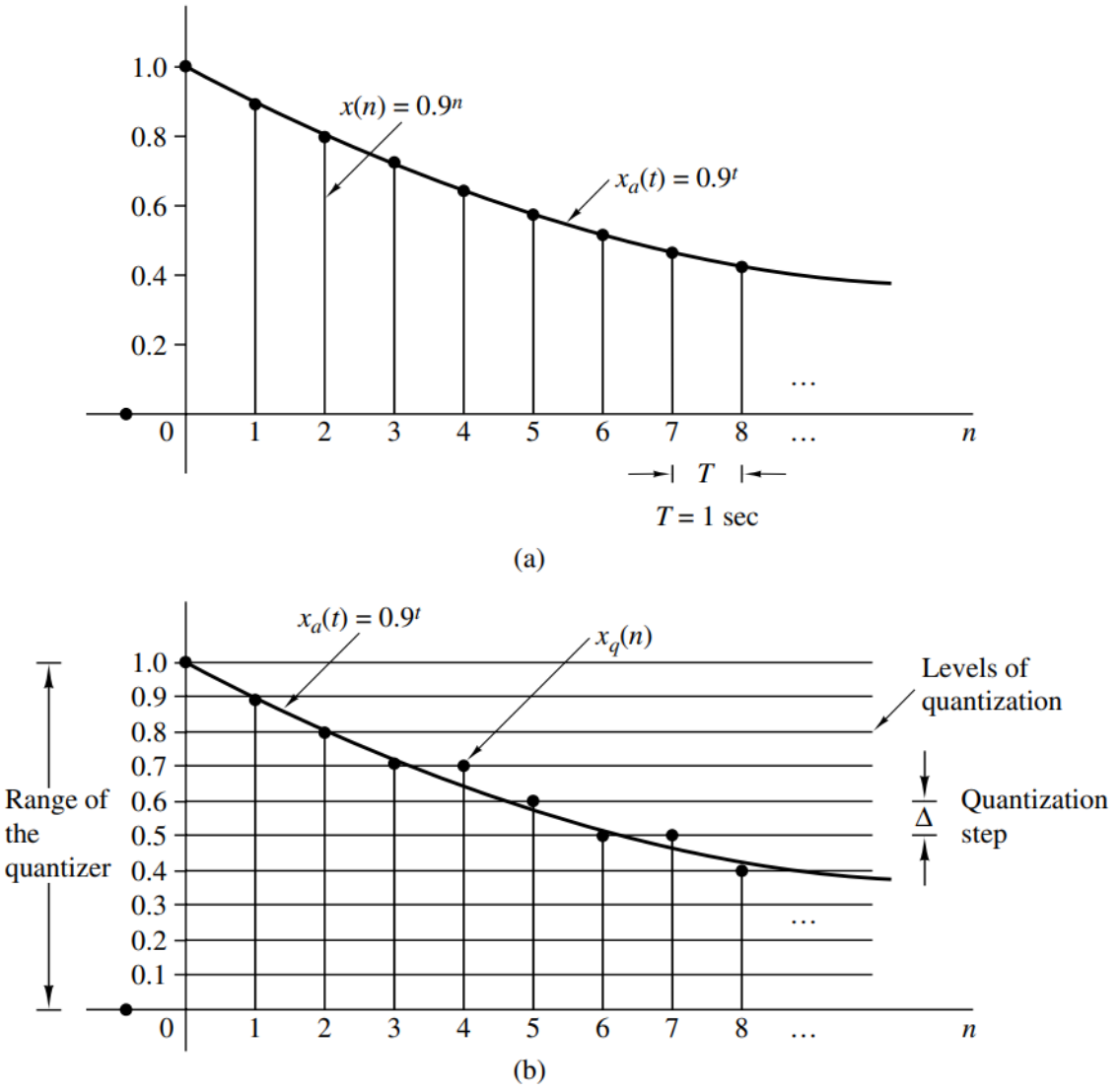

Figure 2.4: Graphical example of quantization. From [15].

processes more explicitly. First, consider the analog exponential signal

$$x_a(t) = \begin{cases} 0.9^t, & t \geq 0 \\ 0, & t < 0 \end{cases}$$

Let $x(n) = 0.9^n, n \geq$ be the discrete-time signal obtained by the sampling process with $F_s = 1$ Hz (see Fig. 2.4 (a)). The resulting samples of $x(n)$ are shown in Table 2.1. As we can see, in this example, a complete description of the sampled signal requires n significant digits. A precise description of each digit would be computationally heavy for most computers, and useless for many real applications. Thus, excess digits can be discarded by rounding the resulting number, as we can see in Figure 2.4 (b) and in Table 2.1. It is important to mention that the distance Δ between two successive quantized values is called *quantization step*.

n	$x(n)$ Discrete-time signal	$x_q(n)$ (Rounding)	$e_q(n) = x_q(n) - x(n)$ Quantization Error
0	1	1.0	0.0
1	0.9	0.9	0.0
2	0.81	0.8	-0.01
3	0.72	0.7	-0.029
4	0.6561	0.7	0.0439
5	0.59049	0.6	0.00951
6	0.531441	0.5	-0.031441
7	0.4782969	0.5	0.0217031
8	0.43046721	0.4	-0.03046721
9	0.387420489	0.4	0.012579511

Table 2.1: Numerical illustration of quantization with one significant digit using rounding. From [15].

As previously mentioned, the last step of A/D converters consists in the coding process. Thus, each sample is changed to an n bit code, such that each quantization level is assigned to a unique binary number.

2.1.3 About Sinusoids

Sinusoids are the building block of signal processing. As is described later in this chapter, all signals can be decomposed into constituent sinusoids of different frequencies using Fourier analysis.
Formally, it is possible to express a discrete-time sinusoidal signal as:

$$x(n) = A\sin(\omega n + \phi) = A\sin(2\pi f n + \phi) = A\sin(\frac{2\pi}{\tau}n + \phi), \qquad -\infty < n < \infty \quad (2.7)$$

The above definition introduces the following parameters [47]:

- The amplitude A, which scales the range of values of the sinusoid from $[-1, 1]$ to $[-A, A]$.
- The frequency ω in radians per sample. Note that $\omega = 2\pi f$, where f is the frequency in cycles per sample. The frequency is inversely proportional to the period $\tau = \frac{1}{f}$, which measures the time the sinusoids takes to perform an entire cycle.
- ϕ is the phase in radians. It can be thought of as a time-shift of the sinusoid.

It is important to note that sine and cosine are orthogonal waveform functions, i.e. they are out of phase by $\phi = \frac{\pi}{2}$. Therefore, from a terminological point of view, we refer to

both cosine and sine functions as simply sinusoids.

A graphical example of a discrete-time sinusoid is given in Figure 2.5.

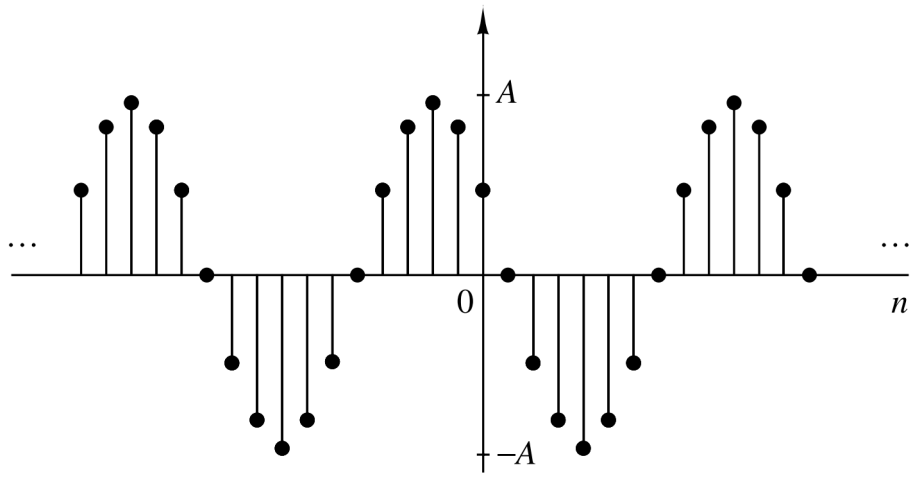

Figure 2.5: Example of a discrete-time sinusoidal signal ($\omega = \frac{\pi}{6}$ and $\phi = \frac{\pi}{3}$). From [15].

It can be demonstrated (see [15]) that discrete-time sinusoids are characterized by the following properties:

- A discrete-time sinusoid is periodic only if $f \in \mathbb{Q}$.
- Discrete-time sinusoids whose frequencies are separated by an integer multiple of 2π are identical. From this property it follows that any sinusoid with a frequency $|\omega| > \pi$, or $|f| > \frac{1}{2}$, is an alias of a corresponding sequence resulting from a sinusoid with frequency $|\omega| < \pi$. For this reason, the range $0 \leq \omega \leq 2\pi$ or $-\pi \leq \omega \leq \pi$ ($0 \leq f \leq 1$, $-\frac{1}{2} \leq f \leq \frac{1}{2}$) is commonly defined as the *fundamental range*.
- The highest rate of oscillation in a discrete-time sinusoid is attained when $|\omega| = \pi$ or, equivalently, $|f| = \frac{1}{2}$.

Complex sinusoids

Complex sinusoids, (or *complex exponentials*) are the building blocks of any real-world signal: as it is stated in Section 2.1.5, it is possible to decompose (or, at least, approximate) most signals of practical interest as a weighted sum of complex exponentials. Thus, it is important to introduce the class of these complex objects.

According to [15], we can define a complex exponential $x(n)$ as:

$$x(n) = Ae^{j(\omega n+\phi)} = A\cos(\omega n + \phi) + jA\sin(\omega n + \phi), \qquad -\infty < n < \infty \tag{2.8}$$

where A is the amplitude and $j \in \mathbb{C}$ is the *imaginary unit*. The link between exponential and trigonometric form of the above relation derives from the Euler identity, which is expressed as follows.

$$e^{\pm j\theta} = \cos\theta \pm j \sin\theta \tag{2.9}$$

2.1.4 Classification of Discrete-Time Signals

The techniques used in DSP depend heavily on the characteristics of the signal. Consequently, it is important to categorize signals based on their attributes. According to [15], important classes of discrete-time signals are the following.

- *Energy signals.* Consider a discrete-time signal $x(n)$, and let define the energy E as:

$$E = \sum_{n=-\infty}^{\infty} |x(n)|^2 \tag{2.10}$$

  If E is finite, then $x(n)$ is an energy signal.

- *Power signals.* The average power of $x(n)$ can be defined as:

$$P = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} |x(n)|^2 \tag{2.11}$$

  If P is finite, then $x(n)$ is an energy signal. Note that if E is finite, $P = 0$, but if E is infinite, P may be either finite or infinite.

- *Periodic signals.* A discrete-time signal $x(n)$ is periodi with period $N(N > 0)$ if:

$$x(n+N) = x(n) \qquad \forall n \tag{2.12}$$

  The smallest value of N for which this statement is true is called the fundamental period. It can be demonstrated (see [15]) that periodic signals are power signals. On the other hand, it is also worth noting that the energy of a periodic signal over a single period is finite if $x(n)$ takes on finite values over the period.

- Even and odd signals. An even (or symmetric) signal $x(n)$ can be defined as a

sequence with the following property:

$$x(-n) = x(n) \tag{2.13}$$

On the other hand, a signal $x(n)$ is called odd (or, equivalently, antisymmetric) if:

$$x(-n) = -x(n) \tag{2.14}$$

Graphical examples of symmetric and antisymmetric signals are given, respectively, in Figure 2.6 (a) and (b).

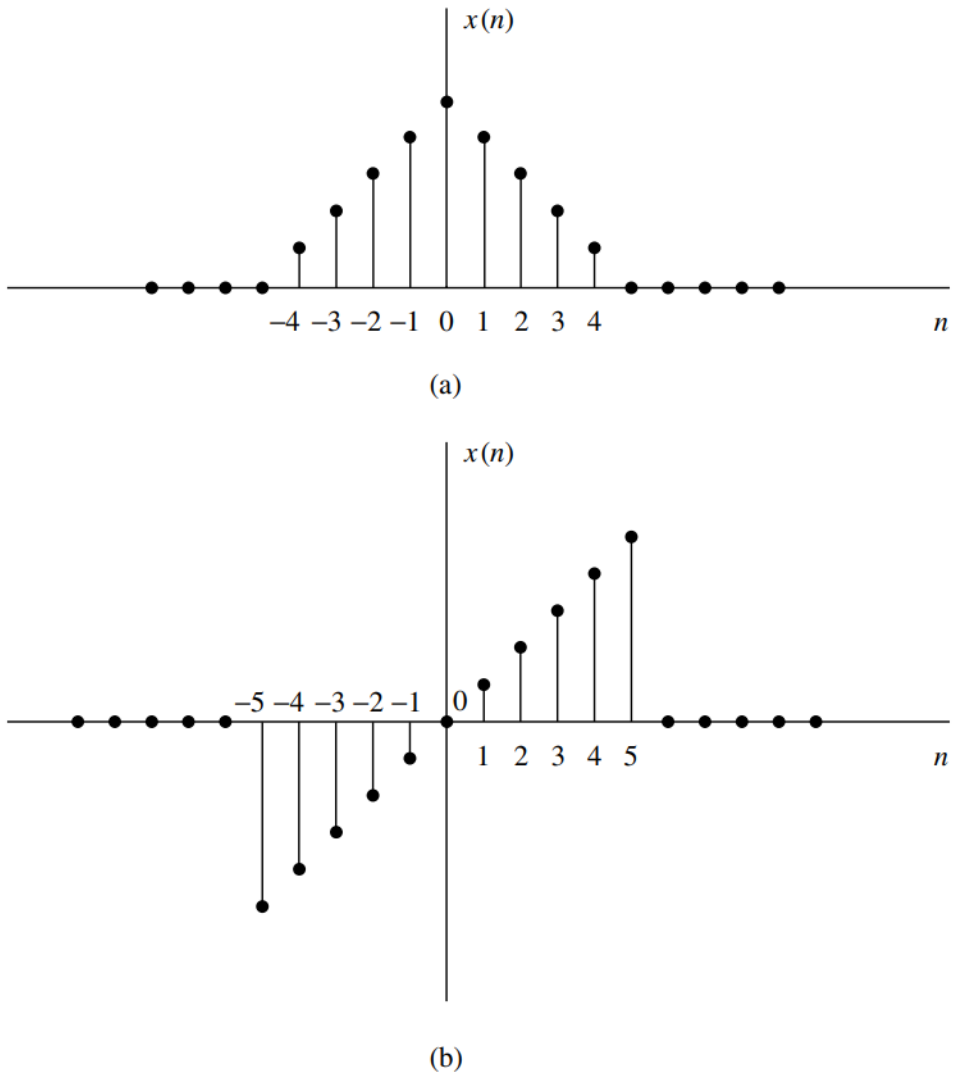

Figure 2.6: Example of even (a) and odd (b) signals. From [15].

2.1.5 Spectral Analysis of Signals

Frequency analysis is a powerful tool that is widely used in a large variety of engineering and signal processing tasks. Briefly, the spectral representation aims to highlight how the energy is distributed across frequency components of the signal. Furthermore, the spectral representation is particularly suited for performing specific classes of operations, such as the convolution.

As previously mentioned, signals can be decomposed into a weighted sum of sinusoidal signal components (or complex exponentials). The decomposition techniques depend on the characteristic attributes of the specific signal. Indeed, there are frequency analysis tools that apply only to specific families of signals. In particular, periodic signals can be decomposed through *Fourier series*; on the other hand, for the class of finite energy signals, the decomposition is called the *Fourier transform*.

This section aims to explain the difference between these tools, by referring mainly to the book [15]. Since this project deals only with digital signals, only definitions concerning discrete time-signals are discussed.

Discrete-Time Fourier Series

Fourier series are used to decompose periodic sequences into a weighted sum of frequency components. Formally, suppose to have a discrete-time periodic signal $x(n)$ with period N such that $x(n) = x(n+N) \quad \forall n$. The Fourier series for that signal can be expressed as:

$$x(n) = \sum_{k=0}^{N-1} c_k s_k(n) \tag{2.15}$$

where c_k are the coefficients in the series representation and s_k consists of N harmonically related exponential functions, such that:

$$s_k(n) = e^{j2\pi n \frac{k}{N}}, \qquad k = 0, 1, \ldots, N-1 \tag{2.16}$$

The set defined in 2.16 is composed of periodic complex exponentials which share fundamental frequencies that are multiples of a single positive frequency term. It is noticeable that, from a terminological point of view, the sequence $s_k(n)$ is called the k-th *harmonic* of $x(n)$.

The relationship 2.15 is called *Synthesis equation* or, equivalently, Discrete-Time Fourier

Series (DTFS). As for the Fourier coefficients c_k, they represents the amplitude and phase associated with the frequency component $s_k(n)$. In other words, these coefficients provide the description of the periodic signal $x(n)$ in the frequency domain. Their definition is provided by the *Analysis equation* expressed in 2.17.

$$c_k = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi kn}{N}}, \qquad k = 0, 1, \ldots, N-1 \tag{2.17}$$

Furthermore, it can be demonstrated that the spectrum of a periodic signal $x(n)$ with period N, is a periodic sequence with period N. Consequently, it is sufficient to observe any N consecutive samples belonging to the signal in either time or frequency domain to provide its complete description.
Lastly, it is important to highlight that real periodic signals have interesting symmetry properties. In particular, we know that for a real periodic signal, the spectral magnitude has an even symmetry property, such that:

$$|c_{-k}| = |c_k|$$

As for the spectral phase, we know that:

$$\measuredangle c_{-k} = \measuredangle c_k$$

As a direct consequence of these properties, we can conclude that it is possible to specify the whole signal in the frequency domain by taking into account only $\lfloor \frac{N}{2} \rfloor$ consecutive values.

Discrete-Time Fourier Transform

Aperiodic signals can be decomposed through Fourier Transforms. From the DSP theory [15] we know that, given a discrete-time signal $x(n)$ with finite energy, it is possible to represent its frequency content by the analysis equation 2.18.

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\omega n} = \sum_{n=-\infty}^{\infty} x(n) e^{-j2\pi fn} \tag{2.18}$$

From a notational point of view, we can write that:

$$x(n) \overset{\mathscr{F}}{\longleftrightarrow} X(\omega)$$

to denote that $X(\omega)$ is the Fourier Transform of $x(n)$.
The formula in 2.18 is also called Discrete-Time Fourier Transform (DTFT). From a conceptual point of view, $X(\omega)$ represents the decomposition of $x(n)$ into its frequency components.
As in the case of DTFS, we can define a synthesis equation (2.19) too.

$$x(n) = \frac{1}{2\pi}\int_{2\pi} X(\omega)e^{j\omega n}d\omega = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(\omega)e^{j\omega n}d\omega \tag{2.19}$$

It is worth noting that, since DTFT is a periodic function of the frequency variable ω, it has a Fourier series expansion. Therefore, its Fourier coefficients are the values of $x(n)$. In other words, the synthesis equation just written can be considered as an analysis equation of the Fourier Series of $X(\omega)$. In fact, the DTFT of $x(n)$ results in a periodic signal, with period 2π. This is demonstrated below.

$$X(\omega + 2\pi k) = \sum_{n=-\infty}^{\infty} x(n)e^{-j(\omega+2\pi k)n} = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}e^{-j2\pi kn} = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} = X(\omega)$$

Also in this case there are some important symmetry properties. It is possible to demonstrate ([15]) that if $x(n)$ is real, then:

$$|X(-\omega)| = |X(\omega)| \qquad \text{(even symmetry)}$$

$$\measuredangle X(-\omega) = -\measuredangle X(\omega), \qquad \text{(odd symmetry)}$$

From these relations, we can conclude that it is possible to determine the frequency content of $x(n)$ from the range of values $0 \leq f \leq \frac{1}{2}$ (or, equivalently $0 \leq \omega \leq \pi$) rather than $-\frac{1}{2} \leq f \leq \frac{1}{2}$ ($-\pi \leq \omega \leq \pi$). In other words, if we know $X(-\omega)$ in the range $0 \leq \omega \leq \pi$ we can calculate $X(-\omega)$ over $-\pi \leq \omega < 0$. Summing up, the whole description of a discrete-time real signal $x(n)$ in the frequency domain is determined over the continuous range of values $0 \leq \omega \leq \pi$.

Discrete Fourier Transform

The Discrete Fourier Transform (DFT) is a computational tool that plays a very important role in a wide variety of DSP applications, such as linear filtering, spectrum analysis and power frequency estimation.

As previously motivated, Fourier transform aims to perform frequency analysis of a signal by decomposing it into different frequency components. Such a spectral representation leads to a discrete function in the case of DTFS, i.e. when the input $x(n)$ is periodic. On the other hand, DTFT on aperiodic signals is a continuous function of frequency; therefore, from a computational point of view, it is not a convenient representation of the signal spectrum.

For this reason, it is important to introduce DFT, which is a powerful tool for representing a signal in the spectral-domain in a computationally efficient manner.

Formally, if we have a signal $x(n)$ of length N, it is possible to represent its frequency content with the following equation.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi kn}{N}}, \quad k = 0, 1, 2, \ldots, N-1 \tag{2.20}$$

Conceptually, the DFT relation in 2.20 expresses the Fourier transform as a function of a N-length equally spaced set of discrete frequencies. Indeed, it is possible to observe that the function is parametrized by the use of a discrete variable $k \in [0, N] \subset \mathbb{Z}$ rather than a continuous one $\omega \in [0, 2\pi] \subset \mathbb{R}$.

The signal $x(n)$ can be recovered from its frequency samples by the N-point Inverse Discrete Fourier Transform (IDFT) relation in 2.21.

$$x(n) = \frac{1}{N} \sum_{n=0}^{N-1} X(k) e^{j\frac{2\pi kn}{N}}, \quad n = 0, 1, 2, \ldots, N-1 \tag{2.21}$$

It is possible to demonstrate (see [15]) that if we have a finite-duration sequence $x(n)$ of length $N_{lim} < N$, then the IDFT yields $x(n) = 0$ for $N_{lim} \leq n \leq N-1$. Otherwise, if the sequence $x(n)$ is time-limited by a value N_{lim} such that $N_{lim} > N$, the N-point IDFT results in an aliased version of the original signal.

DFT is a powerful tool which has many properties. The objective of this section is not to provide an exhaustive analysis of these properties, for which we refer to excellent textbooks (such as [15]); thus, our aim is to determine only the properties linked to real-

valued sequences, since they are of fundamental importance for the developed models (as it is explained in Chapter 3).
In this regard, if $x(n)$ is real, then

$$|X(N-k)| = |X(k)|$$

$$\angle X(N-k) = -\angle X(k)$$

Furthermore, it can be demonstrated that $x_I(n) = 0$.
Finally, it is important to mention that the importance of DFT and IDFT operations is enhanced by the fact that there are computationally efficient procedures, such as the family of Fast Fourier Transform (FFT) algorithms, from which they can be performed. The computational complexity of these algorithms depends on whether or not the input signal has some specific attributes (such as the symmetry for real-valued sequences already described).

2.1.6 Discrete-time Systems

In DSP terminology, each physical device or digital algorithm that performs some operation on signals is called *system*. As previously motivated, when we work with data on a computer, time is discretized, so that each operation is performed on discrete-time signals. For this reason, we focus our attention on discrete-time systems, i.e. devices or algorithms that operate on a discrete-time signal.
Formally, we can define a system S as a set of operations performed on the input (or *excitation*) signal $x(n)$ to produce the output (or *response*) signal $y(n)$. Figure 2.7 illustrates this definition.
We can use equivalently the following two notations to express the transformation from $x(n)$ to $y(n)$ by the system S.

$$x(n) \overset{\mathrm{S}}{\rightarrow} y(n) y(n) = S(x(n)) \tag{2.22}$$

For our analysis purposes, as well as for providing a comprehensive overview of the basic DSP principles, it is important to classify systems according to their characteristics. In fact, the mathematical operations performed in this study are heavily dependent on the properties that the system satisfies.
According to [15], important classes of discrete-time systems are the following.

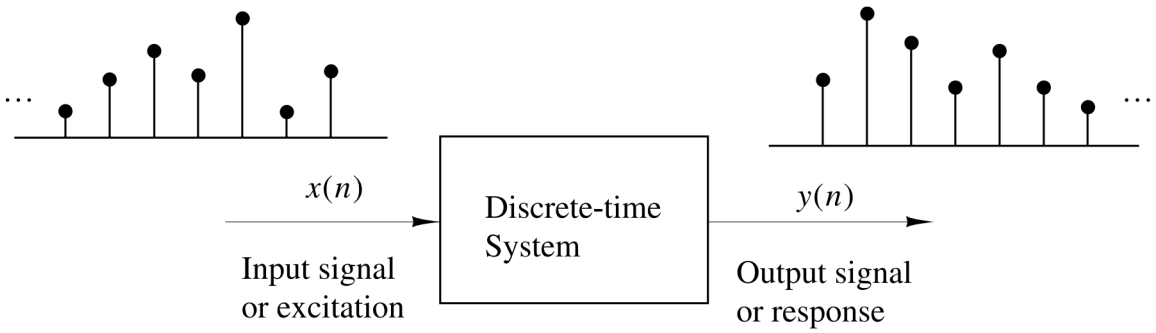

Figure 2.7: Block diagram representation of a discrete-time system. From [15].

- *Time-invariant systems.* A system is called time-invariant if its input–output description $x(n) \xrightarrow{S} y(n)$ do not changes with time. Formally, let $x(n)$ be the input of a system S; if

$$y(n,k) = y(n-k) \quad \forall k$$

where $y(n,k) = S[x(n-k)]$, then S is a time-invariant system.

- *Linear systems.* A system is said to be linear if it satisfies the principle of *superposition*, which is formally described below. Given two arbitrary input sequences $x_1(n), x_2(n)$ and two arbitrary constants a_1, a_2 a system S is linear if and only if:

$$S[a_1x_1(n) + a_2x_2(n)] = a_1S[x_1(n)] + a_2S[x_2(n)] \qquad \forall x_1(n), x_2(n), a_1, a_2 \qquad (2.23)$$

The above definition includes two important properties of linear systems: multiplicative and additive.

- *Causal system.* A system called causal if its output depends only on present and past inputs but does not depend on future ones. Formally, S is causal if:

$$y(n) = F[x(n), x(n-1), x(n-2), ...] \qquad (2.24)$$

where $F[.]$ is some arbitrary function.
It is quite intuitive that causal system are at the basis of real-time DSP applications, since the observation of input future values is physically not possible. On the other hand, off-line processing often involves algorithms that perform *non-causal* operations. A good example is provided by bi-directional RNNs, which, for a given time step, process not only the past data elements in a given sequence, but also the

future ones.

Of particular importance for many DSP applications is the class of Linear Time-Invariant (LTI) systems, whose response to an input sequence is determined by the *convolution* operation.

Convolution

Convolution is an important mathematical tool in both DSP and DL fields. In this section we introduce this operation from a theoretical perspective, while in paragraph 2.2.2 its application in the context of neural networks is described.
In order to introduce the convolution, it is first important to define the *unit sample sequence* (or *unit impulse*), which corresponds to the most basic discrete-time signal. In particular, it is a signal that is zero everywhere, except at $n = 0$ where its value is unity. Formally:

$$\delta(n) = \begin{cases} 1 & \text{se } n = 0 \\ 0 & n \neq 0 \end{cases} \tag{2.25}$$

Its graphical representation is provided in Fig. 2.8.

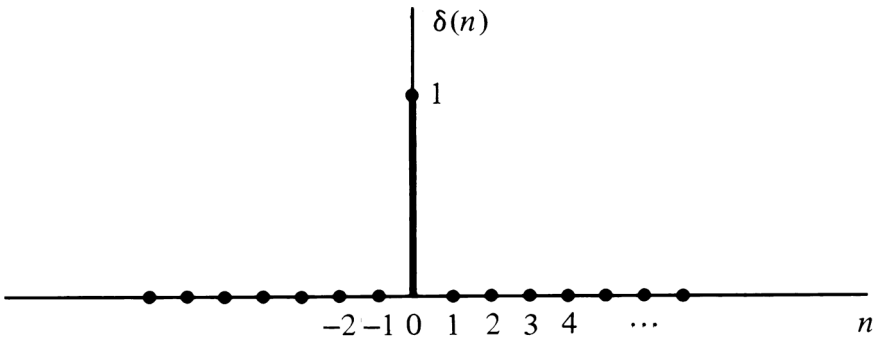

Figure 2.8: Graphical representation of the unit sample signal. From [15].

At this point, it is possible to introduce the equation 2.26, which expresses the decomposition of an arbitrary signal $x(n)$ into a weighted sum of shifted unit impulses.

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k) \tag{2.26}$$

The importance of the relationship above expressed lies in the following considerations.

If we consider the response of a LTI system to $x(n)$ is the corresponding sum of weighted outputs, we obtain the expression 2.27.

$$y(n) = S[x(n)] = S\left[\sum_{k=-\infty}^{\infty} x(k)\delta(n-k)\right] = \sum_{k=-\infty}^{\infty} x(k)S\left[\delta(n-k)\right] \tag{2.27}$$

Let now consider the unit sample signal $\delta(n)$ in a LTI system. We can denote as $h(n)$ the response of S to this signal. Formally:

$$h(n) = S[\delta(n)] \tag{2.28}$$

The formula in 2.28 is of particular importance as it characterizes the system S: as stated in [15], LTI systems can be subdivided into FIR (finite-duration impulse response) and IIR (infinite-duration impulse response) depending on whether $h(n)$ has, respectively, finite or infinite duration. These concepts lie at the basis of the design of digital filters, for which we refer to the cited book.
Because of the system time-invariance property, it can be seen that:

$$h(n-k) = S[\delta(n-k)] \tag{2.29}$$

Consequently, the formula in 2.27 reduces to:

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

which is called *convolution sum* and can be better expressed with the following notation:

$$x(n) \circledast h(n) = y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) \tag{2.30}$$

The previous result (formula 2.30) asserts that the response of an LTI system to a given input signal is obtained by *convolving* the input sequence $x(n)$ with the system's response $y(n)$ to the unit impulse $h(n)$.
Furthermore, it is important to mention that, in DL terminology (see [34]), the first argument (in our definition, the function x) to the convolution is called *input*, while the second argument (h) is called *kernel*. Furthermore, the output y is sometimes referred to

as the *feature map*.

The convolution operation has some important properties:

- *Commutative law*:

$$y(n) = x(n) \circledast h(n) = h(n) \circledast x(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = \sum_{k=-\infty}^{\infty} h(k)x(n-k) \tag{2.31}$$

- *Associative law*:

$$[x(n) \circledast h_1(n)] \circledast h_2(n) = x(n) \circledast [h_1(n) \circledast h_2(n)] \tag{2.32}$$

- *Distributive law*:

$$x(n) \circledast [h_1(n) + h_2(n)] = x(n) \circledast h_1(n) + x(n) \circledast h_2(n) \tag{2.33}$$

The convolution operation is an important tool even when we analyze the frequency response to LTI systems. In particular, we know from the so called *convolution theorem* that convolution in the time domain is equivalent to multiplication in the frequency domain. Formally, let $X(\omega), H(\omega), Y(\omega)$ be the spectra of, respectively, $x(n), h(n), y(n)$; then, the theorem asserts that:

$$y(n) = x(n) \circledast h(n) \overset{\mathscr{F}}{\longleftrightarrow} Y(\omega) = X(\omega)H(\omega) \tag{2.34}$$

From this theorem we can conclude that convolution is an easier operation in the time domain rather than in the frequency one.

2.2 Deep Learning

The term *artificial intelligence* is increasingly used in a variety of ways and behind it, there are many technologies. Rai et. al [50], provide a quite comprehensive definition. "AI is typically defined as the ability of a machine to perform cognitive functions that we associate with human minds, such as perceiving, reasoning, learning, interacting with the environment, problem solving, decision-making, and even demonstrating creativity". Although this definition is rather generic, it is possible to define, more specifically, the

disciplines that belong to it.

Most of the time, researchers refers to the term AI to indicate ML algorithms. We can consider ML as the discipline that extracts patterns of interest from data [34].

Mitchell [8] provides a quite formal definition of ML: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E".

ML is a very active research topic and it has many practical applications. Thus, the task T can range from medical diagnosis and community detection to antispam filtering and text categorization. However, many problems can be roughly categorized into two groups: classification (such as speaker identification) and regression (such as audio SR).

In order to evaluate the performance of a ML algorithm, a quantitative measure must be taken into account. Mitchell refers to this metric, which is always specific to the task T, as performance measure P. As for the learning experience E, ML algorithms can be broadly categorized in two main classes: *superivsed* and *unsuperivsed*. Briefly, supervised learning aims to generate reliable predictions using labeled data. In other words, a supervised learning algorithm experiences a dataset on which each example is associated with a so called "ground truth", which represents the real output the model should be able to reproduce (or, at least, approximate). Audio super-resolution is a good example of supervised learning, since the goal is to learn a function that accurately maps the input LR sequences to the corresponding HR audio frames.

On the contrary, in unsupervised learning, the ground truth is not available. Indeed, the learning process focuses on useful properties of the dataset structure. Most of the time, the goal is to learn the entire probability distribution that generated the data.

It is possible to define a third category, i.e. *semi-supervised learning*. According to Hady and Schwenker [20], this approach aims to maximize the performance of a ML model by integrating part or all of the available unlabeled data in its supervised learning process.

There are several different approaches in ML, depending on the task and the data. On one hand, many problems can be tackled by designing a specific set of features to extract. The feature extraction step is very important, since the performance of the ML algorithms are highly sensitive to the representation of the input data.

On the other hand, in some cases, it is not possible - or at least, is very difficult - to extract representative features for a specific task. For example, in the field of digital signal processing, descriptive variables such as energy, mean, variance, etc... are not always sufficient to detect distinctive patterns of interest. In this case, we can use specific

ML approaches that can discover the right features to solve the problem by using raw data. This *representation learning* approach often results in much better performance than can be obtained with hand-crafted feature extraction. Furthermore, representation learning is a less-time consuming approach since there is a minimal human intervention. As for *Deep Learning*, we can consider it as a ML sub-field, in which the features are extracted through increasingly abstract hidden layers. DL aims to estimate the desired mapping function by breaking it into a series of hidden mappings, each described by a different layer which provides a new representation of the input. DL models result in complex computational graphs whose interpretability is often poor.
A schematic illustration of the newly introduced concepts is shown in Fig.2.9.

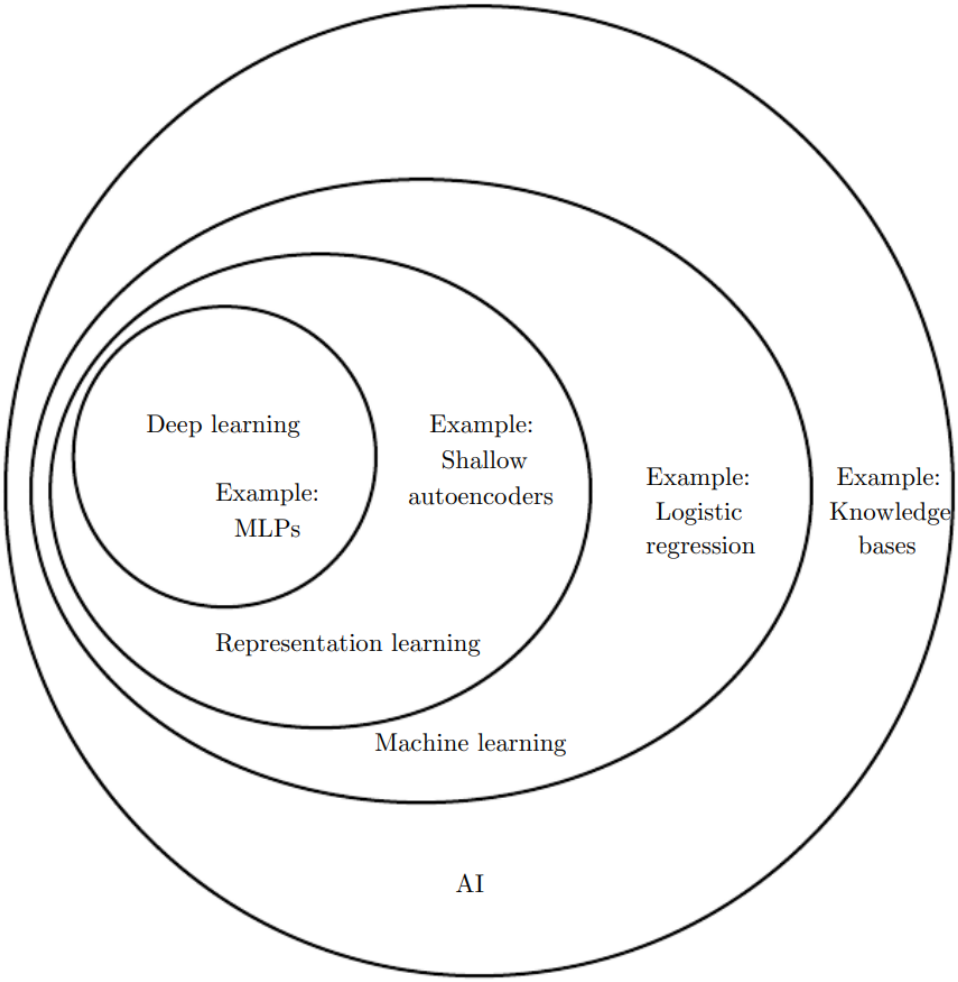

Figure 2.9: A Venn diagram showing the conceptual location of DL, ML and AI in the terms hierarchy. We can consider AI as the term with the wider range of uses. Each section of the diagram includes an example of an AI technology. From [34].

DL is particularly effective when applied to large training sets, i.e. in high dimensional spaces with a large number of data points. Indeed, its use is perfectly suited for DSP

tasks, in which quite a large amount of data is usually required. In this regard, it is noticeable that the popularity of deep-learning based approaches is largely due to their success in large-scale image classification tasks (see ImageNet [29]).
There are many variants of deep-learning algorithms with different architectures and characteristics; some examples includes convolutional, generative and recurrent networks. The main objective of this section is to provide a broad description of these methodologies, trying to cover in a rather exhaustive way the main aspects that characterize them. A particular focus is given to convolutional and recurrent approaches, as they are of fundamental importance for the purposes of this work.

2.2.1 The Learning Process

As previously said, the task of Audio SR belongs to the class of supervised learning problems; for this reason, we focus on the learning process, which is a central aspect that concerns this approach.
Briefly, supervised learning aims to learn a function f_θ that accurately maps input vectors X to target labels Y. This formulation can be applied to a wide variety of problems; for example, we can see an audio SR model as a mapping function of the form $f_\theta : X \to Y$ (see section 1.1).
In order to measure how the function f_θ is accurate, we need to introduce the concept of *objective function* also called *loss function* or *criterion*. We use these terms interchangeably though some researchers assign different meaning to some of these terms.
Formally, we can consider the criterion $L(\hat{y}, y)$ as a function which maps the model output $\hat{y}$ to a real number measuring the quality of the solution in terms of distance from the ground truth y. During the training phase, the parameters θ of the model are then set to optimize the loss.
Optimization in this case refers to the problem of minimizing (or, depending on the problem, maximizing) the objective function L over the training samples. Proper solutions of the optimization problem can concern gradient-based methods; without going into that in detail, we only mention that these iterative optimization algorithms seeks to find the minimum (or the maximum) of a function by obtaining partial derivatives. Derivatives are important as they indicate on which direction each parameter of the model must be adjusted in order to reduce the error.
The most common optimization algorithms are Stochastic Gradient Descent [1], Adam [24], Nadam [27], ... In all cases, the main idea is that, at each iteration, the loss function

must be decreased by moving it in the direction of the negative gradient. The model parameters are then adjusted through the backpropagation algorithm [5]. The convergence toward the minimum of the objective function model - and the speed of this convergence - depends upon the *learning rate*, a positive scalar value.

For more details about these interesting aspects, as well as for a formal statement, we refer to the book "Deep Learning" [34] by Goodfellow, Bengio and Courville.

As for loss functions, we mention, as typical examples, Negative Log-Likelihood, Cross-Entropy and Mean-Square Error (MSE); the choice of which one to use is strongly influenced by the nature of problem.

A key aspects of optimization in DL problems, is that loss functions are not necessarily convex. Ideally, we would obtain the global minimum, but this is not always possible, since there are multiple local minima and plateaus. For this reason, suboptimal solutions are generally accepted, as long as they correspond to significantly low values of the cost function.

The main challenge of DL (or, more generally, ML) problems is that we must produce good predictions on unseen data. Indeed, according to [34], we can determine how well an algorithm performs by looking for its ability to both obtain a sufficiently low error value on the training set and minimize the gap between training and test error. Both abilities are of fundamental importance to avoid *underfitting* and *overfitting* problems. In particular, when the model is not able to make the training error small, we say it underfits. On the other hand, overfitting occurs when we measure a large gap between training and test error. The desired behavior occurs when the algorithm succeeds in learning the data distribution, providing good performance on the training set and, at the same time, can *generalize* well on previously unobserved instances.

Underfitting and overfitting can be defined as function of two different sources of error: *bias* and *variance*. The former is a measure of the expected deviation of the estimator relative to the true value. From a statistical point of view, although unbiased estimators are desirable, since they have important properties, they are not always the best estimators. At the same time, a high bias can cause the underfitting.

On the other hand, variance measures the variability of the estimator about the expected value. If the model gets too sensitive to small variations of the data points during the learning phase, the generalisation capability can then be very poor (overfitting).

Therefore, when training a DL model, it is important to take into account these aspects, as we aim to simultaneously minimize these two sources of error. This conflict is known

in literature as "bias-variance trade-off". Figure 2.10 provides a graphical representation of this problem.

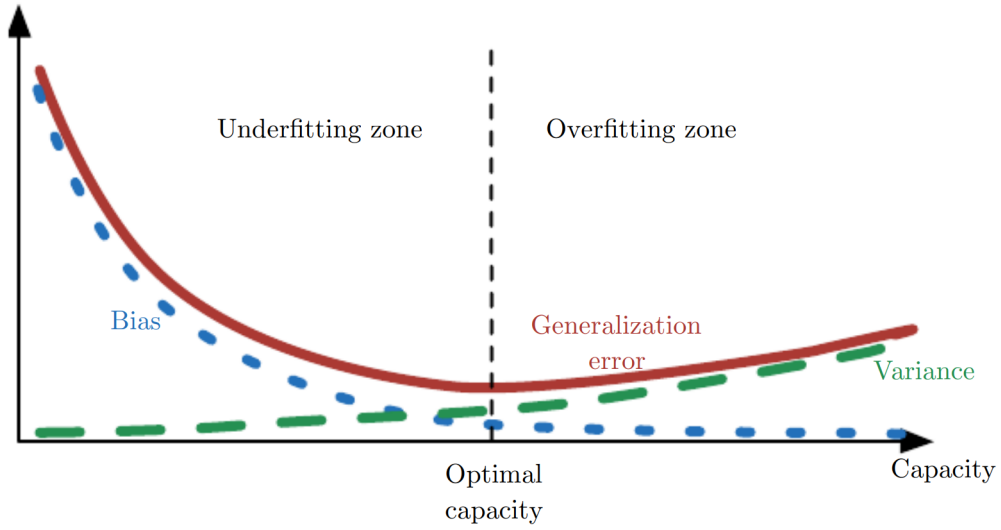

Figure 2.10: Bias and Variance trade-off. As the capacity (complexity) of the model increases (x-axis), bias (dotted) tends to decrease and variance (dashed) tends to increase. The optimal capacity occurs when both bias and variance errors are minimized. From [34].

When overfitting occurs, it is possible to adopt regularization strategies with the specific aim to reduce the variance. Such techniques may vary according to the algorithms and the structure of the problem; to summarize, we only take into account regularization strategies for neural networks.

According to [34], we can refer to regularization as “any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.” Typical examples of regularization techniques concern L1 and L2, which can be theoretically applied to any algorithm as they add parameter norm-based penalty to the objective function.

Another common strategy, especially in DSP applications, is the so called "data augmentation". This is particularly suited when the dataset is small, as it allows the generation of additional and more diversified data points through certain transformations conducted upon original instances. Data augmentation is the easiest and cheapest way to increase the amount of training samples; however, the acquisition of new data, when possible, is preferable. The correct transformation to apply highly dependent on the particular application. In Figure 2.11 it is possible to see different ways of doing augmentation on images.

Figure 2.11: Examples of color augmentations tested by Wu et al [30].

Particularly worthy of mention are *dropout* [25] and *early stopping*, as they are two of the most used regularization strategies in DL (and are both used in this project).
The former is a popular method to improve generalization by stochastically "dropping out" units during training to prevent their co-adaptation.
As for early stopping, briefly, it is based on choosing when to stop training a DL model. The interruption criterion can vary wildly depending on the application; however, it is fair to say that it mostly depends on the loss on a validation set. An intuitive way of thinking about early stopping in DL setting is to consider it as a hyperparameter selection method, where training time is the hyperparameter to be optimized.

2.2.2 Convolutional Neural Networks

As previously motivated, CNNs are an essential tool when facing DSP problems and, for this reason, they are introduced in this section. CNNs have revolutionized both the DSP and the DL fields, to the point that several DL-based applications performs even better than human.
CNN and, more generally DL, came to mainstream popularity when Krizhevsky et al. [19] won the 2012 ImageNet Large-Scale Visual Recognition Challenge (ILSVRC [29]) with a convolutional architecture called AlexNet.

The name Convolutional Neural Networks indicates that the system employs, in at least one of its layers, the convolution operation, which is previously introduced. Unlike in audio processing applications, where the convolution is generally applied only to one-dimensional signals, in DL systems, usually this operation takes place between tensors, i.e. multidimensional vectors. Furthermore, in many DL architectures the operation used in a CNN does not correspond precisely to the traditional convolution: indeed, there are several variants such as the one defined in paragraph 3.2.1.

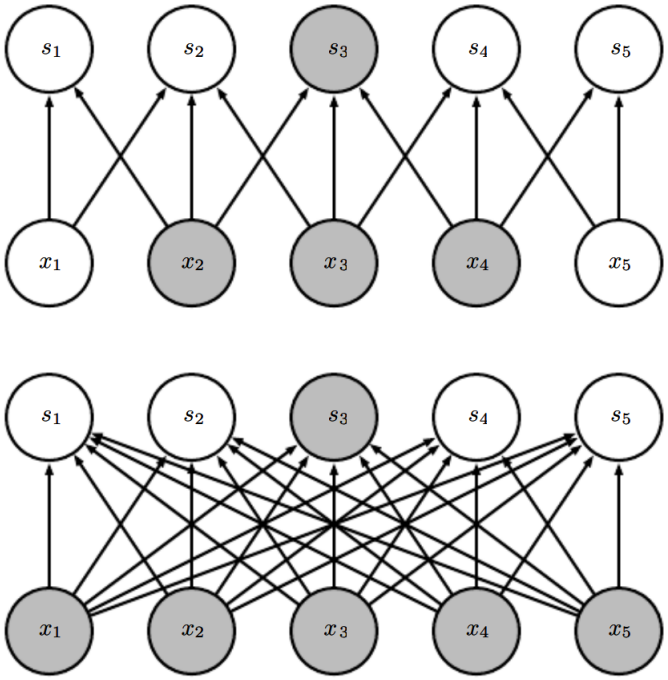

Figure 2.12: Graphical representation of the connections between input and output units in both convolutional and fully connected approaches. *Top:* When we use CNN layers with a kernel of width 3, the connectivity is sparse such that each output unit is affected only by 3 input units. *Bottom:* In fully connected layers each output unit is formed by matrix multiplication, so that all of the inputs affect the highlighted s_3. From [34].

One of the main advantages of CNNs over traditional fully connected layers is that they allows to reduce the memory requirements by limiting the number of connections between input and output units. Thus, while in fully-connected layers forward propagation as well as backpropagation are expressed as matrix multiplications, CNN layers use convolution which connects input and output units sparsely. This *sparse connectivity* (also referred to as *sparse interactions*) in convolutional networks reduces the computational complexity since the output computation requires fewer operations. In particular, the higher the *receptive filed* of a convolutional layer is, the higher is the number of parameters, i.e. the

memory requirements. A graphical illustration of the sparse interactions between input and output units is given in Figure 2.12.

Even if direct connections in a convolutional net are very sparse, units in the higher layers can indirectly interact with a larger portion of the input. In other words, deeper layers receive generally more global information since the receptive field of the units in the higher layers is larger than the one of the units in the shallow layers. Furthermore, this effect is more pronounced if the system includes operations like *pooling* (see the book [34]). A graphical example of this intuitive concept is given in Figure 2.13.

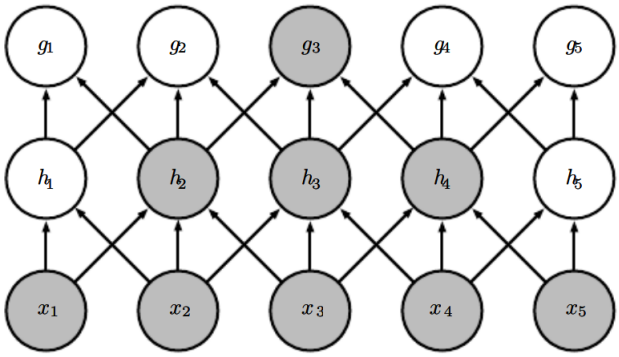

Figure 2.13: Deeper layers can indirectly interact with a larger portion of the input. From [34].

As previously motivated, deep convolutional networks are particularly suitable to capture complicated non-linear interactions between data points. The learning process leads to the creation of deep latent variables, which are quite difficult for a human to interpret. Because of this "black-box" nature, DL models suffer from providing meaningful interpretations of their results. For this reason, the interpretation of deep networks is a very active research area, and many works in literature aims to discover what a CNN really learns from data.

An interesting paper is that of Zeiler and Fergus [26] who works on images. They demonstrate that different layers of a CNN respond distinctively to specific aspects of images. In particular, they discover that early hidden layers detect low-level features such as edges, corners and shapes, while deeper layers extract more abstract (high-level) information, such as categories or objects. Figure 2.14 provides a graphical example of this information hierarchy during the CNN learning process.

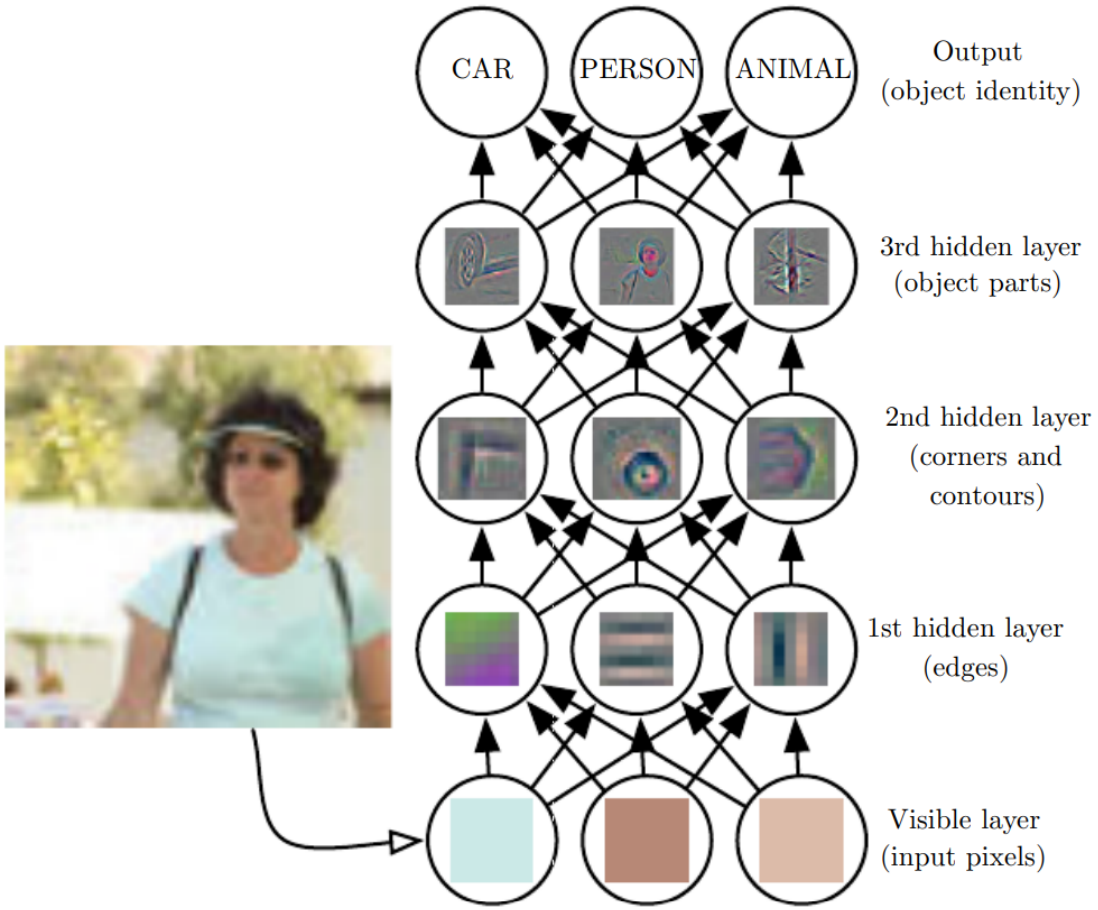

Figure 2.14: A convolutional neural network extracts increasingly abstract features from an image. From [34].

2.2.3 Recurrent Neural Networks

In a traditional neural network we assume that input and output data do not follow a specific order; in other words, the implicit assumption is that data points are independent of each other. Although this assumption is valid in many problem settings, it is extremely limiting in others. For example, if we want to analyze historical data representing the global earth temperature, we have to take into account the implicit time axis between each observation. In speech processing too, data points are dependent of each other and this leads to the need for an appropriate sequential modelling.

Recurrent neural network approaches arise from this need: according to [34], much as a CNN is particularly suited for processing data that have a grid pattern, such as images, a RNN is specialized for processing time series, or, more generally, sequential data. An intuitive way to think about RNNs is that they have a sort of "memory" which stores relevant information about what they have processed so far.

More specifically, RNNs define recurrence relations between data points by introducing

cycles in their computational graph to efficiently model this sequential influence.
The basic idea behind recurrent approaches is that, at each time step, an interaction takes place through hidden recurrent connections. This interaction, which involves a cycle in the graph computation, is graphically represented by a black square in Figure 2.15.

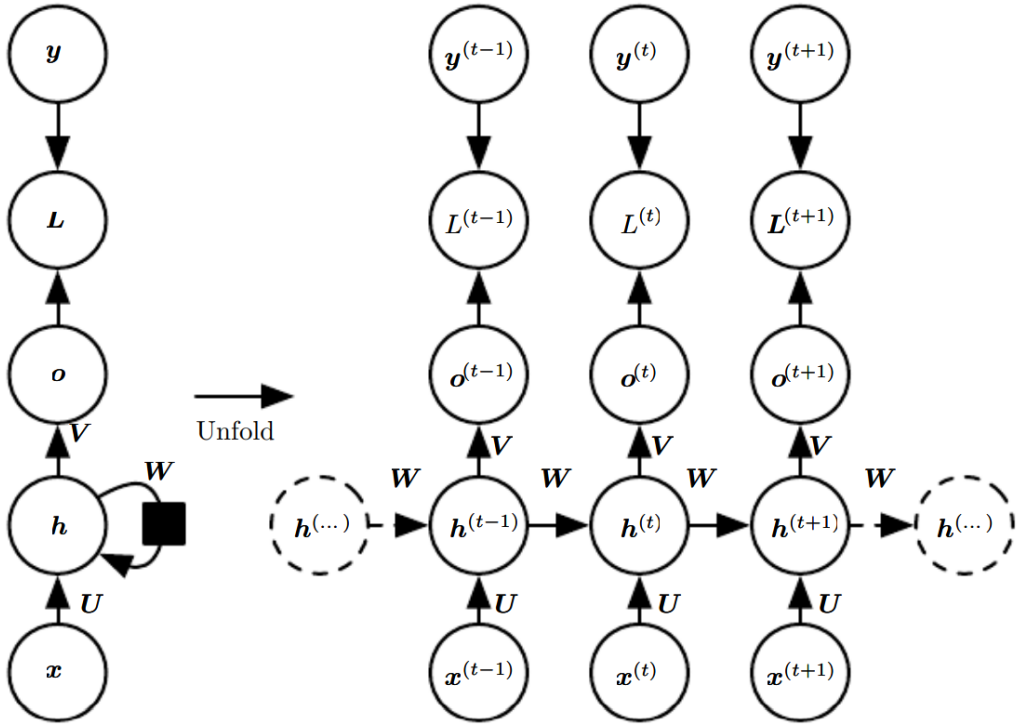

Figure 2.15: "The computational graph to compute the training loss of a recurrent network that maps an input sequence of **x** values to a corresponding sequence of output **o** values. A loss **L** measures how far each **o** is from the corresponding training target **y**. When using softmax outputs, we assume **o** is the unnormalized log probabilities. The loss **L** internally computes $\hat{y} = \text{softmax}(\mathbf{o})$ and compares this to the target **y**. The RNN has input to hidden connections parametrized by a weight matrix **U**, hidden-to-hidden recurrent connections parametrized by a weight matrix **W**, and hidden-to-output connections parametrized by a weight matrix **V**". *Left:* Circuit diagram representation. *Right:* The same RNN seen as an unfolded computational graph. From [34].

Figure 2.15 illustrates a simple RNN that processes the information from the input **x** by incorporating it into the state **h** that is passed forward through time.
It is noticeable that this architecture aims to produce an output at each time step and has recurrent connections between hidden units. Although this is a common schema, it may not be the rule. Indeed, depending on the task, it might be reasonable to adopt other architectural configurations. For example, when working on a speaker identification

task, we aim to obtain the final output after the whole input recording is processed, not after each audio sample. In this case we need a RNN that reads an entire sequence and produces a single output, such as the one illustrated in Figure 2.16.

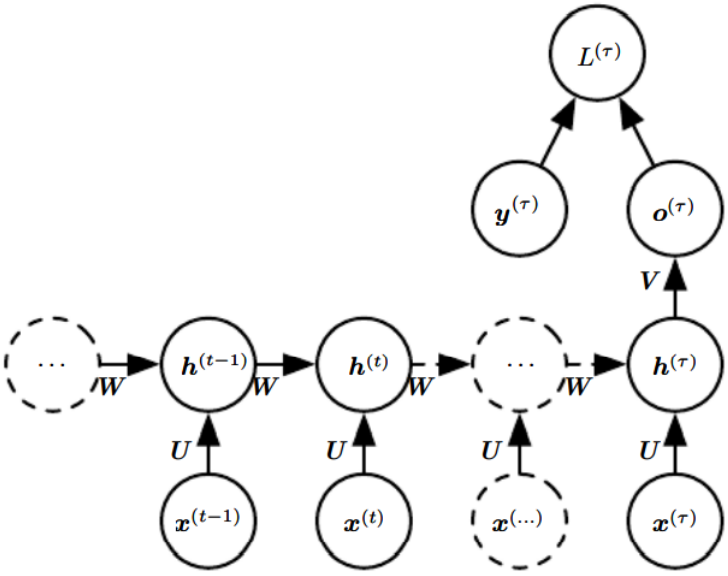

Figure 2.16: Time-unfolded RNN with a single output at the end of the sequence. From [34].

It is important to mention that recurrent networks are trained through the Backpropagation Through Time algorithm ([3], [4]), which computes the gradient of model error with respect to its weights.

Common problems in recurrent DL approaches are the ones of *vanishing* and *exploding* gradients. Especially traditional RNNs suffer from it: if the gradient tends to be very close to zero in the training process, the network weight correction vanishes and, consequently, the backpropagation through time fails.

One possible solution to reduce the effect of vanishing gradients would be to adopt *gated* RNNs architectures. These special sequence models include Long Short-Term Memory (LSTM)[7] and Gated Recurrent Unit (GRU)[22]. Without going into too much detail, gated RNNs are based on the idea of creating few trainable gates (paths) to select the most important information to memorize through time. These pieces of informations are then propagated by connection weights that may change at every time step. For a more formal and comprehensive description of these recurrent approaches we encourage to consider deep learning textbooks, such as [34].

Chapter 3

Methods for Artificial Bandwidth Extension

As said in Chapter 1, the proposed strategy to solve the audio SR task is based on an hybrid architecture that combines both TFNet [42] and TFiLM [46] methods. In order to describe our system, a previous analysis of the two models from which the proposed one derives needs to be done. For this reason, both TFNet and TFiLM approaches are described in paragraphs 3.1 and 3.2, respectively.

All methods for audio SR that are presented in this chapter derive from a mathematical formulation that treats this task as a pure regression problem. As stated in paragraph 1.1, the goal of the model is to characterize the conditional distribution $p(y|x)$ of a HR sequence given its LR input. Therefore, the point-estimate output results in a sequence $\hat{y} = f_\theta(x)$ of real values. This formulation naturally leads to the least-squares optimization problem of the form:

$$\min_\theta \sum_i \|f_\theta(x_i) - y_i\|_2^2 \tag{3.1}$$

where $\|.\|$ denotes the norm function, x_i, y_i are training examples such that $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$ and f_θ is the DL model parametrized by θ. As a natural consequence of this setting, the objective function of this work can be defined as:

$$\ell(\mathcal{D}) = \frac{1}{n}\sqrt{\sum_{i=1}^n \|y_i - f_\theta(x_i)\|_2^2} \tag{3.2}$$

In other words, artificial BWE can be seen as a structured regression task, where the goal is to minimize the difference between the model point estimation and the target sequence. The problem can be tackled by deep neural network architectures, that have enough

capacity to perform non-linear regression. Indeed, many levels of non-linearities allow them to represent a complex non-linear regression function that maps LR input data to HR audio frames. Therefore, all DL systems presented in this chapter are trained in order to minimize the metric given in Equation 3.2.

Another common factor shared by all of these audio SR methods is that the source input is pre-processed with a a cubic B-spline upscaling in order to ensure LR/HR signals are of the same length. This operation has multiple advantages. On one hand, it allows to have a baseline to compare the results of DL models. On the other hand, since this operation is included in the processing pipeline, it is possible to exploit it by the use of residual connections to link the input and the output series. This solution allows to speed up the required training time, because the model f_θ is facilitated by having to estimate only the difference between the LR source and the target HR signal. This and all other pre-processing operations are discussed in detail in Chapter 4.

It is important to point out that, due to computational limitations, it is not possible, in this work, to propose and train a new model that has the same number of parameters than TFNet and TFiLM. Therefore, results reported in the two model papers can not be used to make comparisons between different architectures. In fact, model comparisons must be performed in fair contexts, i.e., by training models with approximately the same number of parameters. For this reason, TFNet and TFiLM models are reduced in size and, subsequently, re-trained so that they can be compared with the proposed architecture under the same conditions.

This chapter aims not only to provide a clear idea of the original configurations of the models, but also to describe the changes made to reduce dimensionality. Therefore, in the paragraphs 3.1 and 3.2, the models TFNet e TFiLM, respectively, are described in their original configurations. Then, in the paragraph 3.4, all details related to the implementation of the models, and the changes made to the original configurations are thoroughly described.

3.1 TFNet

Time-Frequency Network, proposed by Lim, Yeh *et al.*[42], is the first model in literature that addresses the problem of audio SR by operating in both time and frequency domain. In the paper, authors highlight the origin of their intuition. “At the first glance, modeling in both frequency and time domain seems like a redundant representation; From

Parseval's theorem the ℓ_2 difference of prediction error, whether in the frequency or time domain is exactly the same. However, regression from LR to HR in time or frequency domain solves a very different problem. In the time domain, it is analogous to the image super-resolution task, mapping *audio patches* from LR to HR. On the other hand, SR in the frequency domain is analogous to the *semantic image inpainting* task (BWE in spectral domain can be viewed as image inpainting of spectrograms) [35],[40]".

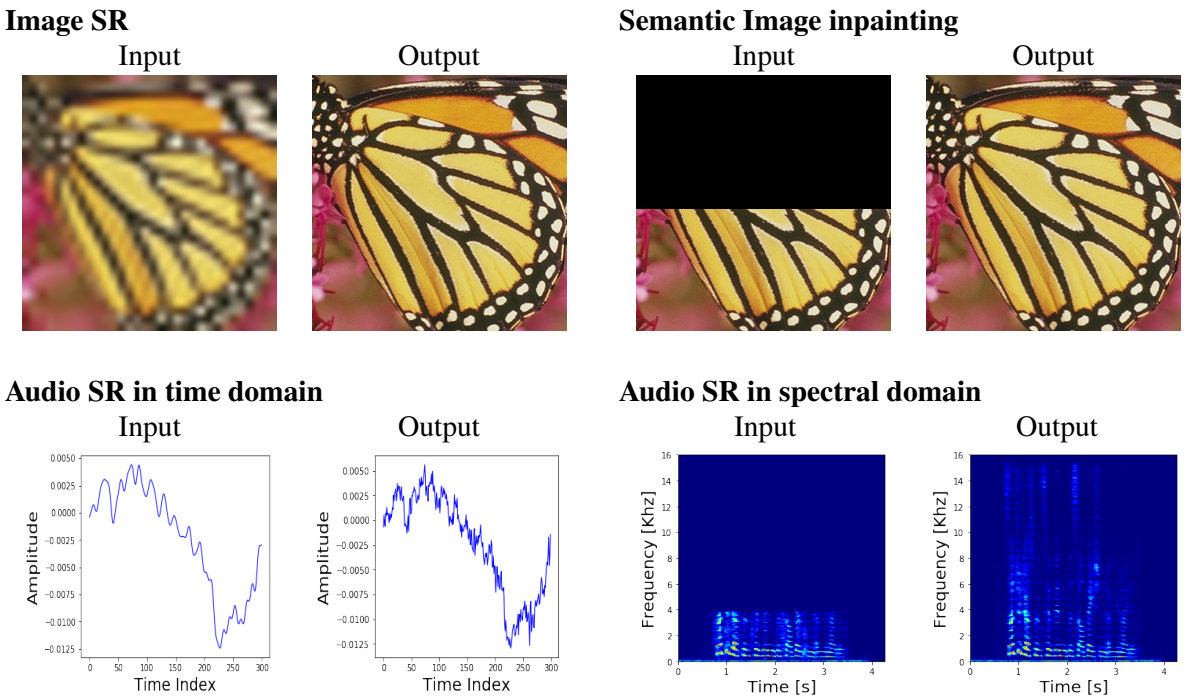

Figure 3.1: *Top row*: Examples of image SR and semantic image inpainting reconstruction. *Bottom row*: Illustration of the input/output for audio SR in time and frequency domain. From [42].

Figure 3.1 sums up the close relationship that links audio SR in time domain with image SR, and audio SR in frequency domain with image inpainting.

Based on these considerations, authors propose a system made of two main branches: one which models signals in the spectral domain, while the other branch explicitly models the reconstruction in time domain.

More specifically, TFNet is a fully differentiable network that estimates, for a given LR input x, the HR audio reconstruction $\hat{z}$, and the HR spectral magnitude $\hat{m}$. The last layer of the model, called *Spectral Fusion Layer*, computes Fourier transform operations in order to combine $\hat{z}$ and $\hat{m}$ in a unique output $\hat{y}$.

A brief overview of the overall pipeline of the model architecture is provided in Fig. 3.2.

As we can see, the time-domain branch processes the signal through the "Slim AudioUNet" that is a fully convolutional component, explained in detail in 3.1.1. The frequency branch performs more operations during the system processing: first, a DFT on the sequence is computed to pass from one domain to another. It is well known that when the DFT is performed on purely real input, such as audio signals, negative phase components are redundant and can be discarded. This is because the DFT of a real signal is Hermitian-symmetric, i.e. negative-frequency terms can be obtained from the corresponding positive terms. Therefore, given the LR input signal x of length T_1, only $\frac{T_1}{2} + 1$ components are taken into account: the zero-frequency term (DC component) followed by the $\frac{T_1}{2}$ positive-frequency terms. These $\frac{T_1}{2}$ values are then processed through the "Spectral Replicator" and "Slim AudioUNet", while the zero-frequency term pass directly through the network. A final concatenation operation is applied in order to combine the DC component with the processed positive-frequency terms.

All the details concerning architectural components not yet presented, such as "Slim AudioUNet", "Spectral Replicator" and "Spectral Fusion Layer", are thoroughly studied in the next paragraphs.

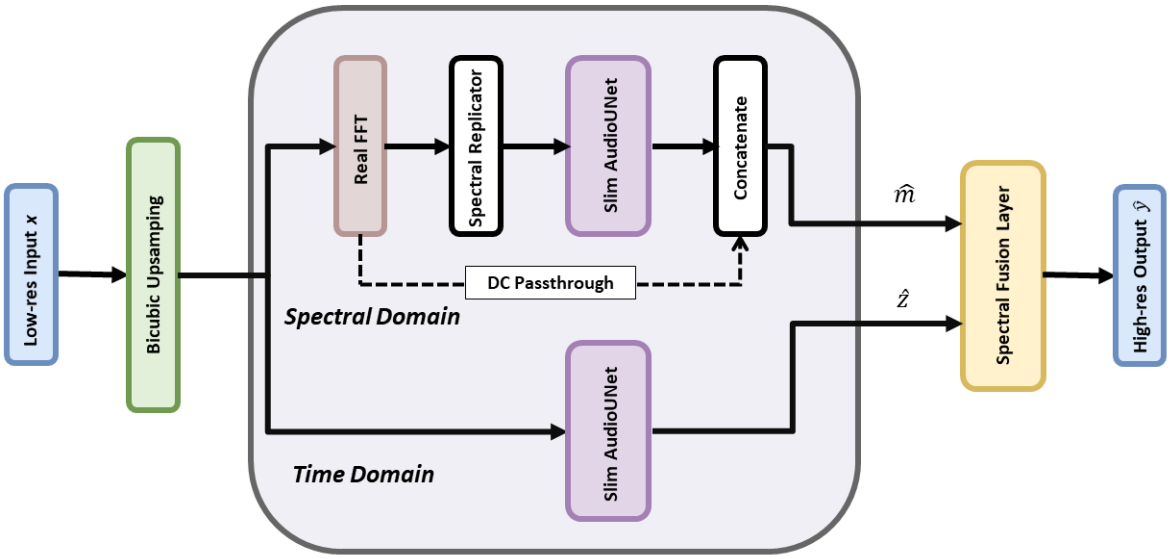

Figure 3.2: Overall pipeline of TFNet. The system exploits both time and frequency domain information in order to map the LR input x to the HR reconstruction $\hat{y}$. From [42].

3.1.1 AudioUNet

AudioUNet is a fully feedforward network that consists of downsampling and upsampling blocks. The model was originally designed by Kuleshov *et al.* [39], who demonstrates the effectiveness of convolutional architectures on the BWE task.

As Figure 3.3 shows, the model has a bottleneck architecture, that is designed in order to encourage the model to learn a hierarchy of features, such as auto-encoders. In this regard, it is possible to reasonably imagine that, on an audio task, bottom layers may learn wavelet-like patterns, while top ones may correspond to more complex audio units, such as phonemes [32].

AudioUNet is a fully convolutional neural network with residual connections. It contains B successive encoder/decoder blocks that produce dimensionally mirrored outputs: downsampling components halve the temporal (or spatial) dimension and double the number of filters, while upsampling components do the opposite.

Moreover, each block performs a specific series of operations. In particular, encoder blocks perform only Convolution and Leaky Rectified Linear Unit (ReLU), while the bottleneck layer and decoder components perform Dropout too.

It is important to mention that, during the upsampling stage, two more operations for the improvement are conducted: one-dimensional Subpixel shuffling and residual connections concatenation. The former consists in a principled reshape of tensors. It was originally designed to work on two-dimensional signals: *Shi et al.* demonstrate that this operation produce less artifacts on images reconstructed by a SR algorithm [36]. Authors of AudioUNet reasonably assume that this property can be extended to audio signals as well. As for residual connections, their use is motivated by the fact that, in a bottleneck architecture, when the input is similar to the target, downsampling features can be also useful for upsampling [37]. These two operations are linked by the following pipeline: the subpixel layer reshuffles a tensor $F \in \mathbf{R}^{T \times C}$ (where T is the temporal dimension, while C is the number of channels), into another one of size $F \in \mathbf{R}^{\frac{T}{2} \times 2C}$; these are concatenated, through a skip connection, with $\frac{T}{2}$ features from the downsampling stage, for a final output of size $F \in \mathbf{R}^{T \times 2C}$. Finally, a further skip connection is used to link the input data and the final reconstruction. Figure 3.2 clearly illustrates that AudioUNet is used to model both the audio reconstruction and the spectral magnitude. It is important to indicate that B (the number of upsampling and downsampling blocks) is equal to 4 in both time and spectral branches.

As for the terminology used by Lim, Yeh *et al.*, the term "Slim AudioUNet" in Fig. 3.2

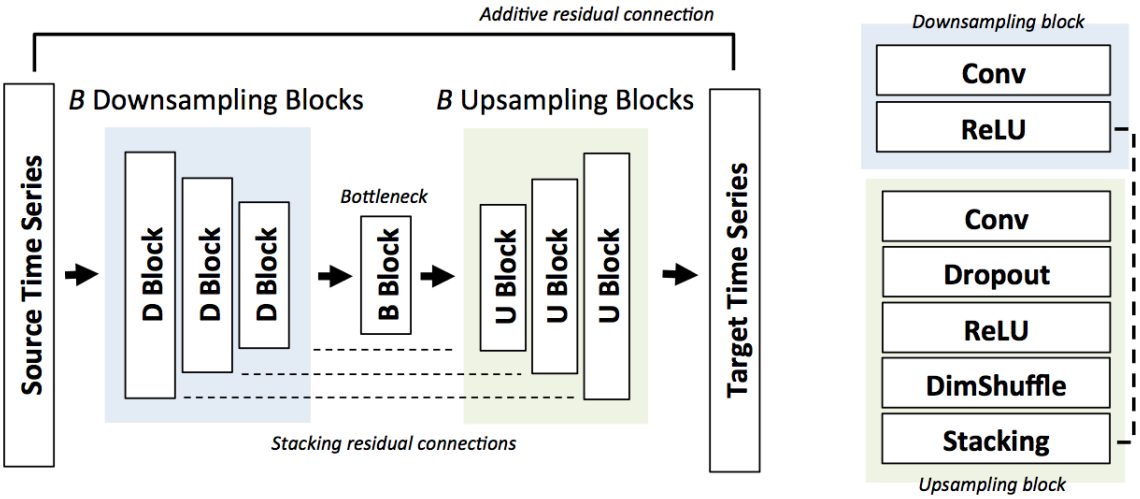

Figure 3.3: AudioUNet architecture. It consists of B successive downsampling/upsampling blocks linked by residual connections. From [39].

refers to the fact that they reduce the original dimension of the network by halving the number of filters in each of the branch. Otherwise, the model is equal to the one just described.

3.1.2 Spectral Replicator

As mentioned previously, the problem of audio SR is based on the hypothesis that it is possible to reconstruct the high-frequency content of a signal from its corresponding low-frequency counterpart. The Spectral Replicator is a layer designed in order to alter the content of an input spectrum such that this low-high frequency dependency is explicitly expressed. The idea behind the design of this component is described in detail below.

Given that the objective of the system is to increase the sampling rate of a signal, it must be taken into account that, due to the Nyquist limit, there are some frequency components in the HR target that can not exist in the LR input sequence. It is possible to explain this concept formally in the following way.

Recalling the notation introduced in paragraph 1.1, suppose we are trying to increase the temporal resolution of a signal x sampled at R_1 by a factor of r. From the DSP theory, we know that the Nyquist rate for that signal is equal to $R_{1_{lim}} = \frac{R_1}{2}$. However, the theoretical limit frequency of the HR sequence is equal to $R_{2_{lim}} = r \times R_1$. From this point of view, audio SR problem can be seen as the task of estimating the missing frequency components between $R_{1_{lim}}$ and $R_{2_{lim}}$. In other words, the aim of the system is to reconstruct the values over the interval $[R_{1_{lim}}, R_{2_{lim}}]$, on which the spectrum of the LR

sequence is null.

As mentioned extensively in the previous section, the reconstruction is made by a convolutional approach. Here arises a problem, since convolution is a local operation, i.e. only short-range dependencies in sequential inputs can be captured and processed due to the receptive field limitation. In other words, convolution cannot fully investigate non-local joint dependencies that could be useful to estimate the HR spectrum. Therefore, it can be too challenging for a fully convolutional model to make the output's high frequency component depend on the input's low frequency counterpart. This limitation is accentuated as the scaling ratio r increases, because the interval $[R_{1_{lim}}, R_{2_{lim}}]$ gets larger.

To overcome this issue, the Spectral Replicator layer explicitly replicates $r-1$ times the low-frequency part of the signal, compensating the limitations of convolutional approaches. Therefore, all the zeros in the high frequency components of the spectrum are replaced by copies of the low frequency respective counterpart.

Specifically, each frequency term $k_i \in [k_1, k_{\frac{T_1}{2}}]$ (where T_1 is the length of x) is copied r times through this layer. The only exception is given by the term k_0 (DC component), that, as said before, is not processed through the Spectral Replicator.

For example, suppose to have a signal x whose spectral content is limited by the frequency term k_{1024}. For $4\times$ upsampling, we replace the zeros over the interval $[k_{1025}, k_{4096}]$ by repeating the 1st component to the 1024th one at 1025 to 2048, 2049 to 3072 and finally 3073 to 4096. Finally, these 4096 values are concatenated with the k_0 term.

Figure 3.4 illustrates the Spectral Replicator layer working on the BWE task with $r = 4$. The input signal x in the example has a sampling frequency of $R_1 = 8$kHz, so that the Nyquist limit is $R_{1_{lim}} = 4$kHz. As we can see, the low frequency patterns are replicated multiple times such that the zeros are replaced.

3.1.3 Spectral Fusion Layer

The spectral Fusion Layer is a key component of the TFNet architecture. It permits to combine outputs of the two branches, i.e. $\hat{m}$ and $\hat{z}$, in a single HR final reconstruction. To recap, the model predicts, for a given LR input x, the estimation of the HR audio reconstruction $\hat{z}$, and the HR spectral magnitude $\hat{m}$. These two quantities are finally synthesized in the temporal domain through this layer.

Specifically, the process through which this happens consists of two main stages. First of all, since the two branches operate in two different domains, there is the need to reduce

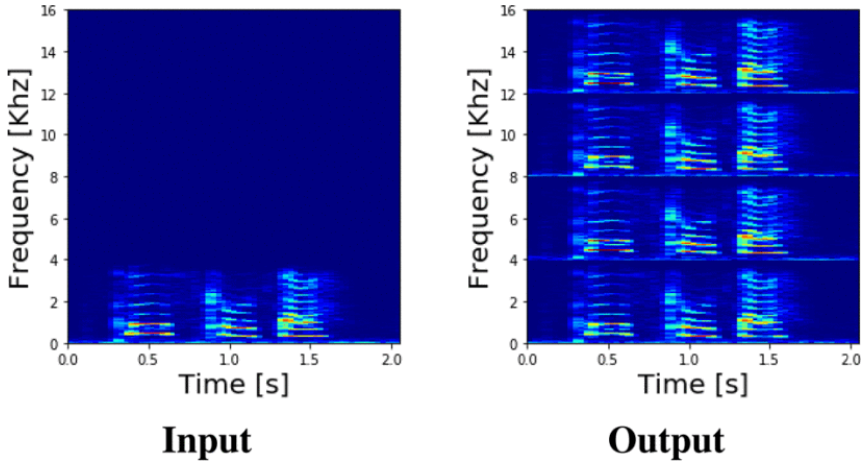

Figure 3.4: Spectrogram showing how the signal is processed through the Spectral Replicator layer. The low-frequency components of the input spectrum are replicated three times in order to replace zeros in the high-frequency counterpart. From [42].

the data to the same domain. This operation is performed according to the following equation, which defines the HR spectral magnitude estimate.

$$M = w \odot |\mathscr{F}(\hat{z})| + (1 - w) \odot \hat{m} \tag{3.3}$$

Equation 3.3 shows that the final estimation of the HR spectral magnitude is a weighted average of the individual estimates provided by each branch. A data-driven approach is used to establish the weight of each branch in this operation. Indeed, this equation involves an element-wise multiplication with the trainable parameter w.
The second stage consists in calculating the final output $\hat{y}$ through Equation 3.4. This one, is used to bring back the data to the original domain, i.e. the temporal one. Moreover, this second equation shows that the HR phase is estimated only in the time domain.

$$\hat{y} = \mathscr{F}^{-1}\left(Me^{j\angle\mathscr{F}(\hat{z})}\right) \tag{3.4}$$

The Spectral Fusion Layer is fully differentiable and can be trained end-to-end.

3.2 TFiLM

The main objective of this paragraph is to present Temporal Feature-Wise Linear Modulation, a neural network component proposed by Birnbaum, Kuleshov *et al.* in 2019 [46]. The key contribution of their work is to show that it is possible to capture long-term information in sequential inputs by combining elements of convolutional and recurrent

approaches.

The main intuition behind the TFiLM approach is as follows. CNNs, which are extensively used for audio SR, can effectively process digital signals, such as audio, images or video and are relatively easy to train. However, these models have some disadvantages that could limit their prediction performance. In particular, convolutional approaches are not well suited for long sequential data processing because of the limited receptive field, which results in insufficient capacity to capture long-range input dependencies. Indeed, the larger the receptive field, the more a convolutional model is computationally complex. To overcome this issue, the researchers propose a particular architectural component that can capture both short and long-term interactions between features.

In summary, the TFiLM algorithm can be viewed as a temporal adaptive normalization layer that modulates the activations of a convolutional layer through a RNN. More specifically, this algorithm takes as input a tensor of 1D multichannel convolutional activations $F \in \mathbf{R}^{T \times C}$ where T, C are, respectively, the temporal dimension and the number of channels, and a positive integer value $B \in \mathbf{N}^{+}$ that identifies the block length. The output is an adaptively normalized tensor of activations $F' \in \mathbf{R}^{T \times C}$. The whole algorithm pipeline can be divided into five steps:

1. Reshape F, along the spatial dimension, into a block tensor $F^{\text{blk}} \in \mathbf{R}^{B \times T/B \times C}$, defined as $F^{\text{blk}}_{b,t,c} = F_{b \times B + t, c}$. Each block can be considered as a region along the time axis in which the audio samples are closely correlated; for example, when processing audio, blocks could reasonably correspond to a set of activations that define a possible phoneme.

2. Obtain a representation $F^{\text{pool}} \in \mathbf{R}^{B \times C}$ of the block tensor by pooling together the channels within each block: $F^{\text{pool}}_{b,c} = \text{Pool}\left(F^{\text{blk}}_{b,:,c}\right)$.

3. Compute, for $b = 1, 2, \ldots, B$, an affine transformation $(\gamma_b, \beta_b), h_b = \text{RNN}\left(F^{\text{pool}}_{b,\cdot}; h_{b-1}\right)$ using an RNN applied to pooled blocks. The first value is $h_0 = \overrightarrow{0}$, where h_b denotes the hidden state and (γ_b, β_b) is a tuple of trainable parameters.

4. Activations in each block b are normalized by γ_b, β_b. Formally, normalized block tensor $F^{\text{norm}} \in \mathbf{R}^{B \times T/B \times C}$ are computed as $F^{\text{norm}}_{b,t,c} = \gamma_{b,c} \cdot F^{\text{block}}_{b,t,c} + \beta_{b,c}$.

5. Reshape F^{norm} into output $F' \in \mathbf{R}^{T \times C}$ as $F'_{\ell,c} = F^{\text{norm}}_{\lfloor t/B \rfloor, t \bmod B, c}$.

Notice that the way through which activations are modulated using long-range input dependencies is well described in the third stage. Indeed, we can see that each γ_b, β_b is a

function of both the current and all the past blocks. By doing so, all long-term informations are captured by the RNN and processed for each hidden state. To better understand the main idea behind Feature-Wise Linear Modulation (FiLM), see paragraph 3.2.2.
The main features of this architectural component are illustrated in Fig. 3.5. It is important to underline that, although the image illustrates a Bidirectional LSTM (BiLSTM) network, actually a unidirectional LSTM is used. Researchers in the paper perform several experiments with the specific aim to investigate the impact of using bidirectional RNNs rather than standard unidirectional recurrent layers. Their results show that, generally, switching from a unidirectional LSTM to a BiLSTM does not provide a significant improvement in performance and, in some cases, it even increase the prediction error (perhaps due to overfitting). Moreover, there is a remarkable saving in computational power cost by using unidirectional recurrent approaches.
This layer efficiently incorporates long-term information when processing sequential data.

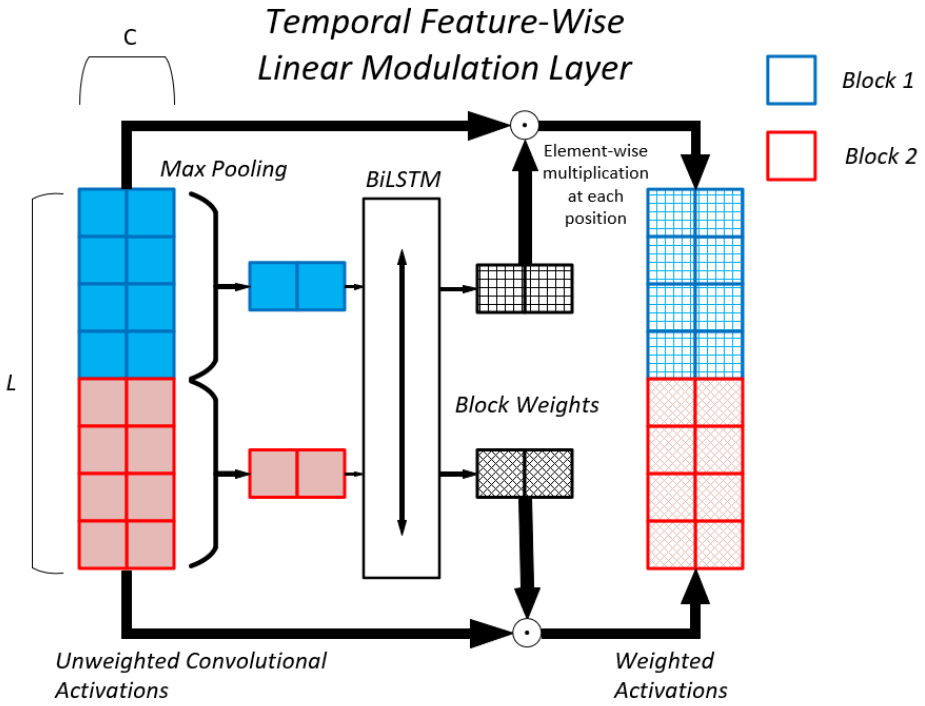

Figure 3.5: The TFiLM layer in detail. In this example the 1D tensor of convolutional activations has the following shape: $F \in \mathbf{R}^{8\times2}$. The 2 blocks are first processed by a Max Pooling layer (whit a pooling factor of 2) and then by the RNN. Finally, the output of the recurrent network is used to modulate the convolutional layer.From [46].

A significant difference between TFNet and TFiLM lies in the fact that, while the former architecture is domain-dependent, the latter is domain-agnostic. Indeed, TFNet performs

domain-specific operations on signals such as DFT, while TFiLM layer does not require any of these operations. This fact has made the application of this model possible with regard to a wide variety of tasks. In general, TFiLM can be useful to process any sequential data. In fact, in the paper, the model's effectiveness is demonstrated on three diverse domains: Text Classification, Audio SR and Chromatin Immunoprecipitation Sequencing. The latter is a particular genomic experiment that consists in reconstructing high-quality measurements taken using a large set of probes (e.g., sequencing reads) from noisy measurements taken using a small set of probes.

It is interesting to point out that, for the researchers, Audio SR and Chromatin Immunoprecipitation Sequencing tasks belong to the same class of generative modeling tasks, called *time series super-resolution*. This problem consists in reconstructing a HR signal from LR measurements.

A brief overview of the model architecture proposed for time-series SR tasks is given in Figure 3.6. We can call this model TFiLM Net.

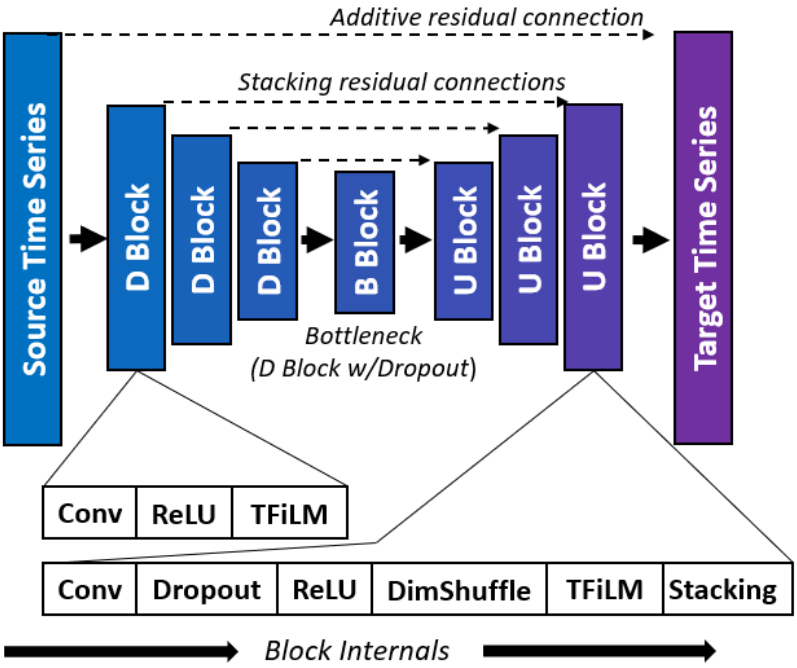

Figure 3.6: *Top*: TFiLM Net architecture used for audio super-resolution. It consists of K downsampling blocks followed by a bottleneck layer and K upsampling blocks; features are reused via symmetric residual skip connections. *Bottom*: Internal structure of downsampling and upsampling convolutional blocks. From [46].

As we can see, the architecture of TFiLM Net follows design patterns of AudioUNet

(described in 3.1.1). Indeed, we can recognize the bottleneck architecture, the subpixel layer, the use of B successive upsampling/downsampling blocks and the adoption of residual connections. This is not surprising because three of the authors are the same for both articles (i.e. [39], [46]).

However, there are several differences between AudioUNet and TFiLM Net, such as the use of recurrent and Feature-Wise Linear Modulation layers. Moreover, while AudioUNet uses standard convolutional layers, TFiLM Net exploits dilated convolutional approaches to extract highly representative features from a wider receptive field. Dilated convolution is described in detail in section 3.2.1.

As for implementation details, it is essential to note that the authors instantiated the model with $K = 4$ (recalling that K is the number of upsampling and downsampling blocks). Another important hyperparameter is the number of TFiLM layer blocks $L = \frac{T}{B}$. In this regard, the block length values B are adjusted for each layer so that the ratio between the temporal dimension T and B is always 32. In other words, the number of blocks L is set to 32 in each TFiLM layer.

In order to facilitate understanding of the whole process, a graphical representation of the entire entire pipeline of TFiLM Net is proposed in Figure 3.7.

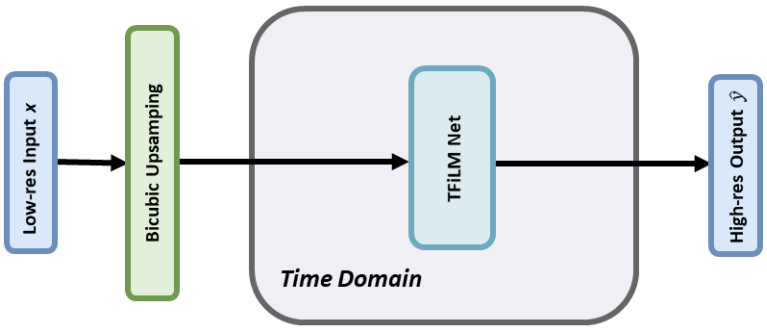

Figure 3.7: Overall pipeline of TFiLM Net. The system integrates convolutional and recurrent layers to efficiently incorporate long-term input dependencies by operating in the time-domain.

3.2.1 Dilated Convolution

To further increase the receptive field of convolutional layers, authors use dilated convolutions with a dilation factor of 2.

Dilated convolution [31] is a generalization of the familiar discrete-convolution (that can be considered as the 1-dilated convolution). In summary, this particular type of convolution, defines a spacing between the values in a kernel. This allows to increase the receptive field of a convolution, without a corresponding increase in the number of trainable parameters. Therefore, the choice to use this type of layer is perfectly consistent with the objectives of this work.

For example, considering the two-dimensional convolution, a 3×3 kernel with a dilation rate of 2 results to have the same receptive field of view as a 7×7 kernel. A graphical representation of this example is provided in Figure 3.8.

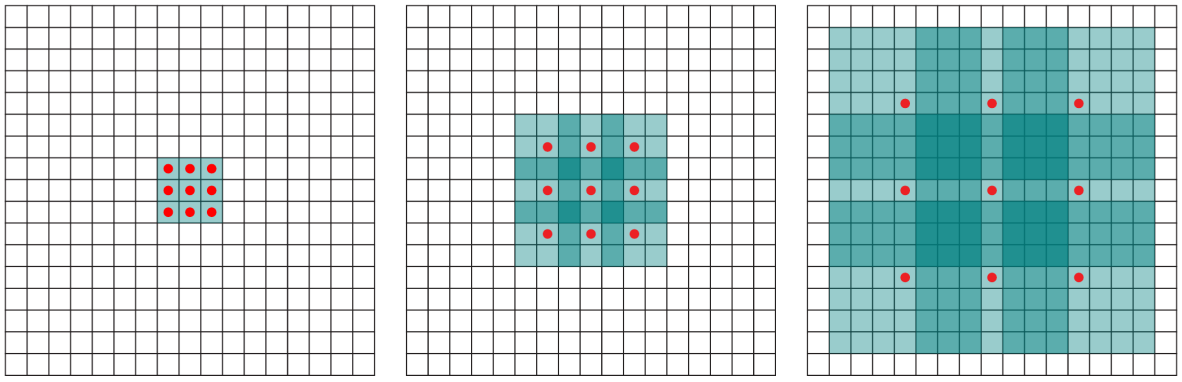

Figure 3.8: (a) 1-Dilated Convolution has a receptive field of 3×3. (b) 2-Dilated Convolution has a receptive field of 7×7. (c) 4-Dilated Convolution has a receptive field of 15×15. The number of parameters associated with each layer is identical. The receptive field grows exponentially while the number of parameters grows linearly. From [31].

3.2.2 Feature-Wise Linear Modulation

As the name suggests, a FiLM layer applies a feature-wise affine transformation, based on conditioning informations of an auxiliary input $z \in \mathbf{Z}$. FiLM can be thought of as a generalization of Conditional Normalization [33], that is used in DL for stabilizing the training of the models. Batch Normalization and FiLM are formally described as follows. Consider a tensor of 1D multichannel convolutional activations $F \in \mathbf{R}^{T \times C}$ (where T is the time axis and C is equal to the number of channels). The batch normalization rescales F and applies an affine transformation, so that it produces an output $F' \in \mathbf{R}^{T \times C}$ whose

(t, c) -th output elements are described in Equation 3.5:

$$F'_{t,c} = \gamma_c \hat{F}_{t,c} + \beta_c \tag{3.5}$$

Where $\hat{F}_{t,c}$ is defined as follows.

$$\hat{F}_{t,c} = \frac{F_{t,c} - \mu_c}{\sigma_c + \epsilon} \tag{3.6}$$

From the notational point of view, it is important to define μ_c, σ_c that are, respectively, the estimates of the mean and standard deviation for the c -th channel. As for $\gamma, \beta \in \mathbf{R}^C$, they are trainable parameters.
FiLM [44] extends this idea by allowing γ, β to be functions $\gamma, \beta : \mathcal{Z} \to \mathbf{R}^C$ of an auxiliary input $z \in \mathcal{Z}$. This technique is domain independent, i.e. can be used for a wide variety of applications. For example, in feed-forward image style transfer [33], z is an image defining a new style; by using different $\gamma(z), \beta(z)$ for each z, the same feed-forward network (using the same weights) can apply different styles to a target image. FiLM has also obtained significant results in speech recognition [38] and image classification [41].

3.3 Proposed Model Architecture

The main objective of this paragraph is to provide a detailed explanation of the proposed model, that aims to combine elements of both TFNet and TFiLM Net in an effective way. Taking inspiration from TFNet, the proposed deep network architecture is made of two branches: one for time-domain and one for spectral-domain processing. Indeed, Lim, Yeh *et al.* demonstrate that this strategy helps the model to better estimate the high-frequency content of the signal. However, their approach is fully convolutional. In this regard, Birnbaum, Kuleshov *et al.* highlight that CNN-based methods have intrinsic limitations when applied to sequential data. As mentioned before, CNNs are not effective for capturing long-term dependency due to the relatively high computational requirements of large receptive fields. To address this problem, the proposed model incorporates TFiLM components in the processing. More specifically, the AudioUNet components used in TFNet are replaced with the TFiLM layers, that can capture long-range input dependencies by modulating the activations of a convolutional layer through a RNN.
To further capture the low-high frequency dependency in the signal, it is decided to keep the spectral replicator layer in the model architecture. It would also be interesting to

evaluate the impact of this operation by training another model that does not have this layer in its pipeline.

To sum up, the proposed approach mainly aims to overcome the limitations of the TFNet architecture, with which it shares a similar pipeline. Indeed, key components, such as the Spectral Replicator and the Spectral Fusion Layer, are maintained in the system.

A graphical representation of the model pipeline is provided in Figure 3.9.

From a notational point of view, the term "Slim TFiLM Net" indicates that the original dimension of the network is reduced by halving the number of filters in each branch.

The Slim TFiLM Nets in the two branches are dimensionally equivalent: both the size and the number of filters applied in each layer are the same. What changes is the length of the input sequence. Indeed, as in the case of TFNet model, a DFT on the sequence is computed to pass from the time to the spectral domain. Because of this, the spectral branch processes a signal that is half the length of the time branch signal.

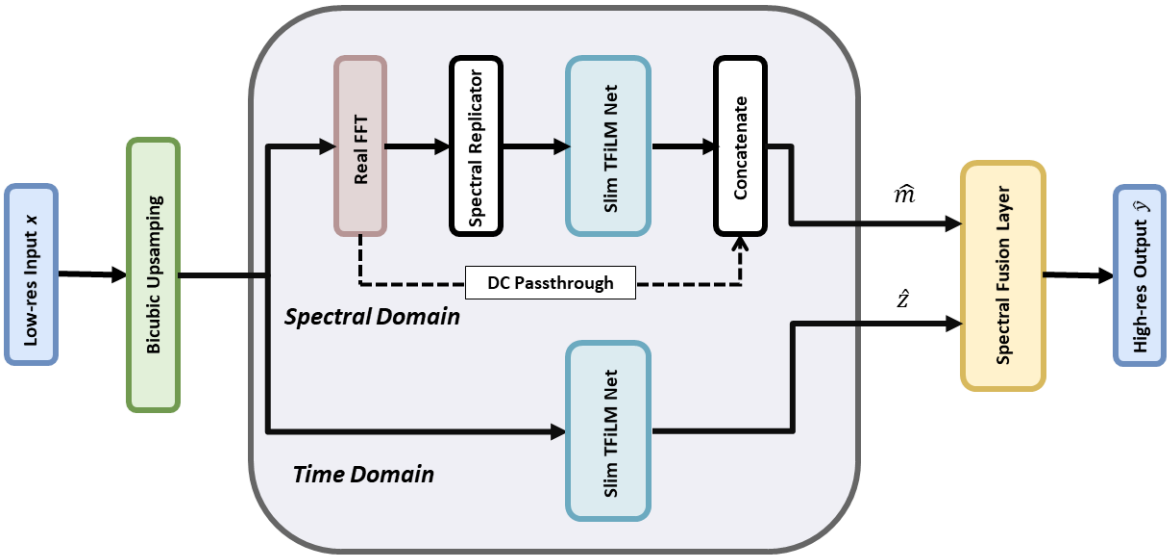

Figure 3.9: Overall pipeline of the proposed model. On one hand, the system maintains the branching structure of TFNet that allows to processes audio in both time and frequency domain. On the other hand, the model uses TFiLM Net to predict both the audio reconstruction and the spectral magnitude.

3.4 Implementation details

As mentioned earlier, in order to identify which is the best architecture between different models, it is important to establish a context that is as fair as possible for the comparison. In this sense, the number of trainable parameters can serve as a benchmark criterion for comparing different results.

Here, however, there is a problem. The available computational resources do not allow to propose and train a model with the same dimensions of TFNet and TFiLM Net. In fact, these two models have roughly 35M parameters, while the available resources allow to train models with approximately 22M parameters. This gap is too wide to compare the results reported in the papers [42], [46] with the novel model architecture. For this reason, it is decided to reduce the dimensionality and re-train both TFNet and TFiLM Net.

To achieve this goal, we proceed in the following way. Recalling that the DL components of the models, i.e. AudioUNet and TFiLM, have a bottleneck architecture that consists of downsampling and upsampling blocks, it is decided to remove the last layer (which is the largest) from all of these components. By doing so, the number of upsampling and downsampling blocks (called B in 3.1.1 and K in 3.2) is reduced from 4 to 3.

This change is enough to enable the model training within the computational limits available to us. Therefore, all other architectural characteristics of the models remain faithful to the original implementation.

Another difference from the papers [42], [46] concerns the data used for the training: the dataset is the same as the one used in both papers, but in this project we sample a subset of recordings because of dimensionality issues. This is explained in more details in Chapter 4.

Chapter 4

Experimental Results

In this chapter, the results of the experiments performed to validate and test the methods presented in Chapter 3 are discussed with the aim to establish the best model architecture. In order to do so, it is first necessary to briefly introduce the experimental setup on which this model comparison is conducted.

Each model is evaluated under two different regimes: *Multi-Speaker* and *Single-Speaker*. The former task aims to develop a system able to generalize the artificial BWE to both female and male speakers with different accents. The latter one, as the name suggests, takes into account only data coming from a single speaker. Specifically for our experiments, the first speaker of the VCTK dataset. Both tasks are addressed in the reference papers too ([42], [46]).

As for software tools, the whole project is written in Python. This programming language is well suited to achieve the desired development purposes, since it offers a diverse set of tools for both DSP and DL applications. The main packages used in this work include Tensorflow, Scipy and Librosa.

We use Google Colaboratory (Colab), a cloud-based service accessible through a web browser. The key advantage of this environment is that it provides up to 25GB of RAM, and a free access to the high-end GPUs. This helps in reducing the computational time as GPUs are very powerful for massively parallel computations, such as the ones required for tensors (signals' data structure) processing.

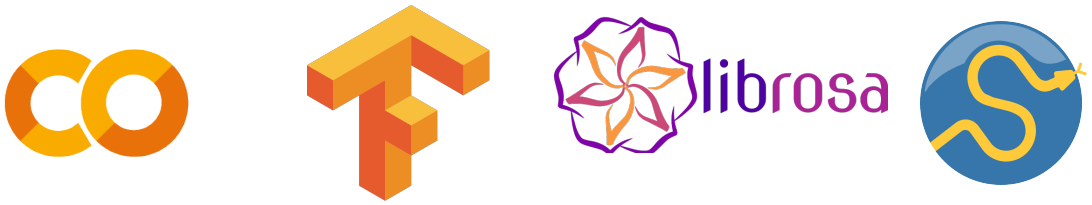

Figure 4.1: From the left, the logos of Colab, Tensorflow, Librosa and Scipy.

4.1 Experimental Setup

This paragraph aims to provide a clear specification of the experimental setup. A full description of the dataset and its preparation is provided along with the metrics used to evaluate the models.

4.1.1 Dataset and preparation

Experiments reported in this thesis are undertaken using the most popular dataset for the audio SR task, i.e. VCTK [51]. In recent years, many researchers have used it (see e.g., [46], [42], [48], [45]), to such an extent that we can now consider this dataset as a benchmark for the BWE problem.

This VCTK Corpus contains approximately 44 hours of speech data uttered by 109 English speakers, each of whom reads out about 400 different sentences. The dataset was initially designed for training Text-to-Speech synthesis systems. In fact, each recording is associated with the corresponding textual transcription. This data are used in this work in order to evaluate models' performance over the Speech-to-Text (STT) task, whose details are discussed in section 4.1.3.

The VCTK Corpus encompasses speakers with various accents, as shown in Figure 4.2.

All speech data are clean, and they were originally recorded using an identical recording setup, i.e. 96 kHz sampling frequency at 24 bits in the same room, with the same microphone. All recordings were finally converted into 16 bits and downsampled to 48 kHz; this is the quality of data available online.

All recordings are further downsampled to 16 kHz for this project: it is possible to consider this sampling rate as R_2, i.e. the one that defines the ground truth $y = (y_{1/R_2}, y_{2/R_2}, \ldots, y_{R_2S/R_2})$. It follows that, for a scaling ratio of $4\times$, R_1 is equal to 4 kHz. It is noticeable that 16 kHz can be reasonably considered as a high quality standard for speech processing applications [17]; suffice to say that a standard telephone audio has a sampling rate of 8 kHz and 16-bit precision [49].

As previously motivated, computational resources for this project are limited: in the multi-speaker task, it is not possible to use the whole dataset, as both reference papers, i.e. [42] and [46], do. In particular, authors of the reference papers train the models for this task on the first 100 VCTK speakers and test on the 9 remaining ones. We re-train their models, and the proposed system, on a sample of 40 speakers. However, the test set used in this project remains the same as the one used in both articles. By doing so,

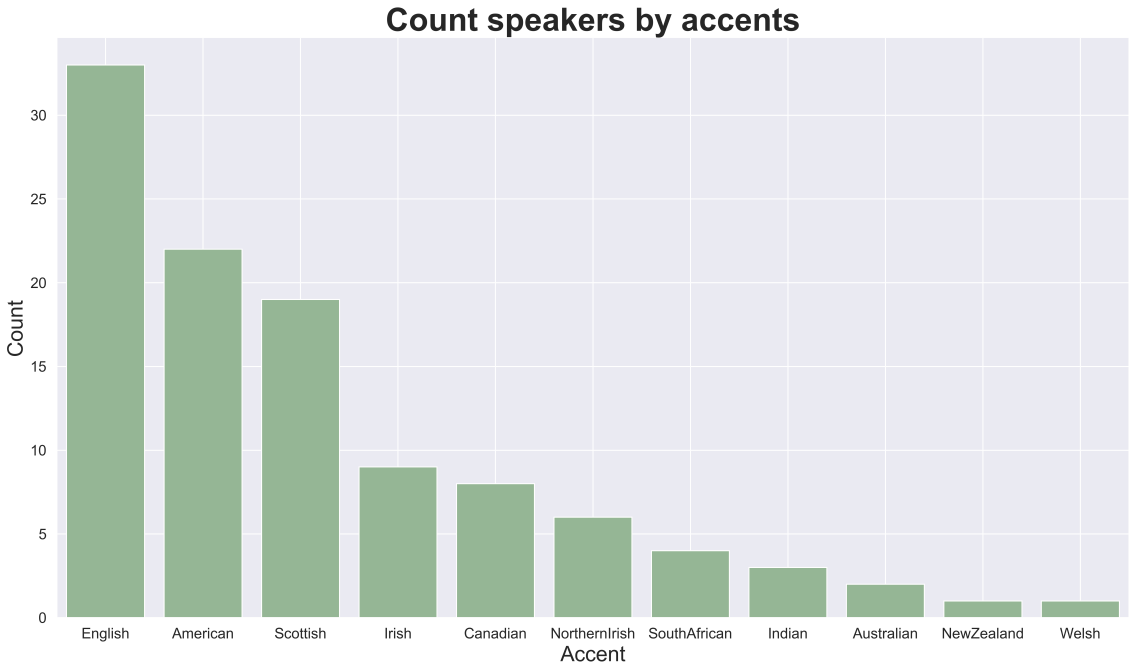

Figure 4.2: Bar chart showing the count of speakers in VCTK dataset based on their accent.

the comparison with newly developed models is placed in a fair context. Furthermore, the difference between the performance - of TFNet and TFiLM - obtained in the original works and those obtained in this project, can be used to measure the negative impact that a drastic reduction in the dimensionality of the problem inevitably has on the performance of the system.

In addition, we also make use of a validation set, the importance of which is explained in section 4.1.2. It is important to highlight that there is no speaker overlap between train, validation and test partitions. This setting is selected in order to reduce performance over-estimation due to potential auto-correlations among different observations pertaining to the same speaker.

Both train and validation sets are sampled in a principled way: the main goal of this sample design is to maximize the model robustness with respect to the main factors of variation. When analyzing a speech recording, important factors concern, for example, the speaker's sex and their accent. Thus, speakers are selected in order to ensure a balanced distribution by gender and a heterogeneous distribution of accents. From a theoretical point of view, this should encourage the creation of a DL model that is robust with respect to variations of accent and gender.

The sampling performed also reflects the age distribution of the original dataset (espe-

cially for the female gender), as can be seen from Figure 4.3.

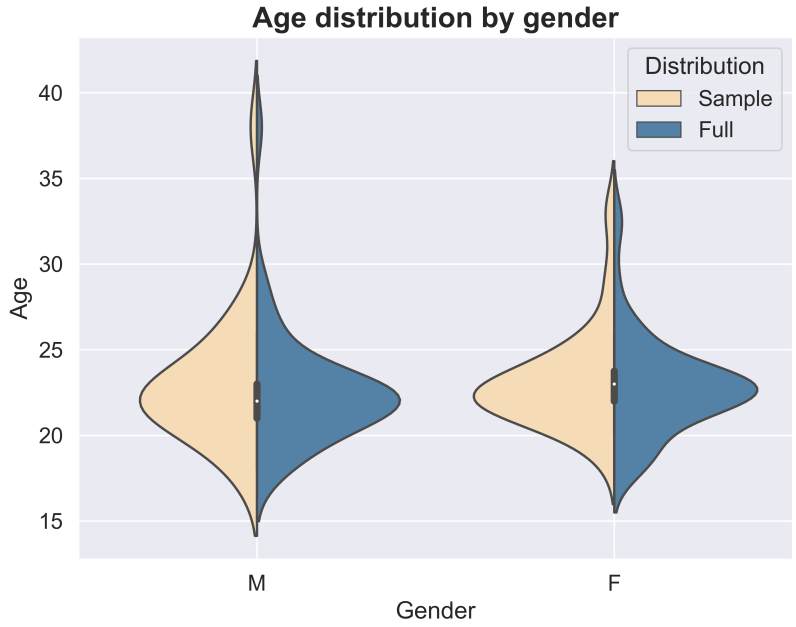

Figure 4.3: Violin plot showing the age distribution of speakers in VCTK dataset based on level of gender.

As for the single-speaker task, we follow the previous works [42] by indicating with "VCTKs" the subset of the whole dataset which only includes the first speaker. It is worth mentioning that this subset does not belong to the set of data used to train, validate or test the models for the multi-speaker task. The train-test split criterion is the same as the one used in both reference papers, i.e. a standard 85% training - 15% test split with no audio-clips overlap.

Furthermore, we keep the same data structure used in both TFNet and TFiLM works: we extract from each recording a set of one-dimensional vectors of length 8192. In particular, each audio clip is divided into several subsequences by the use of a sliding window approach such that each subsequence (or *patch*) shares 75% of its content with the previous one. Therefore, recalling that the sampling rate of our data is 16 kHz, this corresponds to patches of approximately 500ms with the start of every patch 125 ms apart from the previous one.

The corresponding LR patches are obtained by downsampling the original recordings. In particular, we use an order 8 Chebyshev type I low-pass filter with cut-off frequency at $R_{1_{lim}}$. Recalling that in our setup we have $R_1 = 4$ kHz, it follows that $R_{1_{lim}} = 2$ kHz. In other words, the low-pass filter allows to obtain the LR signal by cutting off the frequency

content of the HR recording over the interval $[R_{1_{lim}}, R_{2_{lim}}] = [4, 16]$ kHz.

Another preprocessing step consists in filtering silence sequences by discarding 256-length audio frames whose root-mean-square energy is below a fixed threshold of 0.05. Authors of [42] find that this improves training convergences and stabilizes the gradient. In fact, conceptually it makes perfect sense not to train the model to increase the quality of silent audio data frames. This operation is performed only for the training set, and only in the multi-speaker task.

As mentioned in Chapter 3, we pre-process the LR sequence via a cubic B-spline interpolation in order to obtain (LR,HR) training pairs of the same length. This is one of the best approaches, among all non-adaptive techniques, that can be used to increase the temporal dimension of the input [21].

Dataset statistics for both single and multi-speaker tasks are provided in Table 4.1.

Experiment	**Portion**	**n. of patches**
Single-Speaker	Train	6,656
	Test	768
Multi-Speaker	Train	129,920
	Validation	40,448
	Test	78,592

Table 4.1: Data partition details for both *Single-Speaker* and *Multi-Speaker* tasks.

4.1.2 Training Details

This paragraph focuses on the training of the three models and the choice of their hyper-parameters.

Following the previous works ([42], [46]), we train all models using the Adam optimizer [24] with learning parameters $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-7}$ and a batch size of 128.

As for the learning rate, we use different approaches; Lim, Yeh et al.[42] propose a starting learning rate of 3×10^{-5} with a polynomial decay scheduling with rate of 0.5. We use this setup for both TFNet and the proposed model. As for TFiLM Net, we use the constant value of 10^{-5} which is proposed by its authors. Consequently, this causes a different time for training: TFiLM net convergence speed is slower because of its lower learning rate. In fact, the total number of epochs to finish the training is higher in TFiLM net, as we can see in Figure 4.5.

In this regard, it is important to highlight how the criterion for determining the total

number of steps is chosen: for this decision we use a validation set along with the early stopping criteria. Intuitively, it is possible to consider this setting as a hyperparameter selection method, where the total number of steps is the hyperparameter to be optimized. The interruption criteria is as follows: the model training is interrupted if neither the loss nor the SNR (which is described in paragraph 4.1.3) improve within 25,000 consecutive steps (which correspond to 24 epochs). Training curves for the Multi-Speaker task are shown in figures 4.4, 4.5, 4.6.

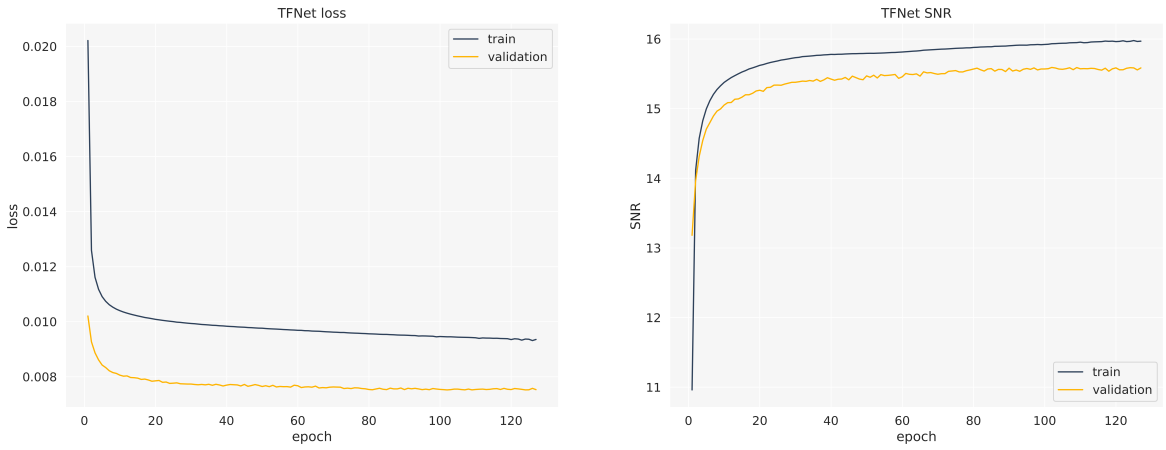

Figure 4.4: TFNet training curves on both training and validation sets. The model is trained for 127 epochs.

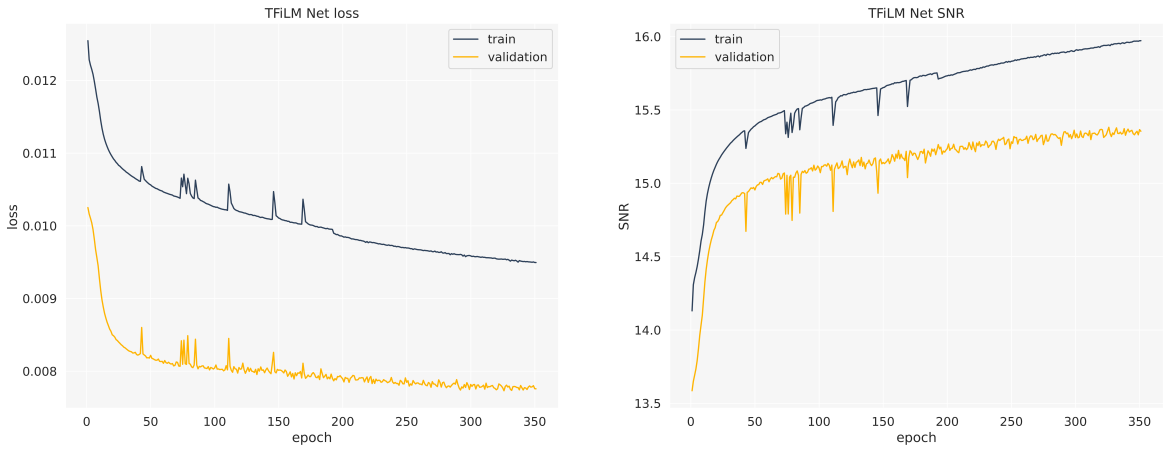

Figure 4.5: TFiLM Net training curves on both training and validation sets. The model is trained for 351 epochs.

We can see that although the loss values in the validation set are better than in the training set, the SNR value is generally lower. This may be due to the different data

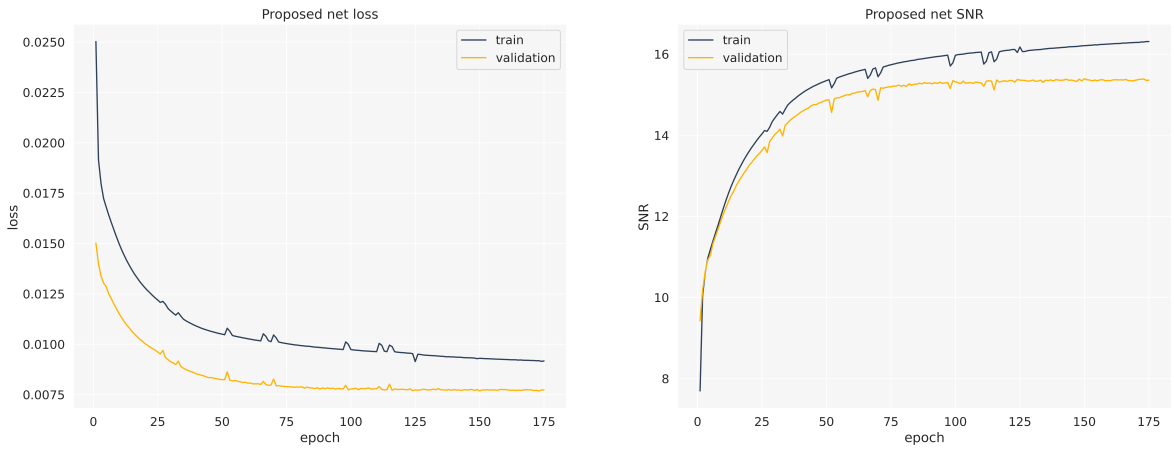

Figure 4.6: Proposed net training curves on both training and validation sets. The model is trained for 175 epochs.

distribution in the two sets; in fact, we remind that in the validation set there are more silence audio frames, which instead are filtered from training data.

Another relevant aspect is the following: initially, each model is trained on the Multi-Speaker task. Subsequently, Single-Speaker models are obtained through a *fine tuning* operation of the previously trained models. Fine tuning consists in training a pre-trained model on a smaller dataset. More specifically, it is the practice of using pre-trained weights from a model trained on a large dataset (in our case, the VCTK corpus) as a starting point for a different dataset (VCTKs). By doing so, we exploit a greater power of generalization (the one of Multi-Speaker models) instead of starting from scratch.

It is worth stressing that the optimizer, the number of training steps, the learning rate, and the other hyperparameters are kept the same for both Single-Speaker and Multi-Speaker regimes. Consequently, this results in a larger number of epochs for the Single-Speaker task since VCTKs is significantly smaller than VCTK.

Finally, we mention that in the Multi-Speaker we select as the final model configuration the one which corresponds to the last validation set improvement on either the loss or the SNR values. On the other hand, in the Single-Speaker task, since there is no validation set, we simply take the model weights at the last epoch.

4.1.3 Evaluation Methods

In this section the metrics used to evaluate the models introduced in Chapter 3 are presented. We use two standard metrics used in the audio SR literature such as SNR and LSD [2]. While the former takes into account a weighted difference between the model

signal reconstruction and the ground-truth data in the time domain, the latter measures the reconstruction quality in the frequency domain.

More specifically, SNR is the ratio, usually expressed on a logarithmic scale in decibels, between the signal power level and the noise power level. Formally, given a reference signal y and an approximation $\hat{y}$, the Signal-to-Noise ratio SNR is defined as:

$$\mathrm{SNR}(\hat{y}, y) = 10 \log \frac{\|y\|_2^2}{\|\hat{y} - y\|_2^2} \tag{4.1}$$

As for LSD, a formal definition is as follows. Let X and $\hat{X}$ be the log-spectral power magnitudes of, respectively, the reference signal y and its approximation $\hat{y}$. These are defined as $X = \log |S|^2$, where S is the Short-Time Fourier Transform (STFT) of the signal. Then, the LSD can be calculated as:

$$\mathrm{LSD}(\hat{y}, y) = \frac{1}{L} \sum_{\ell=1}^{L} \sqrt{\frac{1}{K} \sum_{k=1}^{K} (X(\ell, k) - \hat{X}(\ell, k))^2} \tag{4.2}$$

where ℓ denotes the index of short windowed frames of the audio and k denotes frequencies; in our experiments (as well as in [39], [42], [46]) we use frames of length 2048.

Furthermore, we investigate the effect of each model in the context of a wider system, which also includes a model of STT: as said in Chapter 1, a BWE algorithm has the potential to help in better performing the speech-to-text conversion task by improving the quality of input recordings.

Our objective is to investigate whether or not the results of a STT system obtained on a LR input signal improve after a SR operation is performed. Thus, we use one of the state-of-the-art open-source STT engines, i.e. *Deep Speech* [23] to process the recordings and obtain their textual transcription.

In order to quantify the word level mismatch between real and predicted textual transcriptions we use a standard evaluation metric such as WER. According to [13], we can define WER as the proportion of word errors to words processed. More specifically, let denote with H, S, D, I the total number of word hits, substitutions, deletions and insertions (see Fig. 4.7). Then the WER can be calculated as:

$$\mathrm{WER} = \frac{S + D + I}{H + S + D} \tag{4.3}$$

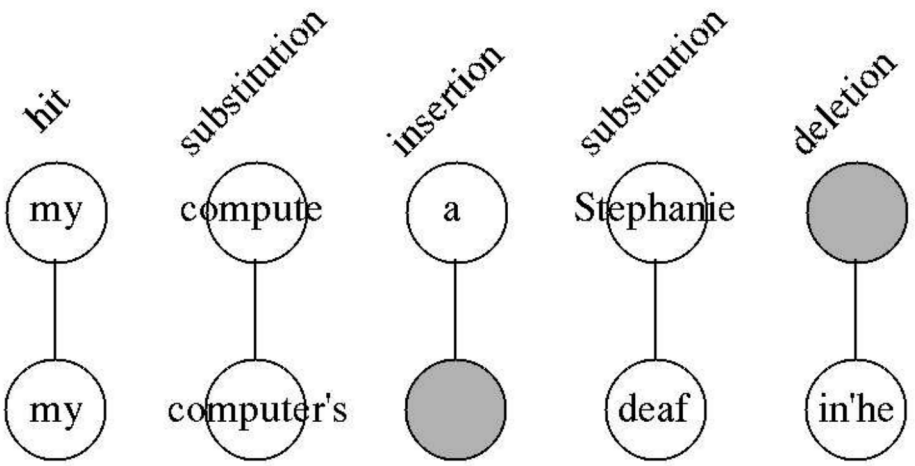

Figure 4.7: Example of H, S, D, I classification on the input words "my computer's deaf in'he?". From [13].

An important aspect is that it is necessary to apply some appropriate pre-processing steps on both the predicted and the ground-truth transcription text. In particular, these operations consists in removing leading and trailing spaces, reducing text to lower case, expanding all contractions and removing punctuation.
Finally, it is important to highlight that, while SNR and LSD are calculated on the test set patches, WER is calculated on the test set full recordings.

4.2 Results

This section describes and discusses the main results achieved in the reported experiments in order to establish the best model architecture. We provide results on both the Multi-Speaker and the Single-Speaker regimes in order to investigate if the proposed algorithm allows improving the audio SR task.
Our method is compared to three baselines: a cubic B-spline — which corresponds to the upsampling criteria used in all model pipelines (Figures 3.2, 3.7, 3.9) - TFNet and TFiLM Net.
We remind that a higher SNR is better, a lower LSD is better and a lower WER is better. The results of our experiments over the training set are summarized in tables 4.2 and 4.3. As we can see, our solution achieves the best SNR and LSD values on both tasks. However, high training performance are not, in general, always associated with high generalization results. Furthermore, we remind that, in our setting, the training set data distribution is not highly representative of the problem as we filter silence audio frames. Therefore, we use a validation set, which allows to have a more accurate estimate of the

generalization power of the models during the training phase. Validation set results can be seen in Table 4.4; the proposed model outperforms the other models on LSD score, while its SNR is not as good as the TFNet system one.

Single-Speaker *Training Set*				
Obj.	Spline	TFNet	TFiLM Net	Proposed
SNR	14.74	16.93	16.71	**17.33**
LSD	5.64	3.44	3.93	**3.24**

Table 4.2: Evaluation of BWE methods (in dB) on the Single-Speaker task over the training set in terms of SNR and LSD. A higher SNR is better and a lower LSD is better.

Multi-Speaker *Training Set*				
Obj.	Spline	TFNet	TFiLM Net	Proposed
SNR	14.29	15.97	15.96	**16.21**
LSD	6.29	3.69	4.25	**3.37**

Table 4.3: Evaluation of BWE methods (in dB) on the Multi-Speaker task over the training set in terms of SNR and LSD. A higher SNR is better and a lower LSD is better.

Multi-Speaker *Validation Set*				
Obj.	Spline	TFNet	TFiLM Net	Proposed
SNR	13.60	**15.57**	15.35	15.34
LSD	5.97	3.57	4.14	**3.31**

Table 4.4: Evaluation of BWE methods (in dB) on the Multi-Speaker task over the validation set in terms of SNR and LSD. A higher SNR is better and a lower LSD is better.

At this point, we can examine the extent to which the models generalize across the test set. The results of SNR in Figure 4.8 show that TFNet approach is the one which achieves the best values on both tasks. As for the other two methods, they achieve similar performance. Furthermore, we can observe that each model exceeds the spline baseline about more than 1dB.

As for LSD scores, we can see from Figure 4.9 that our system shows an improvement

of 0.1 - 2.2 dB over the baseline methods on the Single-Speaker task and even of 0.3 - 2.3 dB on the Multi-Speaker problem. From an intuitive perspective, we can say that, in our model, the spectral distortions of the predicted speech from the clean speech is less significant at high-frequencies.

These two metrics reflect the potential of the models to help the Deep Speech engine in better performing the STT conversion. In fact, as we can see from Figure 4.10, the results of WER show that the two best models are again TFNet and the proposed one. In particular, we can see that our approach outperforms by approximately 0.05 percentage points the TFNet system, by ≈ 0.08 percentage points the TFiLM Net and by ≈ 0.12 percentage points the spline baseline. Although these results are quite far from the WER obtained on original recordings (which is an estimate of the real potential of the Deep Speech engine), the improvements brought by the audio SR models are remarkable when compared to splines.

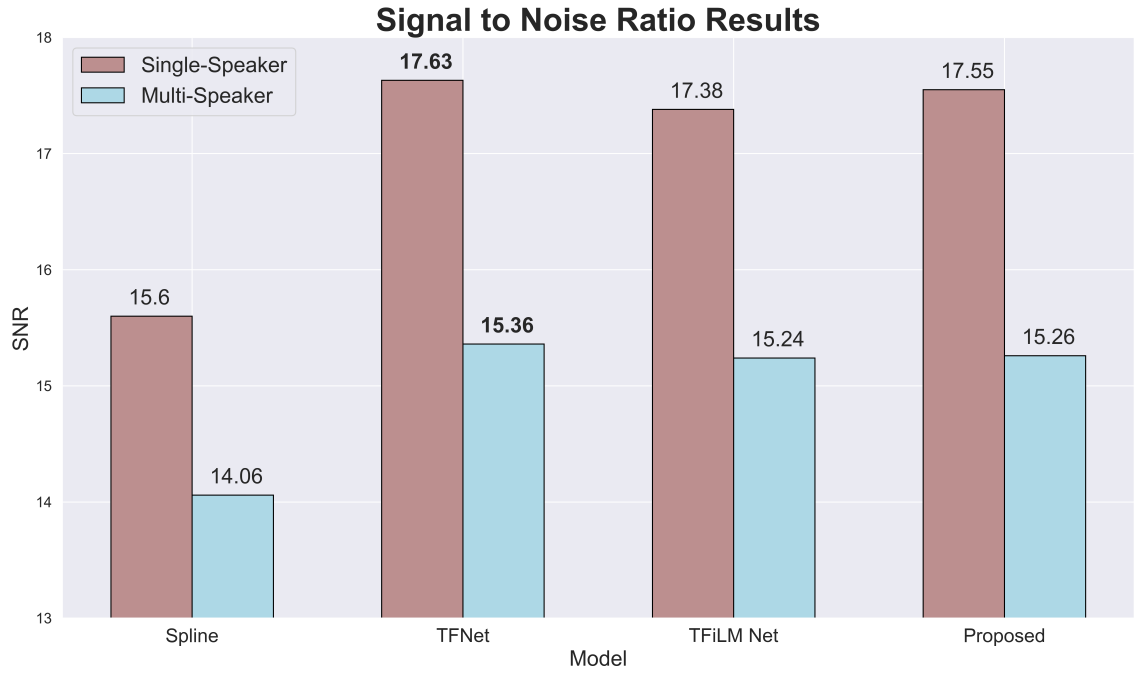

Figure 4.8: SNR results (in dB) on Test Set for both Single-Speaker and Multi-Speaker tasks. Higher is better.

Therefore, it is fair to say that the proposed approach and the TFNet system are the best model according to these quantitative metrics. There is not a clear winner, since the best model changes according to the considered metric. Thus, although TFNet achieves the best SNR, the proposed model shows a better reconstruction quality in the frequency

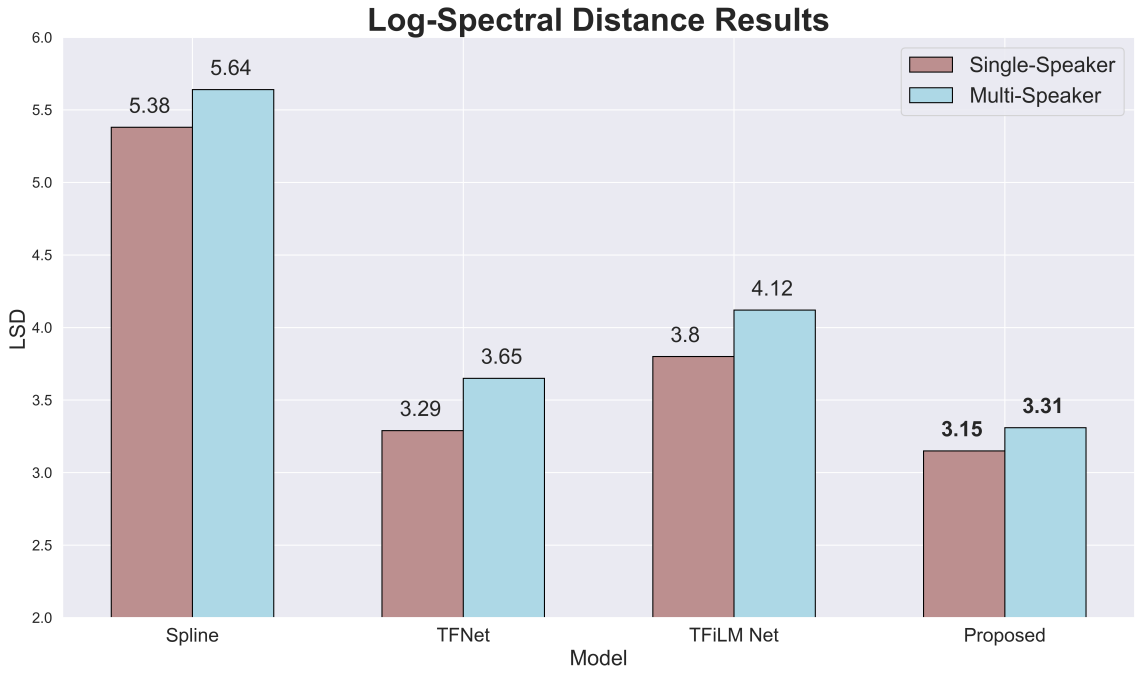

Figure 4.9: LSD results (in dB) on Test Set for both Single-Speaker and Multi-Speaker tasks. Lower is better.

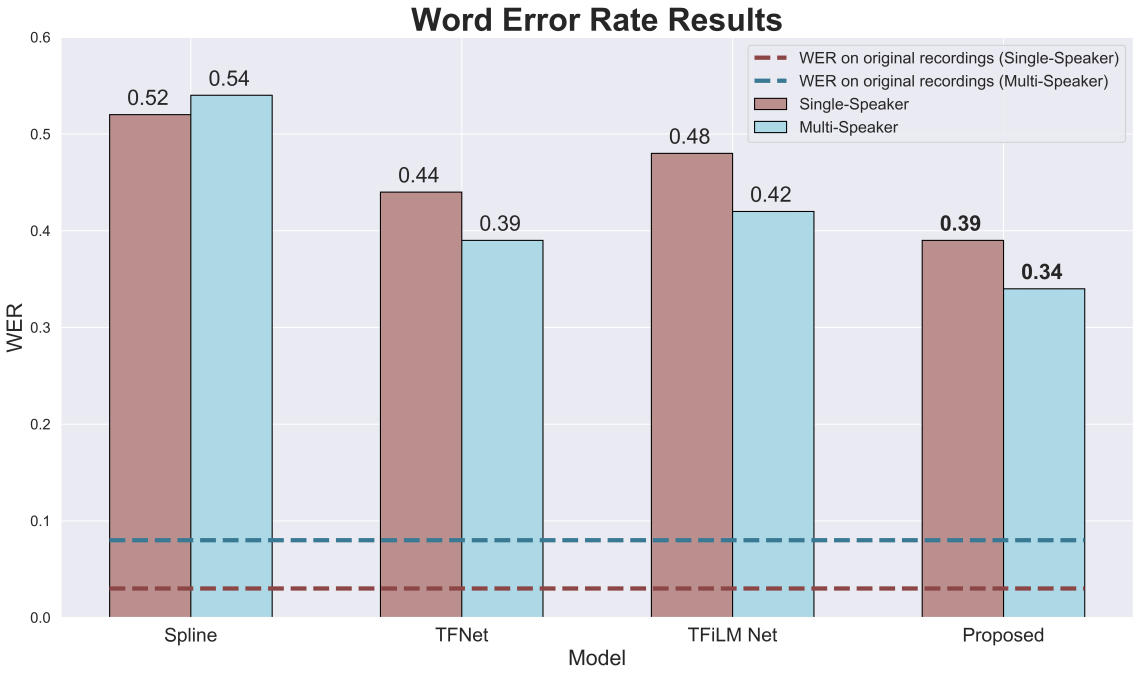

Figure 4.10: WER results (in percentage) on Test Set for both Single-Speaker and Multi-Speaker tasks. WER on original recordings is, respectively, equal to 0.03 and 0.08. Lower is better.

domain. Furthermore, our approach is the one which better helps the Deep Speech engine in converting audio data to text. This latter result is in agreement with the LSD metric evaluation, and suggests that the proposed method provides a better speech quality.
Finally, it is worth observing the audio spectrograms. They show - from up to down in figures 4.11 4.12, 4.13 - a high-resolution signal, its low-resolution version obtained using spline interpolation, and the output of our model. These examples refer to an audio file that belong to the test set of the Multi-Speaker task (speaker ID and audio file ID are, respectively, *p351* and *004*).
It is important to mention that these spectrograms are computed using consecutive Fourier transforms; in particular, we use a *Hamming* window of 512 samples with 50% overlap.
As we can see, the proposed model is generally able to reconstruct a remarkable part of the original signal's spectral content, especially in the frequency range 2 - 4 kHz. It is worth noting that its estimation is mainly focused on the speech frames rather than the silence ones and this makes perfect sense.

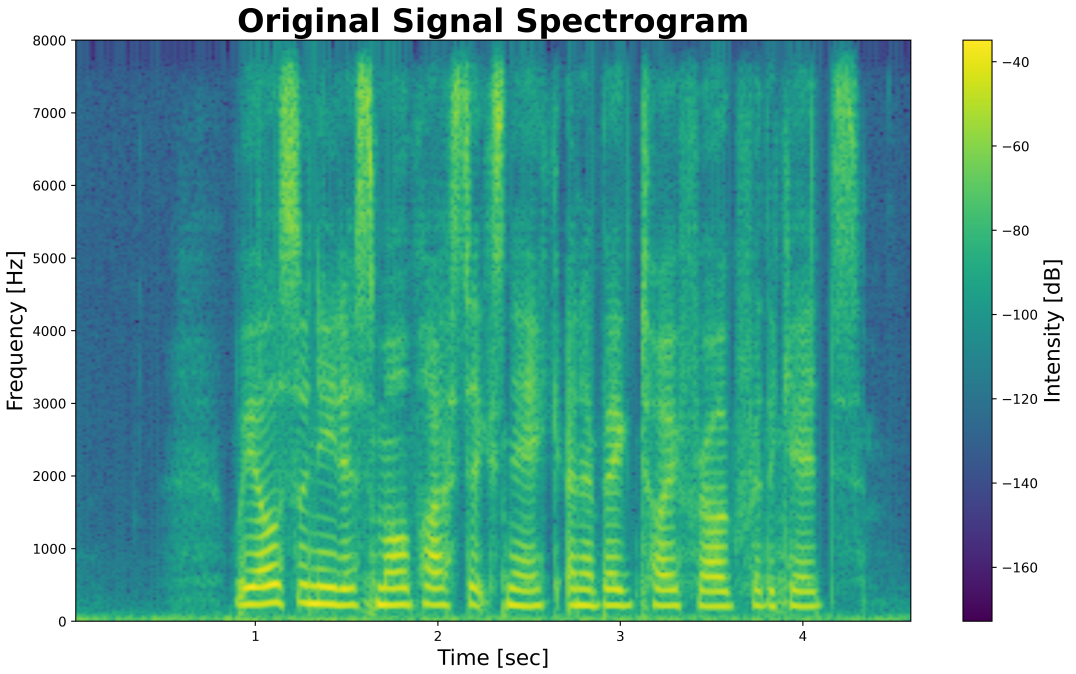

Figure 4.11: Spectrogram showing how the frequency content of a 16kHz signal changes over time.

Spectral Fusion Layer Weights Analysis

We perform an additional analysis on the weights w introduced in the Spectral Fusion Layer (see Equation 3.3). In particular, we remind that w is a trainable vector which establish the weight of each branch for the spectral magnitude prediction; the higher its

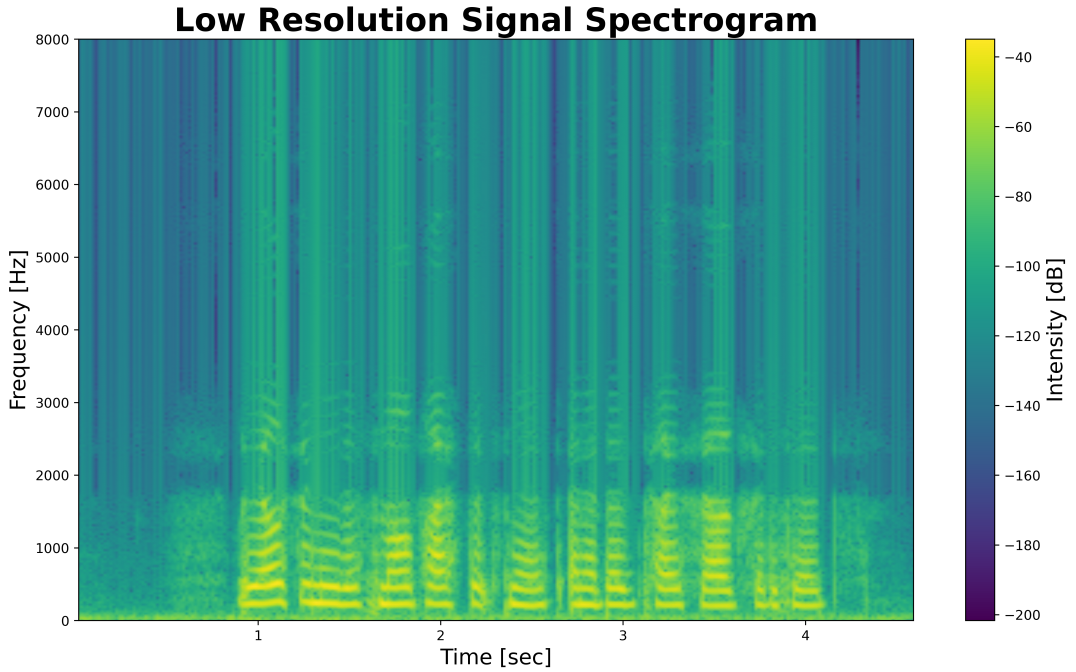

Figure 4.12: Spectrogram showing the spline reconstruction of the signal.

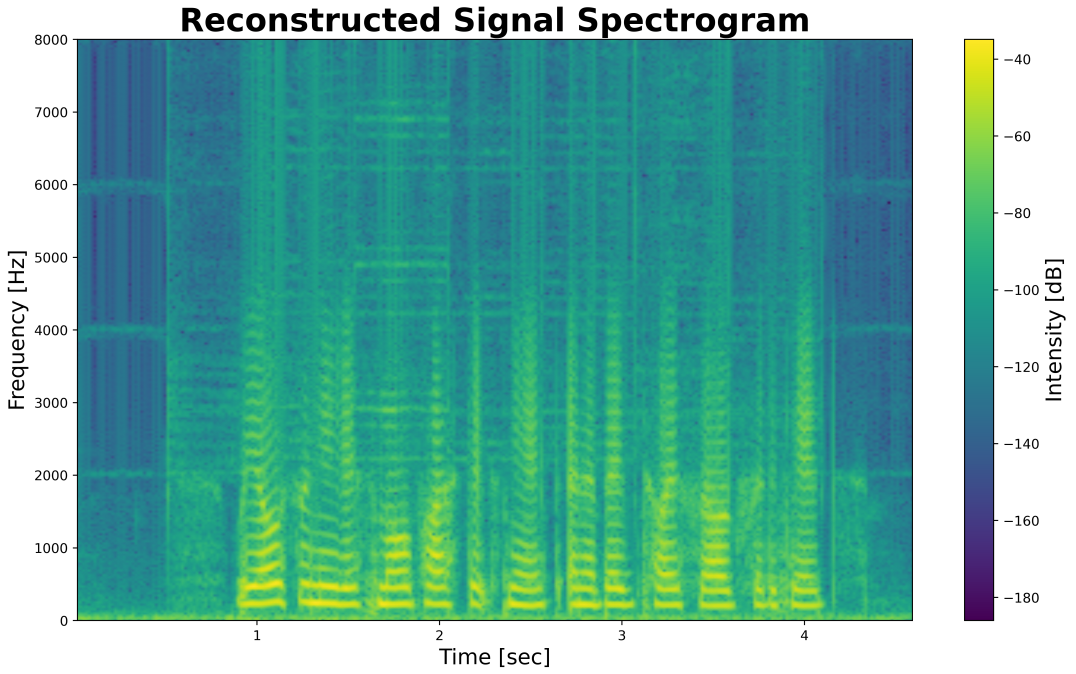

Figure 4.13: Spectrogram showing the model reconstruction of the signal.

values are, the more the weight of the time branch is important in the spectral magnitude M calculation. Our goal is to investigate how these weights are distributed on both the proposed model and the TFNet system.

This analysis is only performed for the Multi-Speaker task; however, it is reasonable to assume that the results are not very different in the Single-Speaker task since the models

are trained using a fine-tuning approach.

The resulted distribution for both the proposed and the TFNet models is provided in Figure 4.14.

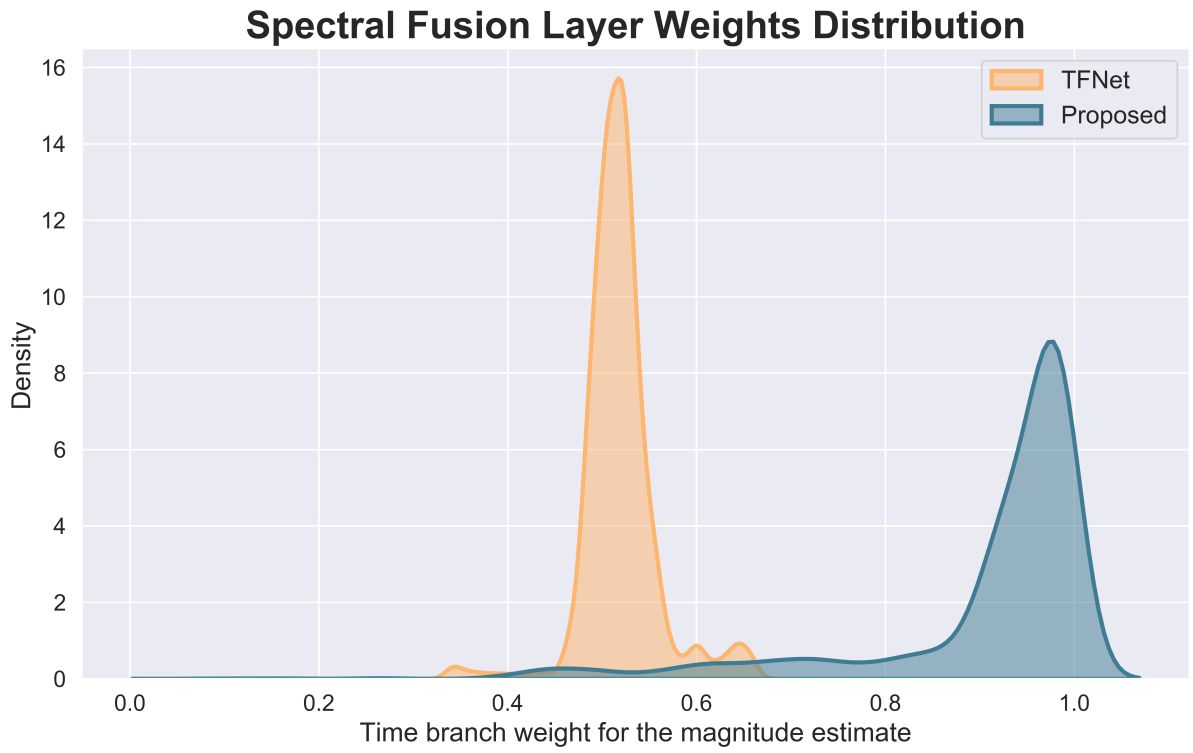

Figure 4.14: Spectral Fusion Layer weights distribution for both the TFNet model and the proposed system.

As we can see, the weights of the two branches are more balanced in the TFNet system than in the proposed model. In fact, the two means are, respectively, $\overline{w}_{TFNet} \approx 0.5$ and $\overline{w}_{Proposed} \approx 0.9$.

Consequently, it is fair to say that, on average, for the TFNet system the two branches contributes equally to the spectral magnitude M estimation. On the other hand, for our model the two branches contribution is heavily skewed.

Although we cannot prove it mathematically, it is reasonable to think that this is a limitation for our model. In fact, the spectral branch, whose only goal is to contribute to the estimate of M, turns out not to be relevant in the final prediction. In other words, since the phase estimation is strictly time-branch dependent, the spectral branch in our model results to be almost irrelevant for the HR reconstruction.

Chapter 5

Conclusions and Further Work

In this thesis we faced the problem of audio super-resolution, i.e. the task of increasing the sampling rate of a given input signal by predicting its missing high-frequency content. More specifically, we aimed to estimate a complex non-linear regression function which maps a given low-resolution input signal to its corresponding high-resolution version.
In order to do so, we focused our attention on deep learning models, which have demonstrated state of-the-art results in many signal processing based applications. More specifically, we worked in a supervised learning setting in which we trained models on a dataset composed by low-resolution, high-resolution 8192-length recording pairs. Input sequences were obtained by downsampling the original signals by a factor of 4; for 16 kHz high-resolution data, this corresponds to patches of approximately 500ms.
Data were taken from the VCTK Corpus, which is a benchmark dataset for this task; briefly, it encompasses dozens of different speakers with various accents.
The proposed model is based on an hybrid architecture that combines two different methods that achieved remarkable results in literature: TFNet and TFiLM Net.
The former system aims to estimate the high-frequency content of an input signal through a branching architecture; authors proposed a model which utilizes both time and frequency domain, in two different branches. In other words, TFNet is an end-to-end system which allows the two domains to be jointly optimized. In the original paper it was demonstrated the superior performance of this strategy with respect to models which operate only in either time or frequency domain.
On the other hand, TFiLM Net is a deep neural network which learns the low-resolution to high-resolution mapping directly in the time domain. Its main contribution lies in the integration of convolutional and recurrent approaches to efficiently incorporate long-range information. More specifically, TFiLM layers modulate the activations of a convolutional

layer through a RNN. This system resulted very effective when processing sequential data, such as audio signals.

The proposed model can be considered as a mix of these two methods. On one hand, the proposed system maintains the branching structure of TFNet which allows to process audio in both the time and the frequency domain. On the other hand, our model uses TFiLM Net modules to process data in the two branches. By doing so, we inherited the advantages of the two methods, such as the branching structure and the adoption of recurrent layers which can expand the limited receptive field of CNNs.

The available resources did not allow to train, validate and test all the models in their original form: therefore, convolutional layers dimension were reduced such that each model shares the same number of trainable parameters.

As stated in Chapter 4, all the training configurations, such as the learning rate, the batch size and the optimizer, were taken from the reference papers [42], [46].

As for the evaluation methods, we used two standard metrics for this task, such as SNR and LSD, and a third criteria which is less common: we investigated whether super-resolution models can help a speech-to-text engine in better performing its automated conversion task. We investigated the effect of the three models by the use of a standard evaluation metric for STT task such as the WER. As a further work, we mention that it would be interesting to include qualitative metrics based on human perception of audio quality. These would certainly be the natural complement of the objective evaluations that we proposed in this work.

Experimental results are quite encouraging: the proposed model outperformed previous super resolution approaches on the LSD metric, which takes into account the reconstruction quality in the spectral domain. On the other hand, we have SNR, which measures the difference between the model signal reconstruction and the ground-truth data in the time domain; according to this metric TFNet was the system which achieves the best value. As for the WER metric, our model showed actual improvements in helping the Deep Speech engine in the STT conversion.

These results suggest that the proposed method provides a remarkable reconstruction quality. However, our Spectral Fusion Layer weights analysis described in Chapter 4 revealed a limitation: the frequency branch of the proposed model, whose only goal is to contribute to the spectral magnitude estimation, turned out to be only partially relevant in the final prediction. This result suggested that the architecture of this branch can be better developed to improve the performance, and this can be considered as one main

improvement of our approach.

Furthermore, it is worth highlighting that computational limitations of the available resources were an important constraint for the achievement of better results. Thus, it would certainly be interesting to evaluate the effect of our approach without a prior drastic dimensionality reduction of both the dataset size and the models' number of parameters.

Bibliography

[1] Jack Kiefer, Jacob Wolfowitz, et al. "Stochastic estimation of the maximum of a regression function". In: *The Annals of Mathematical Statistics* 23.3 (1952), pp. 462–466.

[2] Augustine Gray and John Markel. "Distance measures for speech processing". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 24.5 (1976), pp. 380–391.

[3] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[4] Paul J Werbos. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[5] Shun-ichi Amari. "Backpropagation and stochastic gradient descent method". In: *Neurocomputing* 5.4-5 (1993), pp. 185–196.

[6] Yan Ming Cheng, Douglas O'Shaughnessy, and Paul Mermelstein. "Statistical recovery of wideband speech from narrowband speech". In: *IEEE Transactions on Speech and Audio Processing* 2.4 (1994), pp. 544–548.

[7] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[8] TM Mitchell. *Machine Learning*. McGraw Hill, 1997.

[9] Kun-Youl Park and Hyung Soon Kim. "Narrowband to wideband conversion of speech using GMM based transformation". In: *2000 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 00CH37100)*. Vol. 3. IEEE. 2000, pp. 1843–1846.

[10] Per Ekstrand. "Bandwidth extension of audio signals by spectral band replication". In: *in Proceedings of the 1st IEEE Benelux Workshop on Model Based Processing and Coding of Audio (MPCA'02)*. Citeseer. 2002.

[11] William T Freeman, Thouis R Jones, and Egon C Pasztor. "Example-based super-resolution". In: *IEEE Computer graphics and Applications* 22.2 (2002), pp. 56–65.

[12] Peter Jax and Peter Vary. "Artificial bandwidth extension of speech signals using MMSE estimation based on a hidden Markov model". In: *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03)*. Vol. 1. IEEE. 2003, pp. I–I.

[13] Andrew Cameron Morris, Viktoria Maier, and Phil Green. "From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition". In: *Eighth International Conference on Spoken Language Processing*. 2004.

[14] Federico Avanzini and Giovanni De Poli. *Fundamentals of digital audio processing*. Creative Commons, 2005.

[15] John G Proakis. *Dimitris. G, Manolakis - Digital Signal Processing*. Prentice Hall, New Jersey1996, 2006.

[16] Geun-Bae Song and Pavel Martynovich. "A study of HMM-based bandwidth extension of speech signals". In: *Signal Processing* 89.10 (2009), pp. 2036–2044.

[17] Stefan Steidl. *Automatic classification of emotion related user states in spontaneous children's speech*. Logos-Verlag, 2009.

[18] Hannu Pulakka et al. "Speech bandwidth extension using gaussian mixture model-based estimation of the highband mel spectrum". In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2011, pp. 5100–5103.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[20] Mohamed Farouk Abdel Hady and Friedhelm Schwenker. "Semi-supervised learning". In: *Handbook on Neural Information Processing*. Springer, 2013, pp. 215–239.

[21] Dianyuan Han. "Comparison of commonly used image interpolation methods". In: *Proceedings of the 2nd international conference on computer science and electronics engineering*. Atlantis Press. 2013, pp. 1556–1559.

[22] Junyoung Chung et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[23] Awni Hannun et al. "Deep speech: Scaling up end-to-end speech recognition". In: *arXiv preprint arXiv:1412.5567* (2014).

[24] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[25] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[26] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[27] Timothy Dozat. "Incorporating Nesterov Momentum into Adam". In: 2015.

[28] Kehuang Li et al. "DNN-based speech bandwidth expansion and its application to adding high-frequency missing features for automatic speech recognition of narrow-band speech". In: *Sixteenth Annual Conference of the International Speech Communication Association*. 2015.

[29] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[30] Ren Wu et al. "Deep image: Scaling up image recognition". In: *arXiv preprint arXiv:1501.02876* 7.8 (2015).

[31] Fisher Yu and Vladlen Koltun. "Multi-scale context aggregation by dilated convolutions". In: *arXiv preprint arXiv:1511.07122* (2015).

[32] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. "Soundnet: Learning sound representations from unlabeled video". In: *Advances in neural information processing systems* 29 (2016), pp. 892–900.

[33] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. "A learned representation for artistic style". In: *arXiv preprint arXiv:1610.07629* (2016).

[34] Ian Goodfellow et al. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016.

[35] Deepak Pathak et al. "Context encoders: Feature learning by inpainting". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2536–2544.

[36] Wenzhe Shi et al. "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1874–1883.

[37] Richard Zhang, Phillip Isola, and Alexei A Efros. "Colorful image colorization". In: *European conference on computer vision*. Springer. 2016, pp. 649–666.

[38] Taesup Kim, Inchul Song, and Yoshua Bengio. "Dynamic layer normalization for adaptive neural acoustic modeling in speech recognition". In: *arXiv preprint arXiv:1707.06065* (2017).

[39] Volodymyr Kuleshov, S Zayd Enam, and Stefano Ermon. "Audio super-resolution using neural nets". In: *ICLR (Workshop Track)*. 2017.

[40] Raymond A Yeh et al. "Semantic image inpainting with deep generative models". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5485–5493.

[41] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-excitation networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7132–7141.

[42] Teck Yian Lim et al. "Time-frequency networks for audio super-resolution". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 646–650.

[43] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. "Which training methods for GANs do actually converge?" In: *arXiv preprint arXiv:1801.04406* (2018).

[44] Ethan Perez et al. "Film: Visual reasoning with a general conditioning layer". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[45] Mu Wang et al. "Speech super-resolution using parallel wavenet". In: *2018 11th International Symposium on Chinese Spoken Language Processing (ISCSLP)*. IEEE. 2018, pp. 260–264.

[46] Sawyer Birnbaum et al. "Temporal FiLM: Capturing Long-Range Sequence Dependencies with Feature-Wise Modulations." In: *Advances in Neural Information Processing Systems*. 2019, pp. 10287–10298.

[47] Mads G Christensen. "What Is Sound?" In: *Introduction to Audio Processing*. Springer, 2019, pp. 1–17.

[48] Sefik Emre Eskimez and Kazuhito Koishida. "Speech super resolution generative adversarial network". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 3717–3721.

[49] Uday Kamath, John Liu, and James Whitaker. "Automatic Speech Recognition". In: *Deep Learning for NLP and Speech Recognition*. Springer, 2019, pp. 369–404.

[50] Arun Rai, Panos Constantinides, and Saonee Sarker. "Editor's comments: next-generation digital platforms: toward human–AI hybrids". In: *Mis Quarterly* 43.1 (2019), pp. iii–x.

[51] Junichi Yamagishi, Christophe Veaux, Kirsten MacDonald, et al. "CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit (version 0.92)". In: (2019).