Università
Ca'Foscari
Venezia

CA' FOSCARI UNIVERSITY OF VENICE

Department of Environmental Sciences, Informatics and Statistics

Computer Science Master's Thesis

Exploring audio compression in time-frequency domain with sparse CNNs

Supervisor
Pistellato Mara
Bergamasco Filippo

Graduand
Giovanni Scodeller

Year
2021 - 2022

Abstract

Audio data compression and decompression are usually implemented via software codecs which are handmade crafted, often exploiting the spectral properties of the signal. In this thesis we propose to tackle such problem as a data-driven approach, considering the time-frequency domain of an audio signal as an intensity map to be reconstructed. The main idea is to mask some input values and then apply sparse convolutional operation in order to perform depth completion and reconstruct the missing signal. In particular, our method is divided into two main parts: first, we explore the feasibility of audio signal compression with sparse convolutions varying the level of missing information; we also explored how different levels of sparsity affect the quality of the final reconstruction in order to choose the most suitable one according to the context. Secondly, we aim at creating an ad-hoc binary mask so that the loss of information during the decompression step is minimized. We set the problem of mask generation as an optimization problem using two different approaches: by solving a minimization problem and via genetic algorithms.

Contents

Chapter 1

Introduction

In the digital age we currently inhabit, nearly all aspects of our lives are stored in digital memories, which are inherently limited. In order to make efficient use of this space, we must employ various forms of file compression. Another example that demonstrates the usefulness of this task can be encountered when we need to share files, the speed at which we can do it becomes a paramount concern and the dimension file plays a major role.

In particular in our work, of all possible file types, we have decided to focus on audio files. Classical audio compression algorithms are usually implemented in software as audio codecs [1, 2, 3, 4], they remove redundant or irrelevant information from audio data in order to save space. There exist two main categories of compression algorithms: lossy and lossless. When an audio file is compressed using a lossy algorithm, the codec will analyze the audio signal, it will discard the information that is considered inaudible or less important to human hearing, then it will compress the remaining information. On the other hand, lossless compression aims to preserve all the information in order to reconstruct the signal when decompression is performed.

In this thesis, we want to explore whether is possible to create a lossy compression and decompression models for audio files exploiting their frequency domain. We start extracting the log-spectrogram of the audio waveform which can be seen as an intensity map of signal frequencies. Using those spectrograms allows us to experiment methods that work on images and intensity maps, more precisely we want to address this problem as a depth completion task. The depth completion task usually works with a sparse input, our compressed spectrogram, and performs an interpolation giving a dense map as output.

As we have formulated the problem, we divide it into two distinct parts: in the first one, we want to use a deep neural network (DNN) model in order to decompress our data, in the second one we want to compress the file creating a sparse spectrogram in order to have the reconstructed audio as good as possible.

We decided to use the DNN for the decompression phase because, with the advancement of technology and computational power, they become more and more popular to solve different problems in different scientific fields. Thanks to their exceptional ability to learn from the data is possible to craft new compression method in a short time with respect to domain-specific research that took years to develop and were largely hand-designed.

For the compression phase, on the other hand, we will create a binary mask, with the same size as the spectrogram, that yields the property of having information, represented with a 1, or not in this case we have a 0. The goal of this work is to compress the spectrograms of a fixed level of sparsity 80%, 85%, 90%, and 95%. If we generate a random uniform mask, we will surely remove the amount of information that we are aiming for, but in the end, we could also eliminate useful information in order to restore the entire spectrogram. In order to improve the quality of the decompression, we want to create an ad-hoc mask for each sample. We formulated the generation of the latter masks with two different approaches: in the first one we described and solved a minimization problem, and in the second one we used genetic algorithms.

The remainder of the thesis is organized as follows: in Chapter 1 we outline all the background theories and concepts that are used for this work, in Chapter 2 are described some related work in literature, in Chapter 3 we present the implementation of our methods talking about the architecture of the model and the optimization problems, in Chapter 4 we discuss the results that we have obtained and, lastly, in Chapter 5 we conclude our work make some considerations about what we achieved and talk about some improvement that can be done.

1.1 Theory of audio signals

What is sound? Sound is produced by a vibrating object and such vibrations cause air molecules to oscillate, this change in air pressure will create a wave. More specifically, the sound is a mechanical wave whose particularity is the need of a medium in order to expand, most of the time the medium is the air. A sound is created by a source, such as a musical instrument or a person speaking, which will travel through a channel and can then be detected by our ears and perceived as sound.

How can we represent a complex sound? Is possible to visualize every complex sound, such as music and speech, with a *waveform*. Waveforms are fundamental for audio processing because they carry multiple pieces of information, for example, frequency, intensity, and timbre.

As shown in figure 1.1, each waveform can be divided into different categories:

- Periodic waveform, repeat regularly through time. They are divided in:
  - Simple, for example, a single sinewave, is a smooth and repetitive wave that oscillates between a maximum and minimum value;
  - Complex, which is composed of multiple sinewaves.
- Aperiodic waveform, the wave doesn't have periodicity. They are divided in:
  - Continuous, sometimes referred to as noise, are non-repeating waveform that does not have a distinct shape or pattern;

– Transient, for example, pulse waves, are waveforms that consist of brief, high-amplitude pulses, often with a constant low-amplitude "off" state in between.

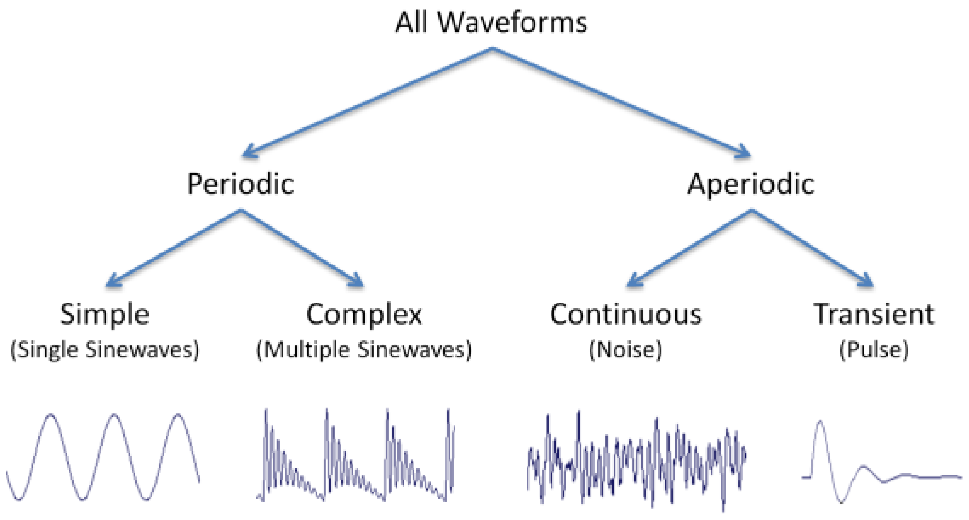

Figure 1.1: Different types of waveform.

1.1.1 Digital vs analog signal

An audio signal is a representation of the sound it encodes all the information needed to be reproduced, is possible to plot an audio signal in a 2-dimensional plane where on the x-axis we represent the time and on the y-axis we represent the amplitude. In real life the signals are analog but in order to use them in our computer, for example, we need to convert them to a digital signal. How we can do that?

In analog signals, we have for time and amplitude continuous values, but when we have to make the signal digital we have a sequence of discrete values and the data point can only take a finite number of values. This analog-to-digital conversion (ADC), also called pulse-code modulation, is composed of two steps:

- *Sampling*, we want to sample our data points in a specific value in time, the x-axis, based on a period T. We can define the sample rate $s_r = 1/T$ as the average number of samples for each second, usually the sample rate is measured in Hertz. Is simple to imagine that if we have a lower sample rate we will be more prone to have the so called sample error because we are going to sample a fewer number of points.

- *Quantization*, the quantization can be seen as sampling our data points in the y-axis, the amplitude of the signal. We choose a fixed number of amplitudes and each sample will be indexed with the nearest value. The number of values is usually referred as resolution and is determined by a number of bits, for example, a CD has a 16 bits resolution which is $2^{16} = 65536$ values. Just like the sampling error, we can also create a quantization error if we choose a lower resolution.

In figure 1.2 is represented an example of conversion from an analog signal to a digital signal, is shown how the operation of sampling and quantization is performed.

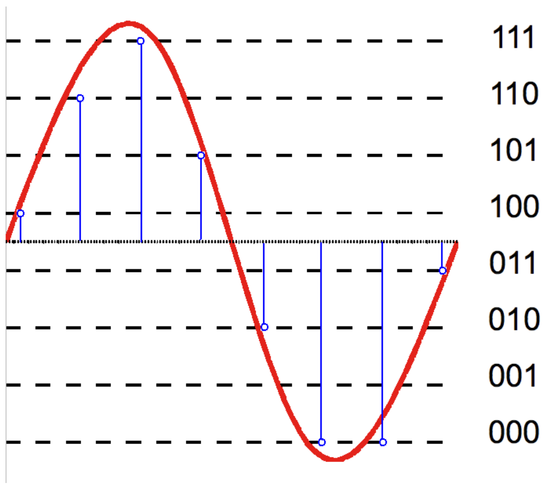

Figure 1.2: Example of sampling and 3-bit resolution.

1.1.2 Fourier Transform

The Fourier Transform (FT) [5] is a mathematical technique that can be used to decompose a complex signal, such as a sound or an image, into its individual frequency components. We need to use this transformation in order to move from the time domain of a signal into its frequency domain, figure 1.3.

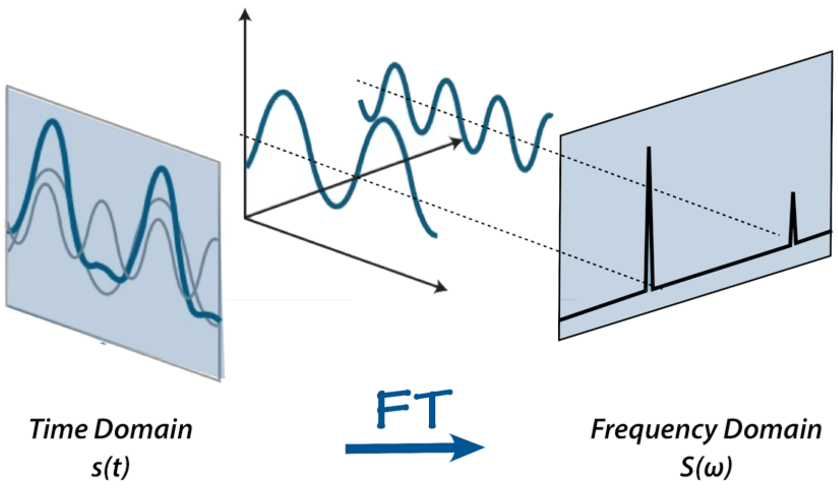

Figure 1.3: Time domain to frequency domain sketch.

The FT work with the intuition of comparing the original signal with sinusoids of various frequencies. The process can be described as follows:

- The original signal is multiplied by sinusoids of particular frequencies. For each of those frequencies, we calculate the magnitude and phase.
- If the resulting magnitude of the product is high, it indicates that there is a high similarity between the original signal and the sinusoid of that particular frequency.

The process of multiplying the original signal with sinusoids of different frequencies, and comparing the resulting magnitudes, allows us to identify which frequencies are present in the original signal. Now a waveform that was represented in a time-amplitude domain is represented in a frequency-intensity domain.

Complex number for signal processing

Is useful to introduce complex numbers because they are used in the Fourier Transform. They provide a convenient mathematical representation of the phase and amplitude of a signal.

A complex number can be divided into two parts: the real part of the number and the imaginary part. Complex numbers allow for the representation of both amplitude and phase information *in a single mathematical object*, making it easier to manipulate and analyze signals in the frequency domain.

Is possible to represent the complex number $z = x + iy$ in the Cartesian plane where the magnitude of the signal is represented by the absolute value of z:

$$|z| = \sqrt{x^2 + (iy)^2} \tag{1.1}$$

and the phase is represented by the angular coordinate denoted as

$$\phi = \arctan\left(\frac{y}{x}\right) \tag{1.2}$$

The point z can be represented in Cartesian coordinate by the tuple $P(Re, Im)$, but also another convenient way to represent the wave is to use its Polar coordinates which are given by the point $P(|z|\cos\phi, |z|\sin\phi)$.

Now, after applying some mathematical computation, the following equation for the point z is obtained:

$$z = |z| \cdot (\cos(\phi) + i\sin(\phi)) \tag{1.3}$$

which is really useful because we can now introduce the Euler formula:

$$e^{i\phi} = \cos(\phi) + i\sin(\phi) \tag{1.4}$$

in order to write it in a much more compact way the value of z as:

$$z = |z| \cdot e^{i\phi} \tag{1.5}$$

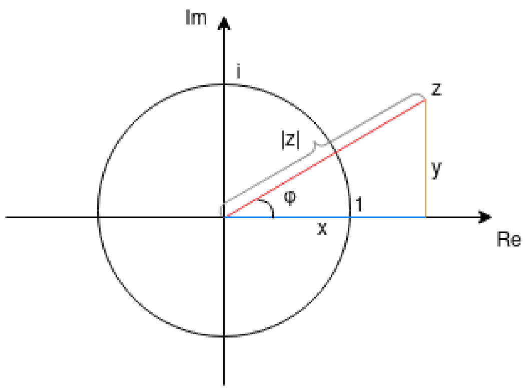

Figure 1.4: Complex number in Cartesian coordinate.

Continuous and Discrete Fourier Transform

To be more precise the Fourier Transform is defined as a mathematical transformation from $\mathbb{R} \rightarrow \mathbb{C}$. In the case of a continuous-time Fourier Transform (CTFT) the transformation formula is the following:

$$\hat{g}(f) = \int_{-\infty}^{\infty} g(t)e^{-2\pi ift}dt = \int_{-\infty}^{\infty} g(t)\cos(-2\pi ft)dt + i\int_{-\infty}^{\infty} g(t)\sin(-2\pi ft)dt \quad (1.6)$$

Where $g(t)$ is our continuous signal function in domain time t and $\hat{g}(f)$ is the frequency-domain representation expressed as a complex number. Thanks to the latter described Euler's formula in equation 1.4 we can divide the integral in two, one for the real part and one for the imaginary part.

But in reality, as we said, we are dealing with discrete digital signals so we are more interested in the discrete version of the FT called Discrete Fourier Transform (DFT). Instead of having a continuous signal $g(t)$ we will have an array of samples $x(n)$, in order to make use of the DFT we have to deal with two issues:

- Because we can not integrate through time we have to fix a certain number of samples N, so we need to map the continuous time domain and the discrete-time domain. The mapping t_n is given by multiplying a discrete value $n = 0, 1, 2, ...$ by the period T.

$$t_n = nT = \frac{n}{s_r} \quad (1.7)$$

- Instead of considering all the possible real values for frequencies, we choose a discrete number of them. Usually, *the number of frequencies considered is equal to the number of samples* in order to make the transformation invertible.

Now we can write the discrete formulation of the Fourier transform as:

$$x(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-i2\pi n \frac{k}{N}} \text{ for } k = 0, 1, 2, ..., N-1 \tag{1.8}$$

The range of all possible frequencies will correspond to $F(k) = \dfrac{k}{NT} = \dfrac{k \cdot s_r}{N}$, which ensures that the maximum number of frequencies is equal to the sample rate.

One last thing is important to say if we plot the magnitudes of the DFT frequencies we can observe that the plot is redundant, in particular, the values are mirrored. They are mirrored when $k = N/2$ so we can take into consideration only half of them, in particular up to the frequency $s_r/2$. The frequency $f = \dfrac{s_r}{2}$ is called Nyquist frequency [6] and represents the highest frequency that can be accurately reconstructed from a given sampling rate without error.

Short-Time Fourier Transform

In order to retrieve the spectrograms that we need in our work, we need to introduce another variation of the FT: the so called Short-Time Fourier Transform (STFT). The standard Fourier Transform provides information for the entire signal, but cannot provide time-specific frequency information. In other words, FT tells us the importance of some particular frequencies in the signal, but can not tell where those frequencies are important through time. Furthermore, our signal is aperiodic, applying the STFT on a small portion of time helps us cheat on the assumption that the signal is periodic.

The STFT is calculated by dividing the signal into small, overlapping segments, and then applying the Fourier Transform to each of them individually. The result is a set of frequency spectra and by examining the content of each segment, the STFT provides information about how the relative frequencies change over time.

The STFT can be represented mathematically as:

$$S(m,k) = \sum_{n=0}^{N-1} x(n + mH) \cdot w(n) \cdot e^{-i2\pi n \frac{k}{N}} \tag{1.9}$$

Where N now corresponds to the *frame size* which is the number of samples that we are considering for the time frame, H is the hop size from one frame to another, and $w(n)$ is a windowing function, the most common functions are the Hamming, Hanning, and Blackman window. The value k is the proxy for frequencies as described in equation 1.8 and the value of m is the proxy to identify the time frame that we are going to consider.

Now instead of having a feature vector, with STFT, we obtain a $F \times T$ matrix where F denotes the frequency bins and is equal to half of the frame size N, for the Nyquist theorem. On the other hand, the value T is equal to the number of frames that we analyze in the waveform.

The choice of the window function, the frame size, and the hop size of the segments will affect the resolution of the spectrogram. In figure 1.5 is represented a little schema of the STFT process.

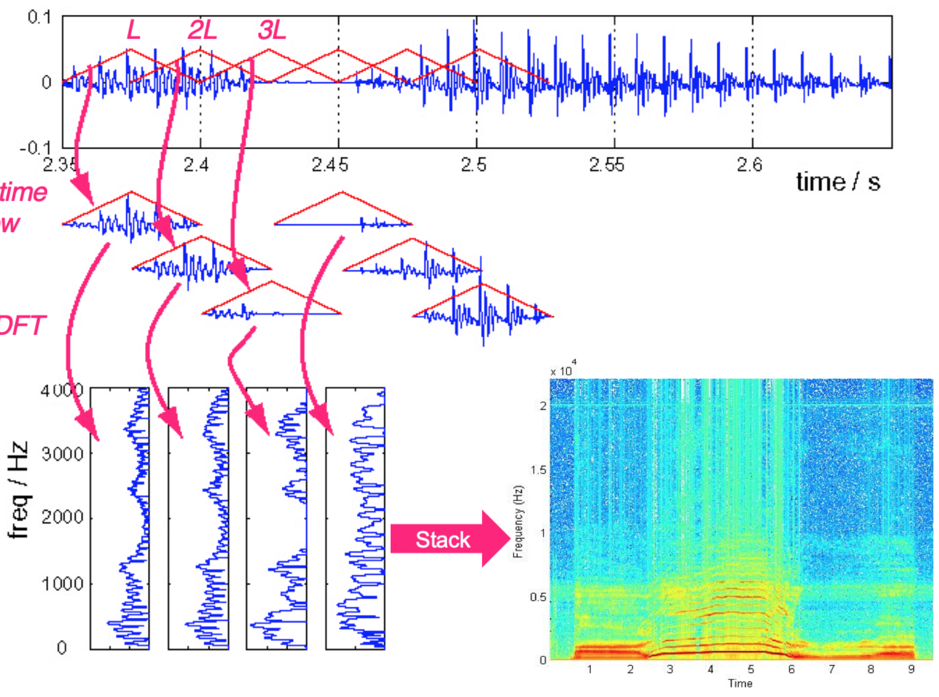

Figure 1.5: Schema of the STFT process.

1.1.3 Spectrogram

Thanks to the Short-Time Fourier Transform we have found a way to transform our audio signal in a matrix of *complex value* $X \in \mathbb{C}^{F \times T}$. From this matrix, we are particularly interested in plotting the magnitude of the frequencies which carry useful information and are much more interpretable than the phases which look almost random. When we plot a spectrogram we will have on the y-axis the frequency bins (Hz) and on the x-axis the time frames, each cell of the matrix will have the corresponding frequency intensity.

There exist different types of spectrograms, each one with its own meaning and characteristics, in our work we have decided to work with the log-power spectrogram (LPS). We can obtain it by the power spectrogram (PS), which in turn is obtained in the following way $|X|^2$, where we remind that $|X|$ represents the magnitude of our signal. Now applying a logarithmic transformation we can retrieve the log-power spectrogram. In particular, we are interested in the following logarithm $10 * \log_{10}|X|^2$, this logarithmic transformation rescales our values in the decibel scale (dB).

The LPS will be the protagonist of our work, as shown in figure 1.6 the spectrogram carries a lot of information about the signal frequencies. Our work will aim to remove as much as possible information from it and then, in a second moment, reconstruct the signal.

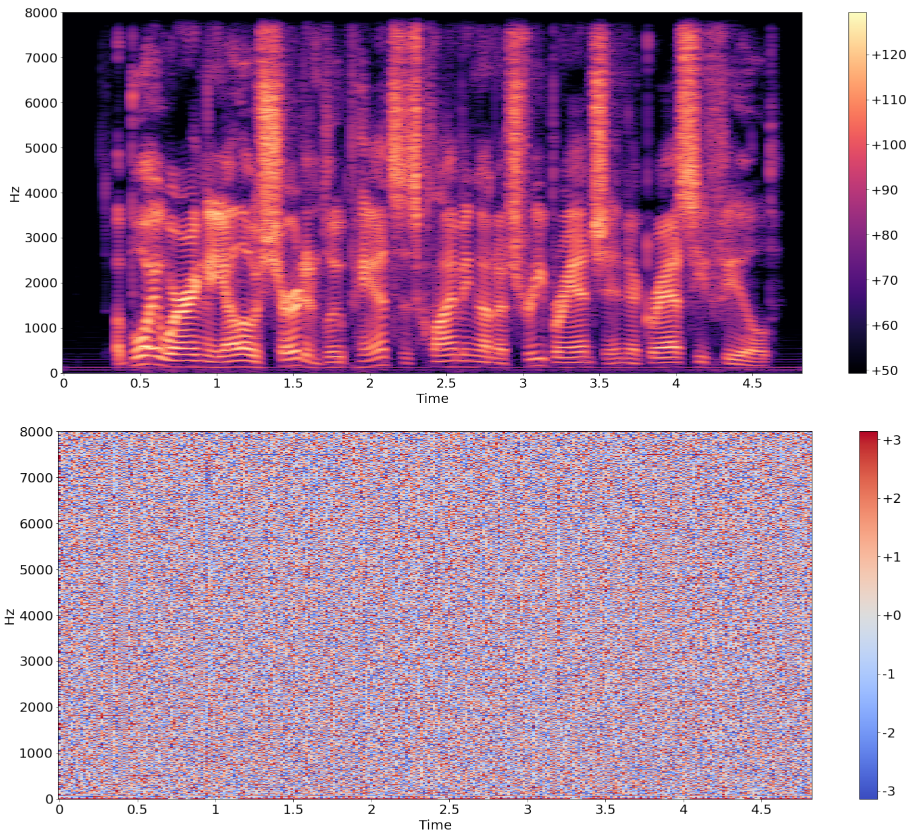

Figure 1.6: Top: Log-power spectrogram of an audio signal. Bottom: phase of the same signal.

1.2 Neural Networks

Neural networks are a type of machine learning model that has been inspired by the structure and functionality of the human brain. They consist of interconnected nodes, called neurons, that are connected by edges, called weights. Each neuron performs a simple computation on the input it receives, then the produced output will become the input for the next layer. This architecture allows neural networks to learn complex patterns and relationships from the data.

Each type of network has its own characteristics and is suited to different types of problems, for example, we can describe:

- **Feedforward neural networks** are the simplest type of neural network, where the information flows in one direction, from the input layer to the output layer. They have been used for tasks such as image classification, speech recognition, and natural language processing.

- **Feedback loop or Recurrent neural networks** are a type of neural network that have feedback connections, allowing information to flow in a cyclic manner. They are well suited for tasks such as speech recognition, natural language processing, and time series prediction.

- **Fully connected**, all neurons in one layer are connected with all neurons on the next layer.
- **Sparsely connected**, only a subset of neurons are connected to the next layer.

The first definition of neural network was made by McCulloch and Pitts in 1943, in their work they have given the definition of the MP neuron [7] which is and highly simplified computational model of a biological neuron.

Given a set of inputs $I \in 0, 1^n$, which values can be 0 or 1, and a set of weights $w \in \mathbb{R}^n$ the neuron "fires" if $\sum_j w_j * I_j$ reaches or exceed a unit's threshold (or "bias") $T \in \mathbb{R}$. The output y of the MP is determined as follows:

$$y = g\left(\sum_j w_j * I_j - T\right) \tag{1.10}$$

where $g(x)$ denotes an activation function such that:

$$g(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \tag{1.11}$$

MP-neuron will be improved years later by Rosenblatt when he introduces the perceptron [8], which has the benefit of weight assignment to variables and can take real values as input.

Nowadays, also thanks to technological advancement, they have been used in a wide range of applications, and they have been particularly successful in tasks that involve large and complex datasets, such as image classification [9], handwriting recognition [10], automatic speech recognition [11], language modeling [12], and machine translation [13], where traditional methods struggle to achieve high accuracy. Deep neural networks (DNNs) are composed of an input layer, multiple hidden layers, and an output layer.

DNNs are capable of learning thanks to the back-propagation algorithm [14] which self-adjusts the weights and bias with respect to an error loss function, denoted by E, that describes the goodness of our model. It is composed of two steps: the forward step and the backward step.

In the forward step, the input of the network is propagated in the forward direction layer after layer. In the backward step, the error measured with the loss function is propagated in the backward direction to update the weights properly. The weights w_{ji} from node i to node j are updated using the gradient descent method through the following formula:

$$w_{ji} = w_{ji} - \eta \frac{\partial E}{\partial w_{ji}} \tag{1.12}$$

where η is a parameter called learning rate and governs the speed at which the network reaches the local minimum. The value for the parameter has to be carefully chosen, if we choose it too small it will converge too slowly and if we choose it too large the gradient starts oscillating. To solve this problem usually is involved a momentum term α that allows the use of large values of η without worrying about the oscillating phenomena, the new variation on the weights is calculated as:

$$\Delta w_{ji}(t+1) = -\eta \frac{\partial E}{\partial w_{ji}} + \alpha \Delta w_{ji}(t) \tag{1.13}$$

1.2.1 Convolutional Neural Network

Convolutional Neural Networks are a particular category of DNNs models, they usually work with images and video to perform a wide range of tasks such as semantic segmentation [15] or object detection [16]. The fundamental operation behind their construction is called convolution, it's a mathematical operation that applies a filter (or kernel) to the image in order to extract certain features from it. Figure 1.7 show how a particular value of the feature mask is calculated through the convolutional operation.

The shape of the extracted feature is determined by different parameters such as the kernel size determines the dimension of the window where the convolution operation is performed, the stride that determines how many units we move for performing the convolution, and padding which adds a fixed value, usually zero, in the contour of the image.

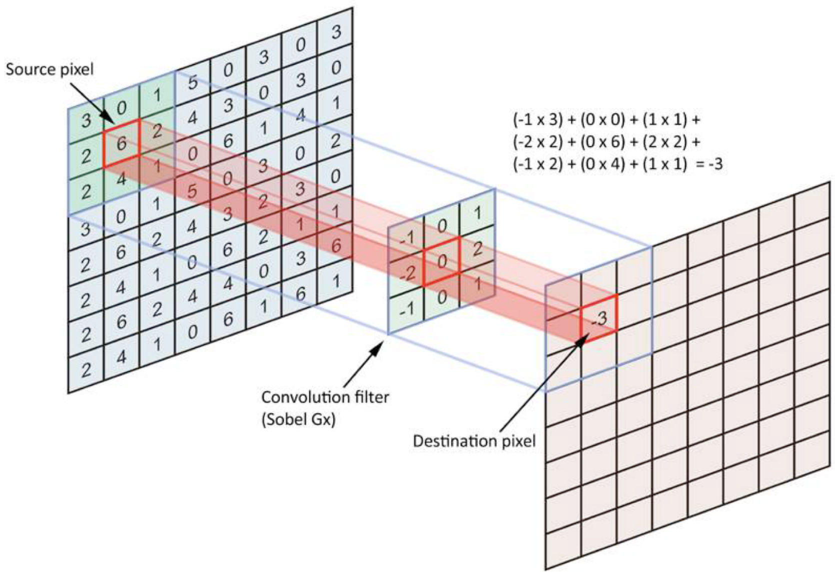
Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

Figure 1.7: Example of convolution operation.

1.3 Interpolation problems

As we remind you, the goal of our work is to reconstruct the values from a sparse spectrogram and such spectrogram can be seen as an intensity map with only one channel. We can refer to the problem of reconstructing those missing values as an interpolation problem.

In the mathematical field, interpolation is a method of estimating or determining a value based on a discrete set of known data points. There exist different types of interpolation, some common algorithms that can be used are:

- *Bilinear interpolation*, this method is an extension of the linear interpolation, it assumes that the function being interpolated is a hyperplane between the known points. We first apply linear interpolation in one direction and then again in the other one.
- *Bicubic interpolation*, this method is similar to bilinear interpolation, but it uses a bicubic polynomial function to estimate the value of the point being interpolated. It's a more complex method than bilinear interpolation, but it produces a better result when the data is more complex and non-linear.
- *Interpolation with convolution*, convolution can be used for interpolation in some cases, convolving our matrix with different kernels can produce different outputs.
- *Interpolation with Neural Networks*, DNNs can also be used to fill the missing values [17, 18], these networks are trained to reconstruct the original data, once they are trained they can fill the missing values by recreating them.

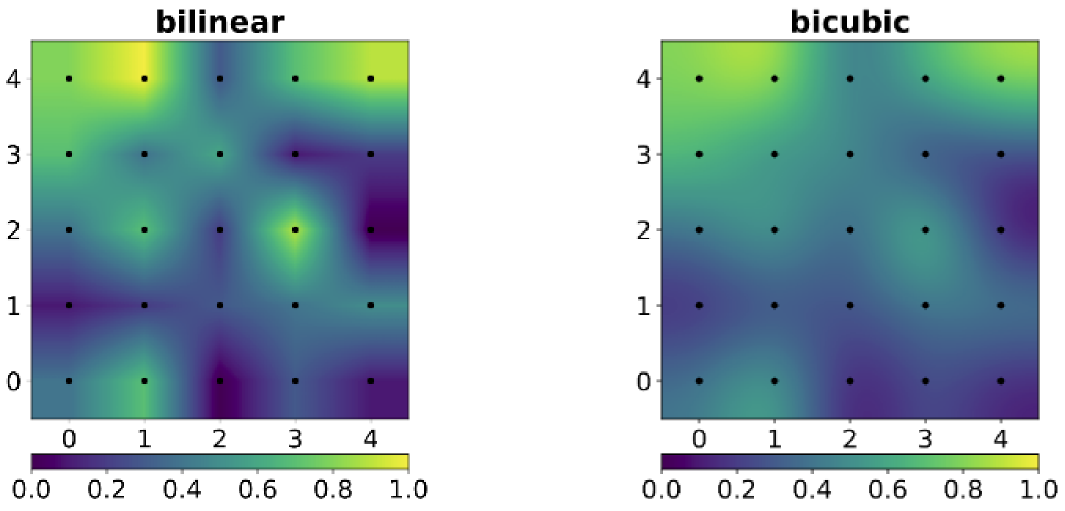

Figure 1.8: Example of bilinear and bicubic interpolation techniques.

Each method has advantages and disadvantages in terms of the quality of the result and speed of computation.

1.3.1 Depth completion problem

The depth completion problem performs the task of predicting dense pixel-wise depth from an extremely sparse map captured from a depth sensor. Those maps obtained by the sensors are often noisy, incomplete, and have holes, the goal is to be able to artificially create those missing values in order to have a dense map as output.

It has a variety of applications in the computer vision field, such as autonomous driving [19], augmented reality [20], and 3D scene reconstruction [21]. The depth map is a representation of the distance between the sensor and the objects in a scene, in our case we are working on spectrograms that don't have real objects and are not retrieved by a sensor too.

Nevertheless, we are interested in more than the what but how the missing values are created, we assume that the spectrogram yields the same characteristics of a depth map. Neural networks, particularly deep convolutional neural networks (CNNs), have been widely used in recent years to tackle the depth completion task. The neural network based methods can handle large missing regions, noise, and occlusions, and provide a dense depth map as output.

1.4 Optimization problems

An optimization problem is a problem in which the goal is to find the best solution among a set of possible solutions. Typically we want to maximize or minimize a certain objective function, which is a mathematical function that describes the goodness of the solution. There exist various types of optimization problems such as linear programming, non-linear programming, and genetic algorithms.

We will introduce a brief theory behind the optimization problems because we need them when we are going to create the best possible binary mask which will maximize the reconstruction of the image. In the following section, we will give an introduction about the two types of optimization problems that we have decided to focus on, later on, we will see how we have applied the theory to the practice.

1.4.1 Non-linear programming problems

We can describe the general non-linear programming problem as [22, 23, 24]:

$$\min_{x \in X} f(x) \tag{1.14}$$

Where the feasible set $X \in \mathbb{R}^n$ is explicitly defined through a finite number of equalities and/or inequalities:

$$X : \begin{cases} h_j(x) = 0, & j = 1, ..., p, \\ g_i(x) \leq 0, & i + 1, ..., m. \end{cases} \tag{1.15}$$

Where in case every h_j are linear and every g_i are convex then for some mathematical proposition the function $f(x)$ is convex.

Now given the described minimization problem we can introduce the following new variables $\lambda_0 \in \mathbb{R}, \lambda \in \mathbb{R}^p, \mu \in \mathbb{R}^m$ and define the Lagrangian function $\mathcal{L}(x, \lambda_0, \lambda, \mu)$ as:

$$\mathcal{L}(x, \lambda_0, \lambda, \mu) = \lambda_0 f(x) + h_j(x)^T \lambda + g_i(x)^T \mu$$

Now in order to define the Karush-Kuhn-Tucker (KKT) optimality conditions for the minimization problem we need to verify one of the constraint qualification conditions, which ensures that $\lambda_0^* \neq 0$.

Hopefully, in our work, we have only one equality constraint which describes our will to have a fixed ratio of compression, so we can verify one of the most used constraint qualification conditions called Linear Independency Constraint Qualification (LICQ):

Given the problem 1.14 and the point $x^* \in X$, we define the set $I(x^*)$ of the active constraints at x^* as the set of indices of the constraints which satisfy:

$$I(x^*) = \{j : h_j(x^*) = 0, j = 1, ..., p\} \cup \{i : g_i(x^*) = 0, i = 1, ..., m\}.$$

The LICQ states that in case the gradients of the vectors of the active constraints are linearly independent, then $\lambda_0^* = 0$, in our case having a single equality constraint the condition is satisfied.

Using the latter constraint qualification conditions we can define the KKT:

Given the problem 1.14, let $f : \mathbb{R}^n \to \mathbb{R}, h : \mathbb{R}^n \to \mathbb{R}^p andg : \mathbb{R}^n \to \mathbb{R}^m$, with $f(x), h(x)$ and $g(x)$ continuously differentiable in an open set containing X. Let $f(x)$ be convex on X, $g_i(x)$ be convex on $X, i = 1, ..., m$, and let $h_j(x)$ be linear (affine) $j = 1, ..., p$. If there exist vectors $\lambda^* \in \mathbb{R}^p$ and $\mu^* \in \mathbb{R}^m$ such that the next conditions at x^* hold

$$\begin{cases} \nabla f(x^*) + \sum_{j=1}^{p} \nabla h_j(x^*) + \sum_{i=1}^{m} \nabla g_i(x^*) = 0, & \\ h_j(x^*) = 0, & j = 1, ..., p, \\ g_i(x^*) \leq 0, & i = 1, ..., m, \\ \mu_i^* g_j(x^*) = 0, & i = 1, ..., m, \\ \mu^* \geq 0, & \end{cases}$$

then x^* is a local (and global) minimum of 1.14. In particular if $f(x)$ is also strictly convex on X, then x^* is the unique local (global) minimum of 1.14.

1.4.2 Genetic algorithms

Genetic algorithms (GA) are also widely used in optimization problems, they perform a search heuristic inspired by Charles Darwin's theory of natural evolution [25, 26]. The idea behind this is to start from an initial population, made of individual, and perform a process of natural selection where in the end we obtain the best individual that minimizes (or maximize) the objective function (or fitness function). The steps of the genetic algorithm can be divided into five points:

- **Initialization**: in this phase, the initial population is generated, it contains individuals which are generated randomly from the solution space. The population size depends on the nature of the problem.
- **Selection**: in the selection phase we want to select the best individuals who are responsible to pass on to the next generation their genes. The selection is made by a fitness score which evaluates the population, then two individuals are chosen, and those parents will reproduce in order to make a child that will inherit their genes.
- **Crossover**: This is one of the operations performed during the reproduction step. The crossover operator split the two parents and then combines different genetic information together in order to generate a new candidate.
- **Mutation** The mutation step is performed to maintain diversity within the population and prevent premature convergence. With a random probability p the genes of the child are perturbed.
- **Termination** The genetic algorithm is repeated until a termination condition has been reached, for example, we can decide to stop when we have reached a fixed number of iterations or the highest ranking solution's fitness does not change anymore.

Chapter 2

Related works

In this chapter, we are going to describe some works in literature that have some connection to our work.

2.1 Depth completion

As we described in section 1.3.1 the depth completion problem performs the task of predicting dense pixel-wise depth from an extremely sparse map captured from a depth sensor.

In the last decades, deep neural networks have shown great performance in the tasks, the work of Hu et al. [27] classified the different methodologies that can be divided into two categories: *unguided methods*, which aim to complete the sparse depth map with only DNN models, an example of input feature is represented in figure 2.1, and *RGB guided methods*, which also use the RGB information in order to reconstruct the sparse depth map.

In particular, from the first category we took into consideration the sparsity-aware CNNs that use a binary mask to differentiate between valid and missing elements during the completion convolution operation, among them [28, 29, 30] we chose the work of Uhrig et al. [28] which was the most compatible with our assumptions. Other depth completion methods, instead, replace the binary validity mask with a continuous confidence map [31, 32] for a better completion performance, but in this way, we can't store only the values that we are interested to keep.

The RGB guided depth completion does not fit at all with our task, because they work along with the RGB image that doesn't exist in our case, for the sake of completeness is right to mention some methods in literature: early fusion methods [33, 34, 35], they directly concatenate the sparse depth map and the RGB image before feeding them to the DNN model; late fusion models [36, 37], usually consist of dual encoders or two sub-networks; explicit 3D representation models [38, 39], which explicitly learn 3D representations; Residual depth models (RDM) [40, 41], learn a coarse depth map and a residual depth map; Spatial Propagation Network (SPN) based [42, 43], they learn the affinity matrix and then use it for depth refinement.

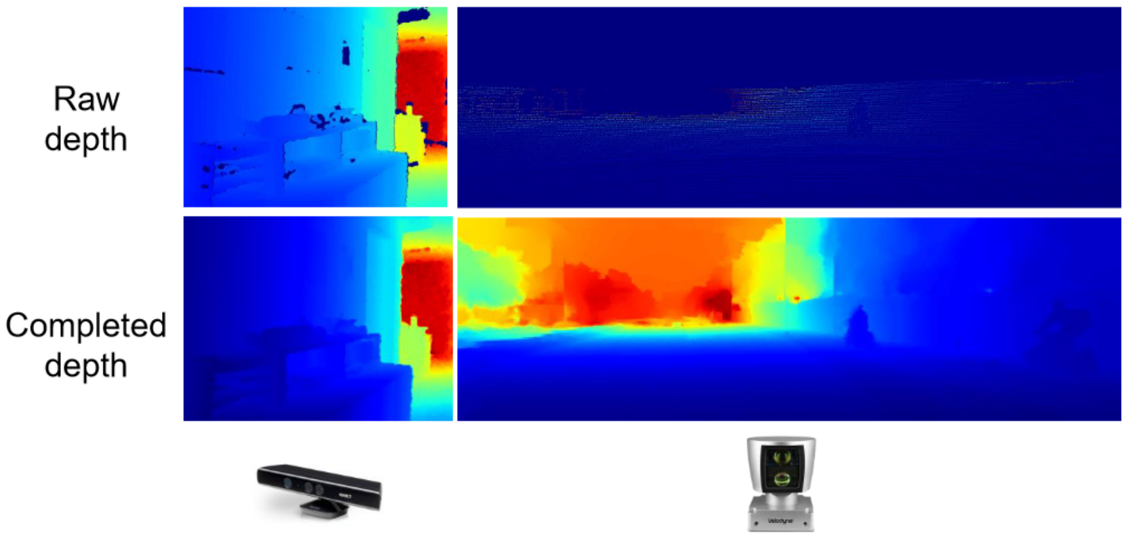

Figure 2.1: Example of depth map for a depth completion task taken by the paper of Hu et al. [27]. In the figure is represented an example of sparse input and dense output from data obserbed by a kinect (left) and a LiDAR (right).

2.2 Speech enhancement

Speech enhancement was always a challenging task to perform, the goal of the task is to extract the speech signal from a mixture of speech and background of ambient noise in order to improve the intelligibility and quality of speech [44].

It depends on the objective of the problem but speech enhancement can be achieved by separating the clean speech from the background or trying to conceal the noise in the audio in order to improve intelligibility or quality. Usually, this type of task is involved in speech compression, recognition, and authentication. Enhancement can also be used when we are communicating through a noisy signal or we want a clean signal that was registered in systems located in noisy environments[45].

In literature there exist different classical enhancement methods, for example, Boll et al. [46] and Berouti et al. [47] proposed a noise reduction technique that estimates the magnitude frequency spectrum of the underlying clean speech by subtracting the noise magnitude spectrum from the noisy speech spectrum.

But with the advent of neural networks, more machine learning-based techniques arose, addressing the task in a supervised manner by training a DNN model to estimate the clean speech signal from the noisy input.

Starting from more or less complex network architecture, each one of them has its unique characteristics and assumptions. For example, the work of Fu et al. [48] proposed a convolutional neural network estimate of the clean real and imaginary spectrogram from noisy ones. In particular, their implementation of a multi-metrics loss function that involved both log-power spectrogram and the real and imaginary one was very performing.

A more complex architecture was proposed by Pascual et al. [49] with their Speech

Enhancement Generative Adversarial Network (SEGAN) which involves the use of a generative adversarial network (GAN) [50] applied to the waveform level of the signal. The idea behind involving the GAN in order to retrieve clean speech is that they play a game of counterbalance between the generator and the discriminator in order to get rid of the noise, it will also help the generalization of the network for different scenarios.

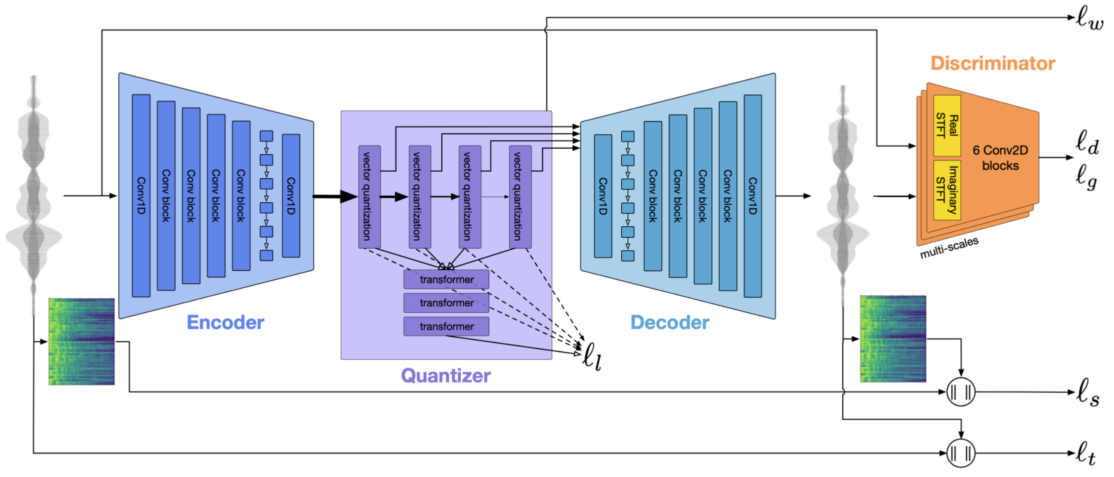

Figure 2.2: Network architecture of the autoencoder proposed by Defossez et al.

2.3 Data compression

Data compression is a technique that reduces the quantity of data used to represent something without excessively losing too much in quality. It can be used in different contexts and applications, for example, it is performed when we store information on permanent memory or for example when we transmit something over a channel. We can divide the data compression algorithms into two categories [51]: lossless compression techniques and lossy data compression techniques.

The first methods ensure that when the data is decompressed we will retrieve the exact same data before the compression. On the other hand, the second technique does not preserve all of the information when the data is compressed, usually, the irrelevant data are removed. In particular, in the audio field, some lossless handcrafted compression techniques are WAV, FLAC, and ALAC; and also lossy techniques: MP3, MP4, and OGG.

Although in recent years, with the advancement of technology, deep neural networks have begun to be used as a data-driven approach to compression. Adopting this approach to compress the data eliminates the need for manual coding and assumptions, allowing the machine learning model to autonomously learn the task.

Most of the state-of-the-art networks use an autoencoder [52] to perform the task, they are usually implemented as a feed-forward network that has a particular hourglass shape. The main idea behind the model is to try to encode the input into a compressed meaningful representation and decode it back reconstructing the input as similar as possible to the original one. In particular, with speech audio files, the compression is usually performed by reducing the bitrate of the signal [53, 54]. For example, Defossez et al. [55] implemented an autoencoder, represented in figure 2.2, that is trained with six different loss functions they can achieve a state-of-the-art real-time neural audio compression model, producing high-fidelity audio samples across a range of sample rates and bandwidth.

Chapter 3

Audio Compression with Sparse CNNs

In this chapter, we describe in a more practical way all the instruments that have been used in our work.

We will discuss the implementation of the network that we have chosen describing how it works and which parameter we have used. Then will be introduced all the steps necessary to transform the audio signal in such a way that we can use them as input for the model. Lastly, we will focus on describing more precisely the problem of the binary mask and how we have tried to solve the optimization problem for finding the best mask for each sample.

3.1 Decompression step

For the decompression step, we rely on a machine learning model in order to perform the task. In figure 3.1 there is an illustration of the pipeline of the operations that we perform. The neural network takes as input the sparse spectrogram and the corresponding binary mask which describes where the information is present or not, and provides as output a dense spectrogram. Then, in order to reconstruct the waveform, we are going to perform the Inverse Short-Time Fourier transform (inverse STFT) using the original phase of the signal and the model output. With the latter waveform, we can now compare it with the original one using audio metrics such as PESQ and STOI to evaluate the quality of the reconstruction.

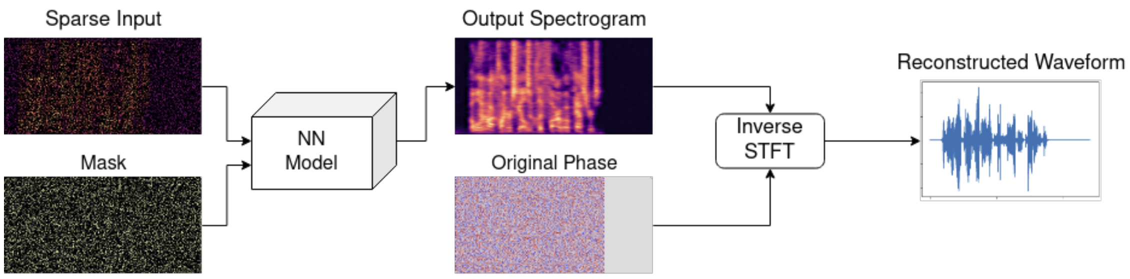

Figure 3.1: Pipeline of the decompression step.

3.1.1 Sparsity invariant CNNs

Our work is based on the sparsity invariant CNNs proposed by Uhrig et al. [28], the authors proposed a new type of convolutional neural network that is robust to sparsity, which means that it can handle images with missing or corrupted pixels and reconstruct it with a certain level of confidence. The neural network model is used to perform depth completion tasks, it works with two input features: a sparse image and a sparse binary mask that describe the presence or not of information in the image. The paper introduced the concept of sparse convolutions, which is an extension of a normal convolution. Let f be a mapping function from the sparse input $\mathbf{x} = \{x_{u,v}\} \in \mathcal{X}$ to the output $\mathcal{Y}$, we denote as $\mathbf{o} = \{o_{u,v}\}$ is the binary mask where if $o_{u,v} = 1$ the input is observed otherwise we have $o_{u,v} = 0$. A standard convolutional layer in CNN is computed in the following way:

$$f_{u,v}(x) = \sum_{i,j=-k}^{k} x_{u+i,v+j} w_{i,j} + b \tag{3.1}$$

with kernel size 2k + 1, weight w, and bias b. Instead, the sparse convolution operation also uses the binary mask information, the motivation behind this convolutional operation is that the output feature map is invariant to the actual number of observed inputs. The output is computed in the following way:

$$f_{u,v}(x) = \frac{\sum_{i,j=-k}^{k} o_{u+i,v+j} w_{i,j} x_{u+i,v+j} w_{i,j}}{\sum_{i,j=-k}^{k} o_{u+i,v+j} w_{i,j} + \epsilon} + b \tag{3.2}$$

The small epsilon is added in order to avoid division by zero where none of the input is present. The binary mask is propagated to the next layer changing its visibility state, we maintain 0 when no information is observed otherwise we have to put a 1. This type of operation is defined by the max pooling operator:

$$f^{o}_{u,v}(\mathbf{o}) = \max_{i,j=-k...k} o_{u+i,v+j} w_{i,j} \tag{3.3}$$

In our work, we maintain the same implementation of the Sparse convolution block, pictured in figure 3.3, but slightly change the network structure represented in figure

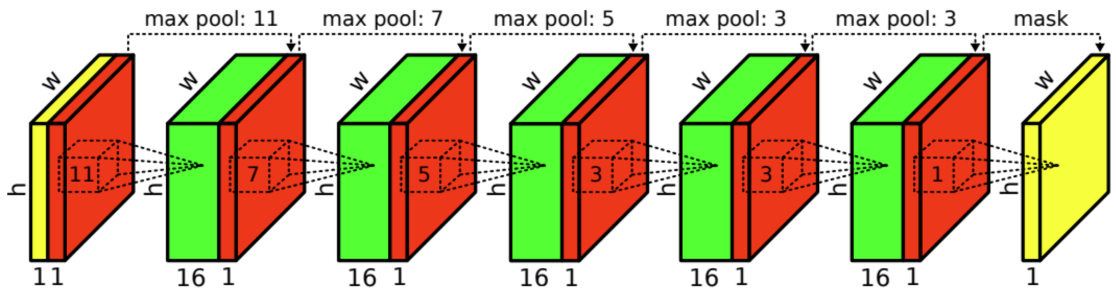

Figure 3.2: Original network architecture of the Sparsity Invariant CNNs (Uhrig et al. 2017).

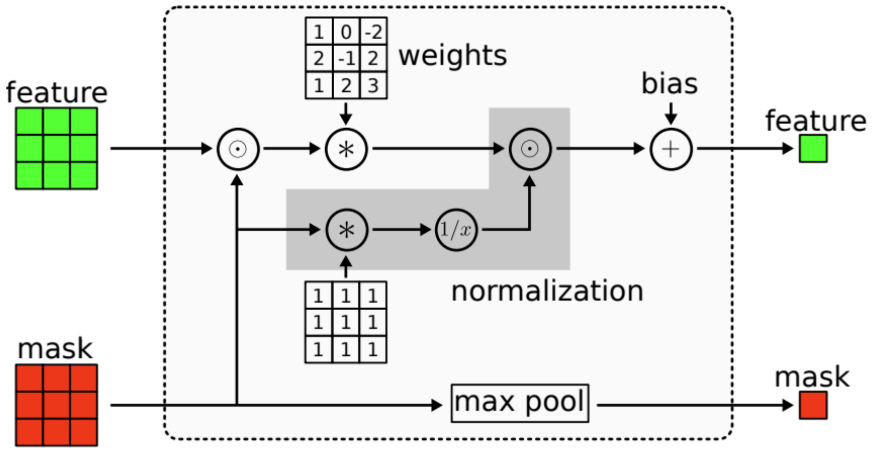

Figure 3.3: Sparse convolutional block (Uhrig et al. 2017).

3.2. In particular, our net is composed of 6 hidden layers with kernels of 7-7-5-5-3-3 and 16 feature maps and the activation function chosen for the neurons was rectified linear unit [56].

3.1.2 Reconstruction pipeline

To summarize, we take a log spectrum and reconstruct the whole signal through the described network.

Now we can apply the inverse STFT using the phases of the ground-truth signal and the dense output and then evaluate the quality metrics that we have chosen. In terms of what we physically store, we also need to store the phases together with the sparse spectrogram in order to reconstruct the waveform. For this reason, we have also tried to use algorithms that reconstruct the signal without the needing of the phase, for example, we tried to use the Griffin-Lim algorithm [57], but the latter result doesn't perform very well as we will see.

3.2 Compression step

The first stage of the compression process involves transforming the waveform signal into its spectrogram and phase representation through the use of the Short-Time Fourier Transform (STFT). Next, the aim is to find the optimally sparse binary mask, denoted by M^*, which minimizes the information loss of the input signal X. This mask must adhere to a sparsity level constraint SL, representing a specified percentage of non-zero pixels.

By multiplying the sparse mask with the full spectrogram, we obtain a sparse representation that can be decompressed at a later stage. The subsequent section outlines the formulation of two different optimization problems, each with its unique advantages and disadvantages.

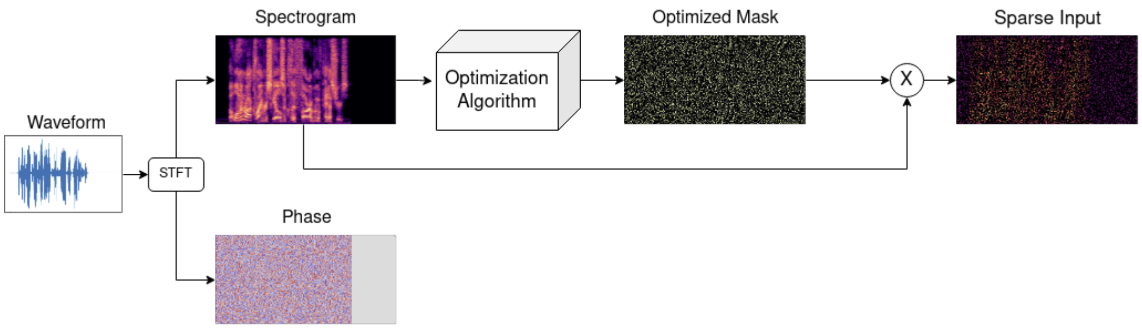

Figure 3.4: Pipeline of the compression step.

3.2.1 Minimization problem

We have seen how is possible to theoretically solve a non-linear programming minimization problem. During our analysis, we tried to impose a constraint minimization problem in order to solve the problem of creating the most suitable ad-hoc mask given a sample. Our idea was to formulate the problem as follows:

$$\begin{aligned} &\min_M f(M) \\ \text{s.t.} \quad &\sum M = SL \\ &M \in \{0,1\}^{n\times m} \end{aligned} \tag{3.4}$$

Where $f(M)$ is the objective function $\{0,1\}^{n\times m} \to \mathbb{R}$ that we want to minimize, subject to the constraint that the mask M is binary and the sum of its values is equal to a fixed value SL. Such value is equal to the number of pixels that we want to keep from the original input in order to make it sparse, for example, if the input has 100 pixels and we want a sparsity level of 90% we will have the value $SL = 100 * (1 - 0.9) = 10$.

The objective function is defined by the ℓ_2 norm squared between the full image and the net output $R(X \cdot M, M)$, which takes as input the dense spectrogram and the mask that we are minimizing. The mathematical definition of the objective function is the following:

$$f(M) = \frac{1}{N}\|R(X \cdot M, M) - X\|_2^2 \tag{3.5}$$

But solving the optimization problem for a discrete value binary mask in NP-Hard. We have to construct a new formulation that approximates the binary mask with a real values mask, in this way the latter mask will be differentiable. The new values of the mask are sampled in the domain of a smooth function that maps values from $\mathbb{R} \to (0, 1)$, in particular, we have chosen the logistic function $\phi_n(x) = \frac{1}{1+e^{-nx}}$, shown in figure 3.5, the parameter n determine the growth rate or steepness of the curve.

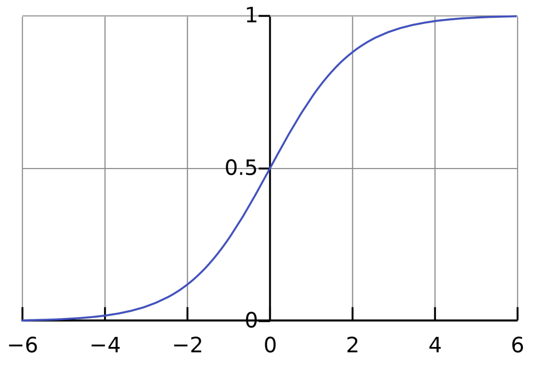

Figure 3.5: Standard logistic function with $n = 1$.

After introducing the new elements, we can rewrite the problem as:

$$\begin{aligned}
&\min_M f(\phi_n(M)) \\
&\text{s.t.} \sum \phi_n(M) = SL \\
&\quad M \in \mathbb{R}^{n \times m} \\
\text{where } &f(\phi_n(M)) = \frac{1}{N}\|R(X \cdot \phi_n(M), \phi_n(M)) - X\|_2^2
\end{aligned} \tag{3.6}$$

Thanks to the Lagrange multiplier methods we can solve the optimization problem using a stochastic gradient descent optimization method for a certain number of steps, using for example AdamW. After a certain number of iterations, or the reaching of the local minimum for the problem, we will have an optimal mask $M^* \in \mathbb{R}^{n \times m}$ which minimizes the objective function.

3.2.2 Genetic algorithm

In contrast with the non-linear programming problem, respecting the sparsity level constraint is easier because every mask of the population is created to respect it. As for the minimization problem, we will describe the real implementation of the genetic algorithms theory. In particular, given a fixed spectrogram, we are going to define how we have implemented each step of the algorithm:

- **Initialization**: We initialize the population with a fixed number of individuals, each of them corresponding to a randomly generated sparse mask with the sparsity level constraint that we have chosen.
- **Selection**: The selection is performed by multiplying every mask of the population for the same spectrogram, the one that we want to sparsify, then, in order to select the best individuals, we have to compute the model output for each of them.

  Given all the outputs of the model, we rank them using two fitness functions: the PESQ metrics applied to the reconstructed waveforms and MSE applied to the spectrograms. Now we can construct the new population that will be partitioned into three parts: partition 1 have a subset of the best possible individuals, partition 2 is generated by reproducing a portion of the best individuals, and the last partition will be composed of new masks which are generated at random.
- **Mutation and Crossover** During the selection two individuals are chosen as parents in order to reproduce and make the child, the mutation and crossover were implemented following this logic: both parent masks have a 1 where the information is present, and 0 when the information is missing. The parents were chosen by their fitness function score, so whether there is or not information is important and we want to pass this knowledge to the child. We sum together the parents' masks and we obtain a new matrix that we will divide by 2, the values of the child mask are: 1 when both parents have the information, 0 when both parents didn't have it, and 0.5 when only one of them have the information. In order to make the constraint respected again, with a random probability, we will set 0 or 1 values until we have the right number of sparsity. In figure 3.6 is represented a little schema of the reproduction between two masks, and in figure 3.7 we have a schema of how the next population is created.
- **Termination** For the termination condition we have chosen to stop the algorithm after a fixed number of iterations.

Although the genetic algorithm will surely respect the constraint condition, does not guarantee to give the best possible mask at the end of its execution. If we could perform more iterations would be possible to find a better mask, but we have to take into consideration the trade-off between the computational time of the algorithm and the performance of the mask during the reconstruction.

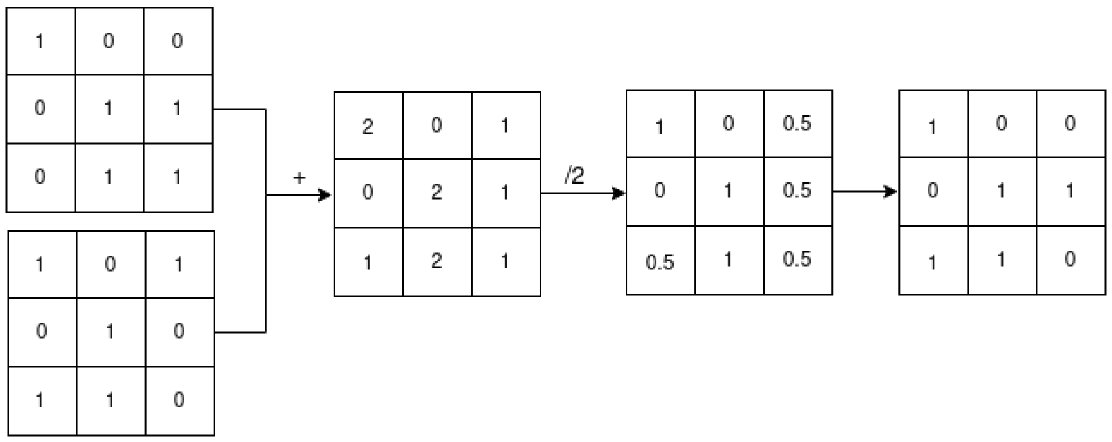

Figure 3.6: Example of reproduction of two individuals.

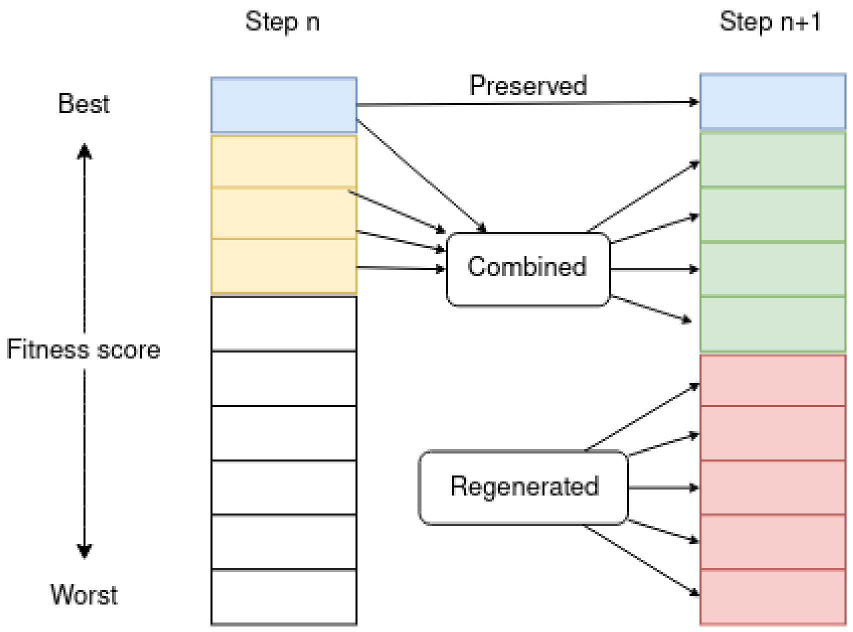

Figure 3.7: Partitioning of the next population for the genetic algorithm. In this experiment, each block is five individuals.

Chapter 4

Experimental results

In this chapter we are going to discuss the results of our work, we start by introducing the dataset that we have used and the environment where the tests were performed. In our work, we have decided to study how the quality of the compression and decompression were influenced using masks with sparsity levels of $80\%, 85\%, 90\%$, and 95% and show how the two optimization algorithms can further improve the quality of the compression, or not. The experiments were performed using python, in particular the PyTorch library, for implementing the model and an ASUS Dual GeForce RTX 3070 8GB OC were used to train it.

4.1 Flickr 8k Audio Caption Corpus

The dataset that we have chosen for our work is an audio corpus version of the Flickr 8k [58]. The original dataset contains approximately 8,000 images captured from the Flickr photo-sharing website, each of which depicts actions involving people or animals. Each image was annotated with a text caption by five different people (non-expert workers or "Turkers") who can choose to complete the task for a small amount of money, resulting in a total of 40,000 captions. For example, the annotations that the turkers made for the input in figure 4.1 of Flickr were the following 5 phrases:

- A woman climbs up a cliff.
- A woman rock climber scales a cliff far above pastures.
- A woman rock-climbing on a cliff.
- A woman rock-climbs in a rural area.
- Woman climbing a cliff in a rural area.

Harwath and Glass [59] took the Flickr 8k and made the Flickr 8k Audio Caption Corpus by asking turkers to record a set of 10 random captions from the image caption dataset. The collection involved 183 unique subjects, that read 218 captions on average.

Figure 4.1: Flickr8k reference image for description.

In our tests, we decided to use 10.000 audio samples. Because we need all waveforms to have the same length, in order to create spectrograms with the same cardinality, we have to make the shortest audio file long as the longest one. This operation can be performed by adding zeros at the end of the shortest samples.

More precisely, the audio signals of the dataset are sampled with a sampling rate of $s_r = 16000$ and the longest file of our data is composed by $\#\text{Sample} = 77276$. Furthermore, if we divide the number of samples by the s_r we will obtain the duration of the audio signal in seconds which is ~ 4.83".

In order to have suitable images for our model we have to extract the spectrograms from the audio files.

A crucial step in order to generate the imaginary spectrograms X is to select the appropriate parameters for the Short-Time Fourier Transform. In this study we used the class `torchaudio.transforms.Spectrogram` from PyTorch [60] for generating them, it is composed of different parameters, the following ones are responsible to decide the size of the images:

- `n_ftt`, it determines the number of frequency bins of the spectrogram. It creates $\lceil \texttt{n\_ftt}/2 \rceil + 1$ bins, for the Nyquist theorem as we remind you. In our work, we have decided to use $\texttt{n\_ftt} = 2048$ for a total of 1025 frequency bins.
- `win_length` defines the size of the window that will move through the waveform, in other words, it identifies the number of samples used to perform the DFT. Usually, the `n_ftt` is chosen as the default value because it guarantees that the STFT will be a perfect reconstruction of the original signal.
- `hop_length` define the number of samples that we skip between STFT windows. In this work, the value is set to $\texttt{hop\_length} = 256$.

Using those parameters, the cardinality of the x-axis, which describes the number of frames, is given by:

$$\#\text{frames} = \frac{\#\text{Sample - } \texttt{win\_length}}{\texttt{hop\_size}} + 1 = \frac{77276 - 2048}{256} + 1 = 294 \qquad (4.1)$$

which after adding some padding we will have a total of 302 time frames, each representing $\frac{\texttt{win\_length}}{s_r} = \frac{2048}{16000} = 0,128$ ms of audio.

From the imaginary spectrogram X we can now retrieve the dense log-power spectrogram by applying the `power_to_db` function of the librosa library [61] to $|X|^2$.

Now, given the dense log-power spectrogram and a binary mask, we can multiply them together in order to create the sparse input. We can now train the model feeding as input the tuple composed by the maskedInput 4.2 and the mask 4.3. The training target Y is the corresponding full spectrogram, an example is shown in figure 4.4.

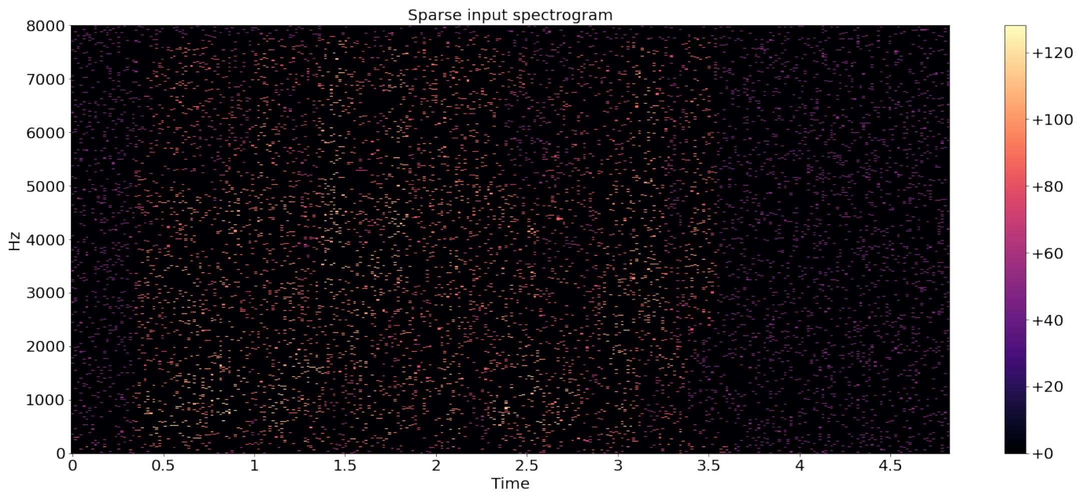

Figure 4.2: Example of sparse input at 95%.

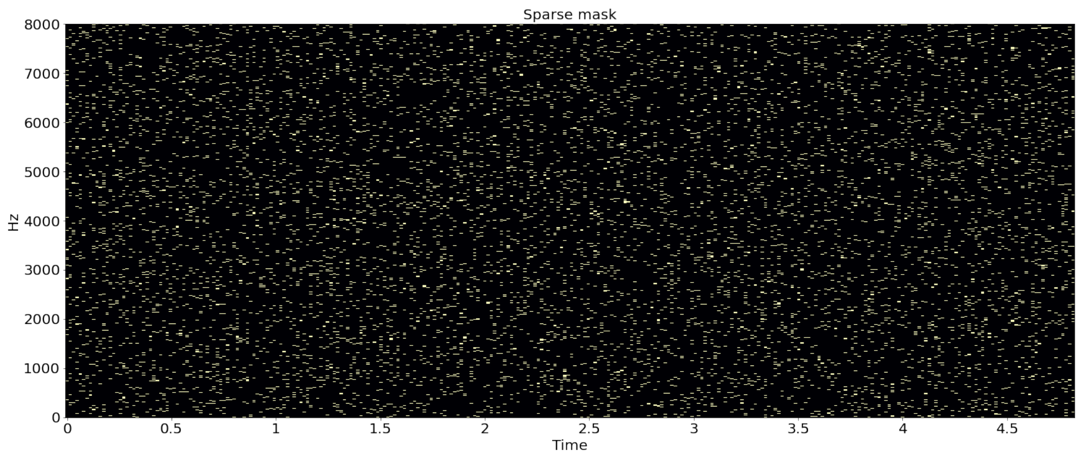

Figure 4.3: Example of sparse mask at 95% for the input in figure 4.2, where black mean no information.

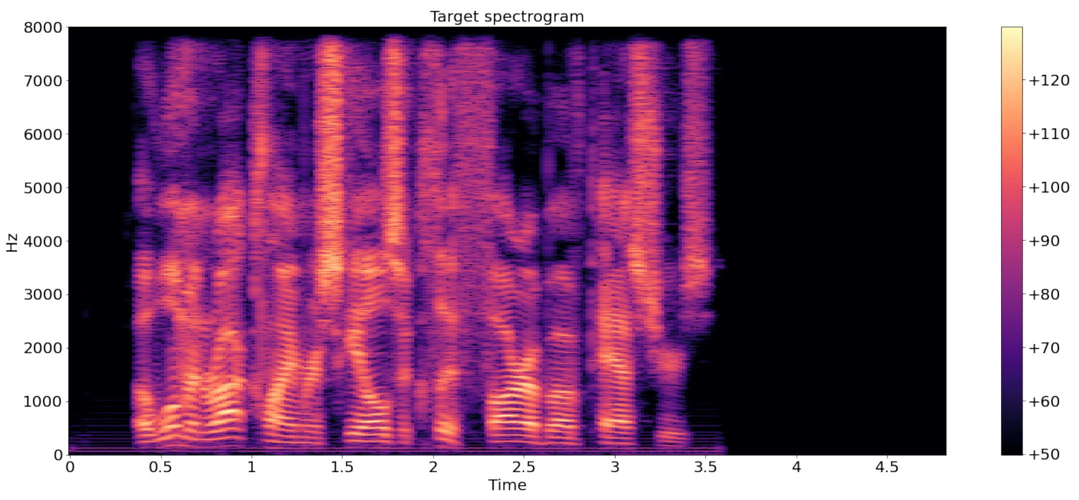

Figure 4.4: Target spectrogram for the input in 4.3.

4.2 Training procedure

In the precedent section 3.1.1 we have described the implementation of the network, in the following one we are going to give more detail about the parameter that we have used for the experiments. Also for the network implementation we have used the PyTorch library.

For the training phase, the 10000 samples were divided into two parts: 9000 were used as the training set and the remaining 1000 for the test set. For the purpose of computing the quality metrics, additional 2500 samples will be used as the validation set.

The training loss function that we have decided to use for the model is the MSE loss between the dense output spectrogram and the groundtruth spectrogram. We chose this particular loss function because it's usually used for depth completion problems, in the following we give its mathematical definition:

$$\text{MSELoss}(X, Y) = \frac{1}{N} \|X, Y\|_2^2 = \frac{1}{N} \left(\sqrt{\sum_{i=1}^{N} (x_i - y_i)^2} \right)^2 \tag{4.2}$$

Each model was trained for 20 epochs and a batch size of 16 samples, as optimizer for the gradient we have used adaptive moment estimation with weight decay, called AdamW [62], with learning rate $l_r = 1e^{-4}$ and weight decay equal to $5 * 1e^{-4}$.

After the training process the model is able to complete sparse inputs, in figure 4.5 is represented the dense output spectrogram of the input in figure 4.2.

4.3 Signal reconstruction

In the following section, we are going to talk about the behavior of the model with the validation set, for the first part we are going to consider sparse spectrograms

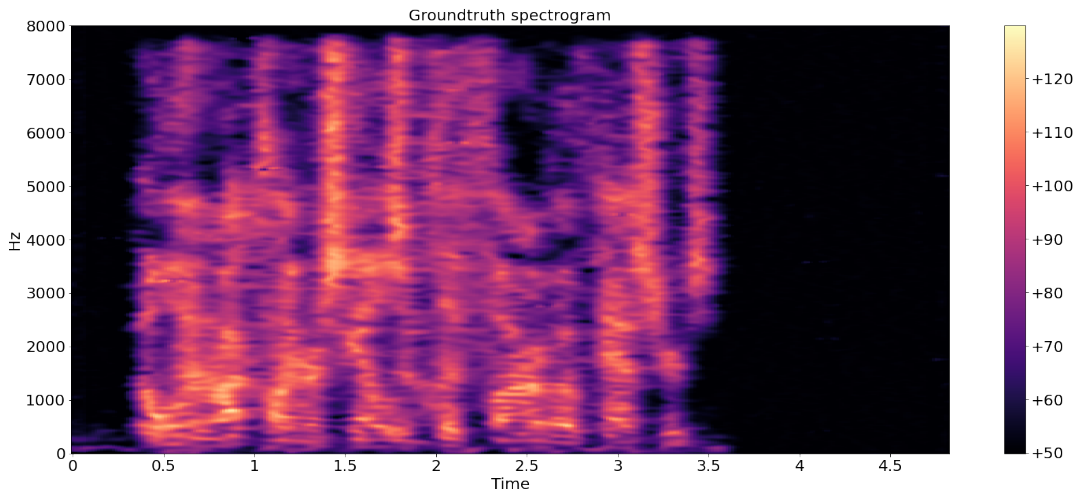

Figure 4.5: Example of output of the model.

created with sparse random masks. But before discussing the results, we have to introduce the metrics that we have used to evaluate how good is our model to perform the reconstruction task. We have identified the following metrics:

- Mean Squared Error (**MSE**), computed between the output spectrogram and true spectrogram. We chose to employ this metric for evaluation because it is consistent with the metric utilized during the training process.
- Perceptual evaluation of speech quality (**PESQ**) [63] is a method for speech quality assessment of telephone networks and codecs which is also widely used to evaluate speech. Returns a score from -0.5 to 4.5, with higher scores indicating better quality. Is performed between the reconstructed waveform and the original one.
- Short-Time Objective Intelligibility (**STOI**) [64], is a metric to evaluate intelligibility of not so much noisy speech, it does not measure the speech quality. It is an algorithm that predicts the intelligibility of speech signals, and it is based on the correlation between the original and the processed speech signals. The output score is between 0 and 1, where 1 indicates perfect intelligibility, and 0 indicates poor intelligibility. Is performed between the reconstructed waveform and the original one.

When we are dealing with audio signals, in particular when we are working with speech audio signals, is possible to use different metrics [65, 66] for assessment of quality. The metrics can be divided into two larger groups: full-reference metrics and non-intrusive metrics.

Full-reference metrics, also known as intrusive or similarity metrics, compare a noisy signal with a clean reference. Worth to mention are POLQA [67], VISQOL [68], DPAM [69], CDPAM [70]. On the other hand, no-reference metrics, also called non-intrusive metrics, don't need a clean reference in order to give a score, some examples are DNSMOS [71], NISQA [72], SQAPP [73].

4.3.1 Reconstruction with all the phases

Now we are going to present the results of our experiments on the four models trained with the different sparsity levels 80%, 85%, 90%, and 95%. We have summarized the results in three different plots, one for each metric, and we have reported the mean and standard error on the validation set. On the x-axis, we have the different sparsity levels that we have considered, and on the y-axis the score of the metric to which we refer.

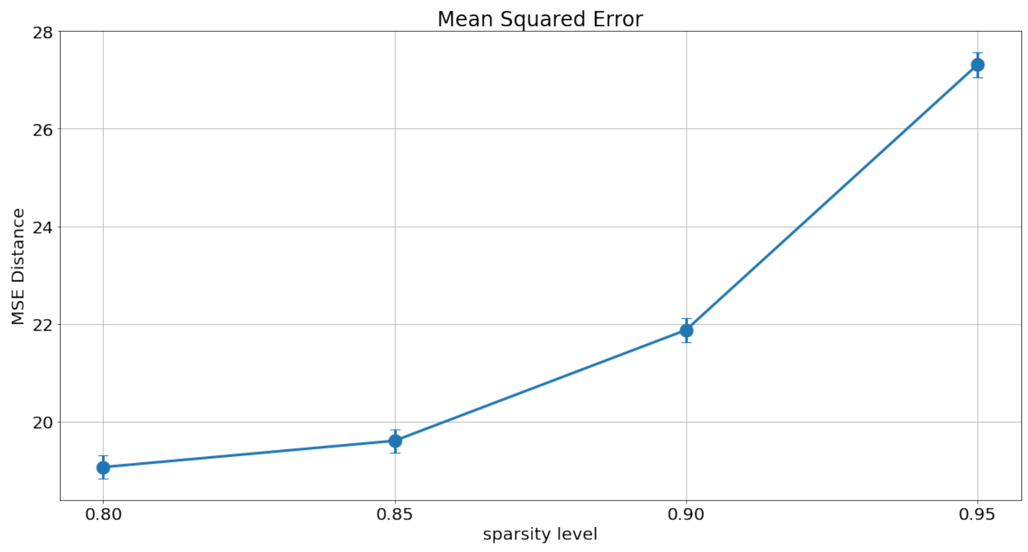

Figure 4.6: Evaluation of the MSE on reconstructed spectrograms.

In the first figure 4.6 we are evaluating the mean squared error, where lower is better, between the output and reference spectrograms. We can notice that with 80% and 85% of sparsity, our models are able to reconstruct the spectrograms with more or less the same error rate. With the increasing sparsity, the model begins to struggle to reconstruct the spectrogram, but we obviously expected it because is more difficult to perform such task. In particular, having less known data available during the training the model will perform a sort of Gaussian blur to minimize the MSE.

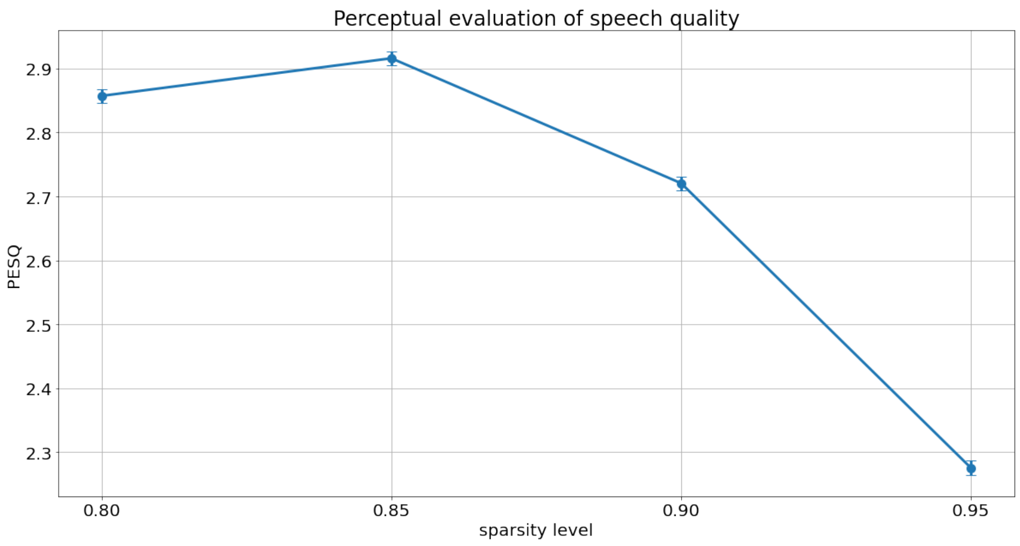

Figure 4.7: The four model compared with PESQ.

As described before the PESQ metric, where higher is better, evaluates the audio quality. Summarized in figure 4.7 is possible to observe a similar trend described with the previous metric. With a low level of sparsity, the model is able to reconstruct the signal with an average score considered audible with a little noise. Instead with the increasing of missing information, the error in the reconstruction starts to degenerate.

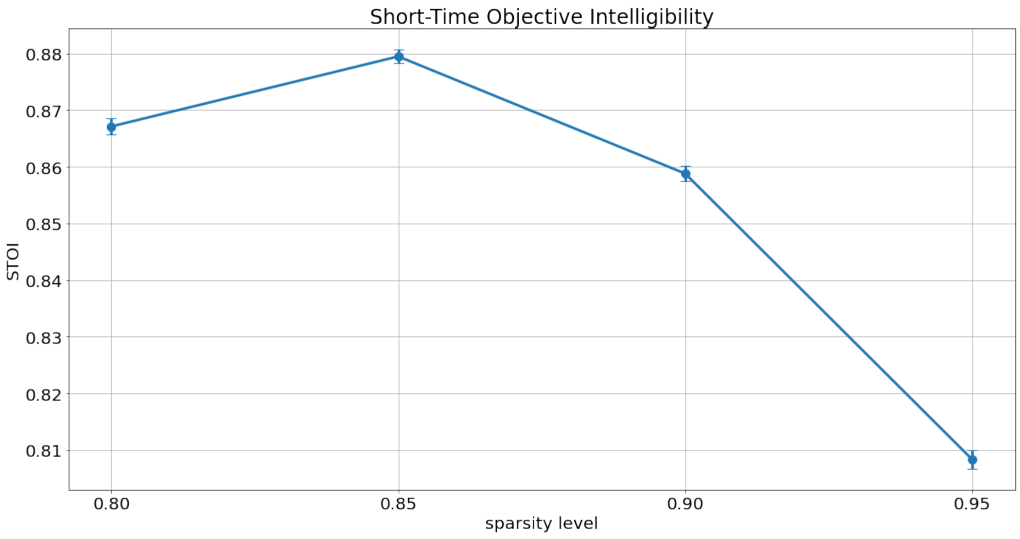

Figure 4.8: The four model compared with STOI.

The last metric that we have considered is summarized in figure 4.8, STOI metric represents the intelligibility of the reconstructed waveforms. Although it seems to follow the two precedent evaluation, the worst intelligibility is given by the model with 95% of sparsity with a score of 0.81 (higher is better). Despite a slight decrease in quality, the intelligibility of the phrases is maintained to a considerable extent, making the score still acceptable.

4.3.2 Reconstruction with masked phases

During our analysis, we wondered whether all the phases were necessary to reconstruct the signal or not. In the perspective that we need them all to perform the inverse STFT, if is possible to save only a few of them without losing speech in terms of quality, then we could store less information in memory for the decompression. If we analyze the signal phases, we can clearly see that they are highly uncorrelated with each other, so if we want to mask them we need to do it with criterion. Specifically, we have decided to investigate the consequences of discarding the phases corresponding to the weaker frequencies, as they are deemed to have the least impact on reconstructing the waveform. In particular, we have tried to remove the phases whose frequencies were below 60dB, 65dB, 70dB, and 75dB, then evaluate the PESQ and STOI metrics.

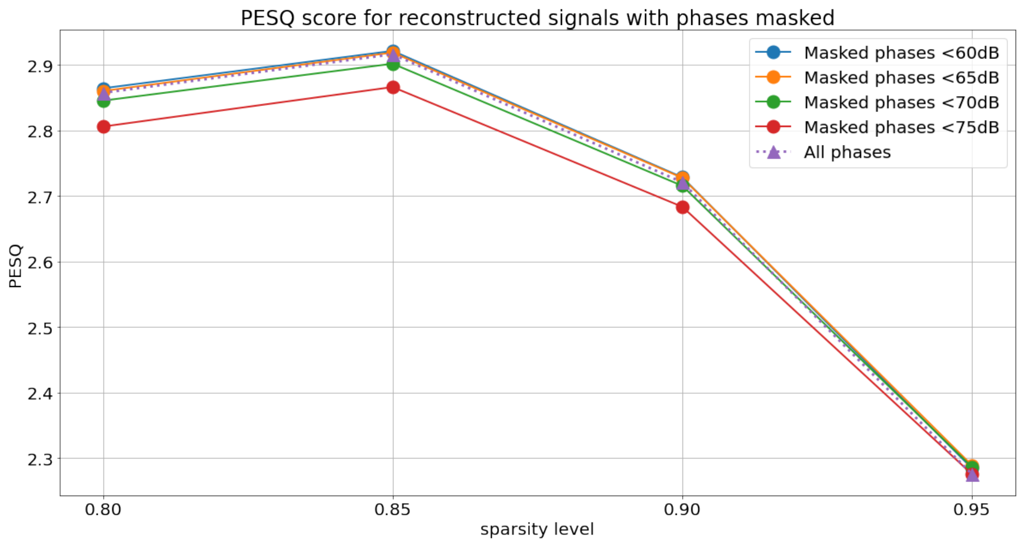

Figure 4.9: Different threshold for different different sparsity level evaluated with PESQ.

Summarized in figure 4.9 is represented the evaluation of the PESQ metrics with the phases masked at different thresholds in relation to a reconstruction that uses them all. Is interesting to notice that, when the sparsity level is low, the phases play a relevant role in the quality of the reconstruction, instead when the sparsity is higher using all the phases or a portion of them seems to be the same. The behavior could be justified by the fact that the dense output of the model makes the low frequency more uniform and the quality is only determined by the higher ones.

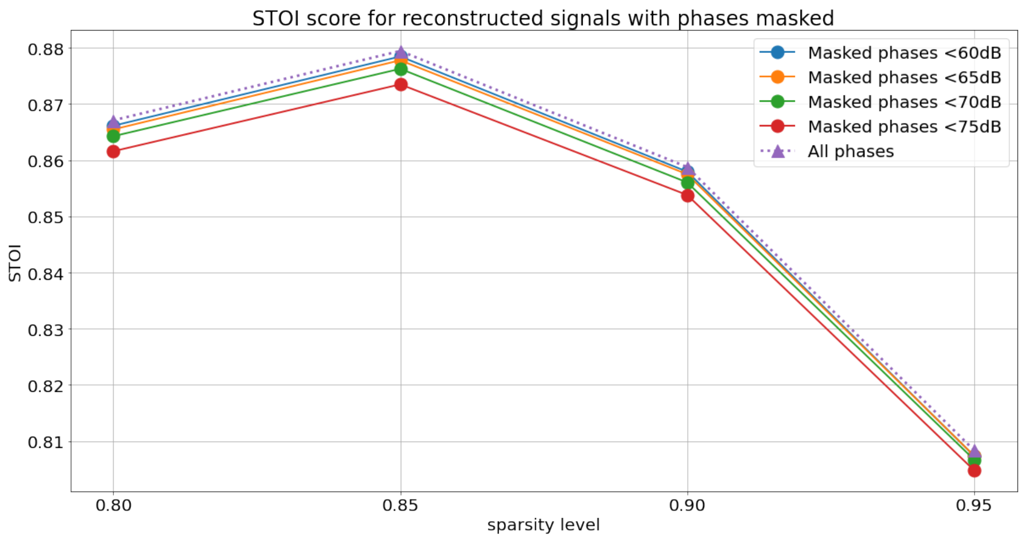

Figure 4.10: Different threshold for different different sparsity level evaluated with STOI.

On the other hand, in figure 4.10 we represent the average intelligibility score of the phrases. In contrast with the other metrics, removing a portion of the phases will negatively affect the reconstruction, but the difference is negligible.

In conclusion, we can say that removing phases of the weakest frequencies does not significantly reduce the quality and the intelligibility of the reconstructed signal. In particular, for the validation set, we could save, on average, the 44%, 38%, 32%, and 26% of the information of the phases masking them with the respective threshold at 60dB, 65dB, 70dB, and 75dB applied to the frequencies of the spectrogram.

4.4 Mask optimization

Throughout this section, we are going to talk about the results of our work regarding mask optimization. The challenge can be framed as follows: given a dense spectrogram, how can we represent it in a more compact form while minimizing the information loss that may occur during reconstruction by a Convolutional Neural Network (CNN)?

In our experimental setup, the spectrograms have a density of $1025 \times 302 = 309.550$ pixels, if we want to reduce the information by 95% the sparsity level constraint SL is set to $SL = 309.550 \times 0.95 = 15477$ which will correspond the number of zeros in the mask. As we said, the masks are generated ad-hoc for each sample, to describe the results of the optimization methods we have to fix one. The audio sample that we have analyzed is the one corresponding to the phrase: 'A woman rock climber scales a cliff far above pastures' which spectrogram is shown in figure 4.4.

For the first part, where we introduce the implementation of our optimization methods, we want to use as baseline the metrics evaluated using a random mask. In particular, by dwelling on input with a sparsity level of 95%, we have the following scores: MSE = 35.234, PESQ = 2.48, and STOI = 0.845.

4.4.1 Minimization problem

In order to solve the minimization problem we need to create the real values mask that we mentioned when we described the theory, we have sampled 95% of the values from the interval (-5,-4) and the remaining 5% in the interval (4,5). In this way, if we discretize the mask with a threshold on the sigmoid function at 0.5, we will preserve the right proportion of zeros and ones. We have calculated the minimization problem by approximating it using a gradient descent approach with the lagrangian multipliers. The evolution of the gradient of the Lagrangian function is shown in figure 4.11 which after 10000 steps, as we can see, reaches the local minimum and a good stopping point.

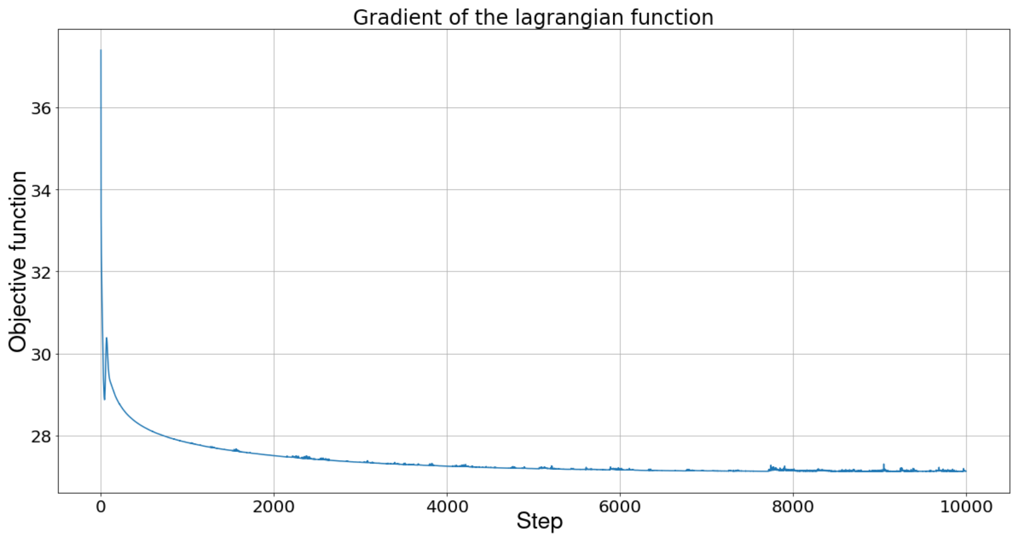

Figure 4.11: Gradient of the lagrangian function.

After the last iteration, we will obtain the optimized mask $\phi(M^*)$ that we can use to create the sparse input and evaluate it to see how the baseline is improved. The reconstruction performs better in every metric: MSE $=$ 27.360, PESQ $=$ 2.889, and STOI $=$ 0.875.

But will subsist a substantial problem, the optimized mask $\phi(M^*)$ has real values in (0,1), we could consider it a sort of weighted mask, so it violates the assumption of having only 0 and 1 values. If we discretize back the mask, setting the greatest SL values to one and the remaining to zeros, we notice that the results are slightly worst with respect to the previous mask obtaining the following scores: MSE $=$ 36.676, PESQ $=$ 2.36, STOI $=$ 0.832. The scores of the mask are worst because during the optimization process, the algorithm has difficulties flipping values to the other side, so we have 0 that move a little towards the 1 but don't flip and vice versa.

4.4.2 Genetic algorithm

For the genetic algorithms, we have performed the optimization process using a population of 50 individuals for a total of 500 steps. We did a total of four experiments, divided into two reproducing methods and two different fitness functions, the MSE and PESQ. The first experiment that we performed was to create random masks for each iteration and keep track of only the best one, we conducted this type of experiment to understand whether continuing to create random masks without criteria could find a mask that was suitable for the compression task.

The second experiment was made by actually reproducing the individuals as described in the section 3.2.2, in particular, in order to create the new population we have partitioned the individuals into three parts: %10 of them are the best possible one and they will directly pass in the next generation, %40 of the new population will be created by reproducing the %40 best individuals within them and the remaining %50 individuals are recreated at random.

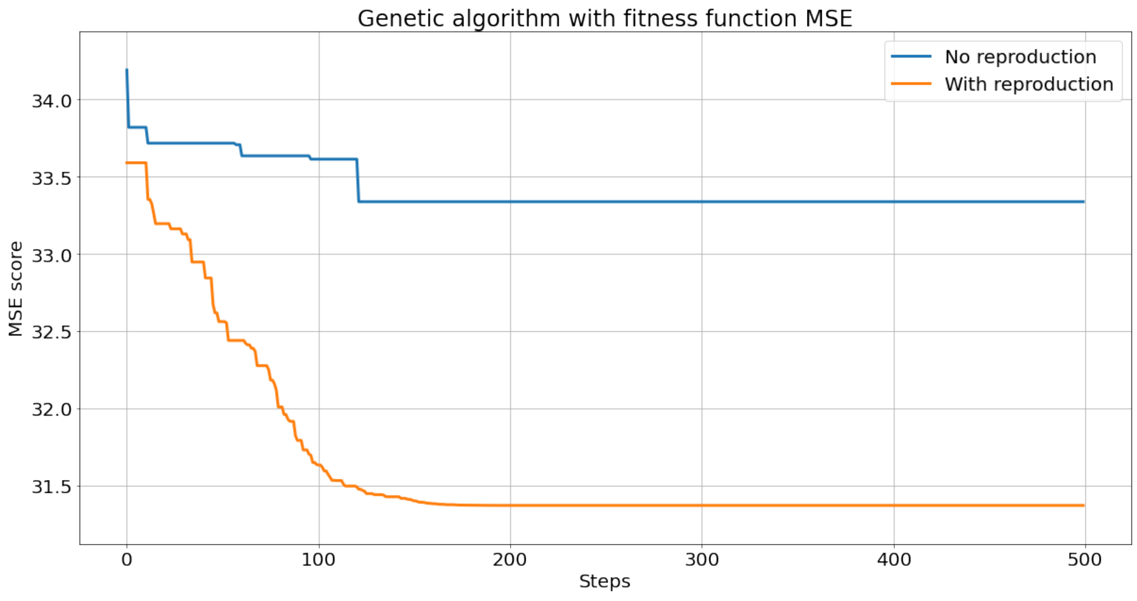

Figure 4.12: Evolution of the fittest individual with MSE ↓.

We have reported in figure 4.12 and in figure 4.13 the evolution of the genetic algorithms using the MSE and the PESQ metrics as fitness functions. As we expected, even if we generated $50 \times 500 = 25000$ masks at random the quality of the compression does not improve so much. In table 4.1 we have also summarized the performances of the dense spectrogram reconstructed by using the masks obtained with the genetic algorithms, in the rows of the table we have the metrics that we took into consideration, in the columns we represent if the genetic mask were obtained by reproduction or not and which metric was used as the fitness function.

About the results in the latter table we can make some important conclusions, although all algorithms minimize what is asked, in the end, we are more interested in metrics that have to do with the audio signal and not only with the spectrogram.

If we carefully observe, the MSE obtained by the mask generated with PESQ metrics is obviously higher than the one generated with the fitness function MSE, but what we didn't expect was the so much lower score for the PESQ metric generated with the MSE.

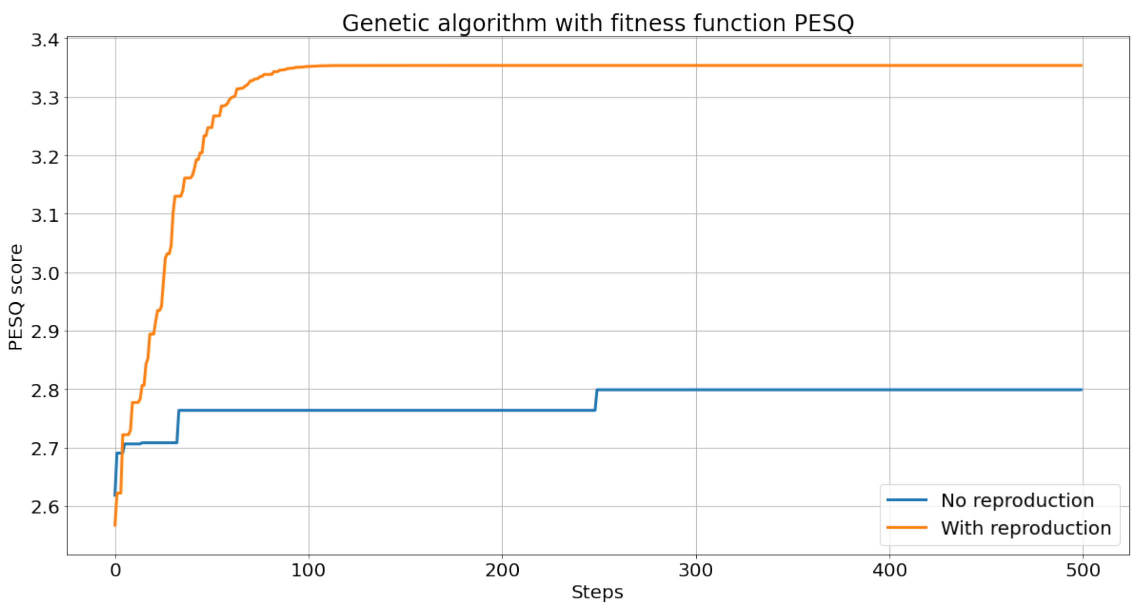

Figure 4.13: Evolution of the fittest individual with PESQ ↑.

	With reproduction		**No reproduction**	
	MSE	PESQ	MSE	PESQ
MSE ↓	**31.370**	34.597	33.337	34.553
PESQ ↑	2.802	**3.353**	2.481	2.798
STOI ↑	0.853	**0.869**	0.851	0.857

Table 4.1: Different performance for a mask at 95% of sparsity generated with genetic algorithms.

4.4.3 Different sparsity level

In this last subsection, we want to compare all the models that we have created, one for each target sparsity level, and discuss the quality of the reconstruction.

Starting from table 4.2 are summarized the evaluations of the PESQ score for the reconstructed signal of all the masks that we have generated, excluding the ones generated at random with genetic algorithms which are not pertinent. Our results suggest that the compression process is optimized when using masks generated through a genetic algorithm with PESQ as the fitness metric. This approach resulted in a noticeable improvement in perceptual quality, with an average increase of at least 0.5 across all sparsity levels.

	Type of mask				
	genetic PESQ	genetic MSE	min real	min binary	baseline
Sparsity 80%					
all phases	**3.54**	3.06	3.10	2.83	3.00
phase mask	3.52	3.06	3.09	2.83	3.01
Sparsity 85%					
all phases	**3.51**	3.08	3.16	2.94	2.92
phase mask	3.49	3.07	3.14	2.94	2.92
Sparsity 90%					
all phases	**3.40**	3.01	3.09	2.74	2.89
phase mask	3.37	3.01	3.10	2.75	2.89
Sparsity 95%					
all phases	**3.35**	2.80	2.88	2.36	2.48
phase mask	3.30	2.81	2.88	2.35	2.49

Table 4.2: Evaluation of PESQ ↑ metrics for all the masks and sparsity levels.

On the other hand in terms of intelligibility, which performances are summarized in table 4.3, we can see one more time how the decompression made with the mask generated with the genetic algorithm and PESQ perform slightly better. Despite the change in sparsity levels, the speech intelligibility of the phrases remains relatively unaffected, with only a 10% loss in comparison to a perfect reconstruction.

	Type of mask				
	genetic PESQ	genetic MSE	min real	min binary	baseline
Sparsity 80%					
all phases	**0.90**	0.89	0.89	0.88	0.88
phase mask	0.89	0.89	0.89	0.88	0.88
Sparsity 85%					
all phases	0.90	0.88	0.88	0.87	0.87
phase mask	**0.90**	0.88	0.89	0.88	0.87
Sparsity 90%					
all phases	0.89	0.87	0.89	0.86	0.87
phase mask	**0.89**	0.87	0.89	0.86	0.87
Sparsity 95%					
all phases	0.87	0.85	0.87	0.83	0.85
phase mask	**0.87**	0.85	0.87	0.83	0.85

Table 4.3: Evaluation of STOI ↑ metrics for all the masks and sparsity levels.

Chapter 5

Conclusion and future work

In this work, we have seen how is possible to create a lossy data-driven audio compression and decompression pipeline using a sparse convolutional neural network. In particular, the assumption that we have made of considering a spectrogram of an audio file as a depth map seems to be a good intuition. We have shown how is possible to reduce a spectrogram up to 95% of sparsity and reconstruct it, after the generation of an ad-hoc mask, without losing too much audio quality and intelligibility. Furthermore, for the initial phase of this exploration work, we need the phases in order to reconstruct the waveform back, from a perspective of having also this information stored in memory we tried to filter out the weaker frequency obtaining practically the same result, but also reducing, on average, the phases information of about 44%, 38%, 32%, 26% respectively for thresholds at 60dB, 65dB, 70dB, 75dB.

Is also important to make some considerations about the compression step, the ad-hoc masks were generated with two different optimization problems: crafting a non-linear programming problem and genetic algorithms. The first method is the more efficient in terms of convergence to the local minimum of the problem, but the drawback is that by softening the binary constraint the resultant mask will be weighted and not discrete. At the moment that we discretize such mask worst results are obtained because the ones and zeroes of the mask struggle to change state, but move a little from the 0 and 1 values.

On the other hand, genetic algorithms, in which the mask generation takes much more time with respect to the previous method, can reconstruct the signal in a better way. In particular, we have used two metrics as fitness function MSE and PESQ. Although the first metric is also the same one that we have used to train our model, in term of audio quality the second metric, which is more interesting in the end, outperform the audio quality retrieved with the first metric. From this we can assert that MSE is not a proxy for audio quality metrics despite the MSE loss is commonly used for depth completion tasks.

The first step that can be taken to possibly improve the quality of the reconstruction is to train the model using a perceptual loss, for example, we can compute the Mel-frequency cepstral coefficients (MFCC) [74] for both the original and reconstructed signals and then compute the l_2 loss as a proxy of perceptual distance [53].

Is also interesting to explore how we can further improve the quality of the audio file if we can enhance the noisy output spectrogram [48] with the information that we have available.

Lastly, we want to mention that in signal audio processing working with speech files or music compression can lead to differences in terms of assumption. In particular during the construction of the masks, if are working with speech files, we can try to introduce some psychoacoustics constraints in order to improve the quality and intelligibility of the reconstruction.

Bibliography

[1] Jonathan Sterne. *MP3: The meaning of a format*. Duke University Press, 2012.

[2] Karlheinz Brandenburg. Mp3 and aac explained. In *Audio Engineering Society Conference: 17th International Conference: High-Quality Audio Coding*. Audio Engineering Society, 1999.

[3] Davis Pan. A tutorial on mpeg/audio compression. *IEEE multimedia*, 2(2):60–74, 1995.

[4] Mat Hans and Ronald W Schafer. Lossless compression of digital audio. *IEEE Signal processing magazine*, 18(4):21–32, 2001.

[5] Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

[6] Enders Robinson and Dean Clark. Sampling and the nyquist frequency. *The Leading Edge*, 10(3):51–53, 1991.

[7] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[8] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[9] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3642–3649. IEEE, 2012.

[10] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868, 2008.

[11] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[12] Martin Sundermeyer, Hermann Ney, and Ralf Schlüter. From feedforward to recurrent lstm neural networks for language modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):517–529, 2015.

[13] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[14] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[16] Farhana Sultana, Abu Sufian, and Paramartha Dutta. A review of object detection models based on convolutional neural network. *Intelligent computing: image processing based applications*, pages 1–16, 2020.

[17] Leonard Berrada, Andrew Zisserman, and M Pawan Kumar. Training neural networks for and by interpolation. In *International conference on machine learning*, pages 799–809. PMLR, 2020.

[18] Nathalie Plaziac. Image interpolation using neural networks. *IEEE transactions on image processing*, 8(11):1647–1651, 1999.

[19] Christian Häne, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, Paul Furgale, Torsten Sattler, and Marc Pollefeys. 3d visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection. *Image and Vision Computing*, 68:14–27, 2017.

[20] Arindam Dey, Graeme Jarvis, Christian Sandor, and Gerhard Reitmayr. Tablet versus phone: Depth perception in handheld augmented reality. In *2012 IEEE international symposium on mixed and augmented reality (ISMAR)*, pages 187–196. IEEE, 2012.

[21] Hongmin Liu, Xincheng Tang, and Shuhan Shen. Depth-map completion for large indoor scene reconstruction. *Pattern Recognition*, 99:107112, 2020.

[22] Mokhtar S Bazaraa, Hanif D Sherali, and Chitharanjan M Shetty. *Nonlinear programming: theory and algorithms*. John Wiley & Sons, 2013.

[23] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[24] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[25] Randy L Haupt and Sue Ellen Haupt. *Practical genetic algorithms*. John Wiley & Sons, 2004.

[26] Lawrence Davis. Handbook of genetic algorithms. 1991.

[27] Junjie Hu, Chenyu Bao, Mete Ozay, Chenyou Fan, Qing Gao, Honghai Liu, and Tin Lun Lam. Deep depth completion: A survey. *arXiv preprint arXiv:2205.05335*, 2022.

[28] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *2017 international conference on 3D Vision (3DV)*, pages 11–20. IEEE, 2017.

[29] Nathaniel Chodosh, Chaoyang Wang, and Simon Lucey. Deep convolutional compressed sensing for lidar depth completion. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part I 14*, pages 499–513. Springer, 2019.

[30] Zixuan Huang, Junming Fan, Shenggan Cheng, Shuai Yi, Xiaogang Wang, and Hongsheng Li. Hms-net: Hierarchical multi-scale sparsity-invariant network for sparse depth completion. *IEEE Transactions on Image Processing*, 29:3429–3441, 2019.

[31] A Eldesokey, M Felsberg, and FS Khan. Propagating confidences through cnns for sparse data regression. arxiv 2018. *arXiv preprint arXiv:1805.11913*, 2018.

[32] Abdelrahman Eldesokey, Michael Felsberg, Karl Holmquist, and Michael Persson. Uncertainty-aware cnns for depth completion: Uncertainty from beginning to end. 2020 ieee. In *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12011–12020, 2020.

[33] Martin Dimitrievski, Peter Veelaert, and Wilfried Philips. Learning morphological operators for depth completion. In *Advanced Concepts for Intelligent Vision Systems: 19th International Conference, ACIVS 2018, Poitiers, France, September 24–27, 2018, Proceedings 19*, pages 450–461. Springer, 2018.

[34] Fangchang Ma and Sertac Karaman. Sparse-to-dense: Depth prediction from sparse depth samples and a single image. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4796–4803. IEEE, 2018.

[35] Jiaxiong Qiu, Zhaopeng Cui, Yinda Zhang, Xingdi Zhang, Shuaicheng Liu, Bing Zeng, and Marc Pollefeys. Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3313–3322, 2019.

[36] Chen Fu, Chiyu Dong, Christoph Mertz, and John M Dolan. Depth completion via inductive fusion of planar lidar and monocular camera. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10843–10848. IEEE, 2020.

[37] Yongchi Zhang, Ping Wei, Huan Li, and Nanning Zheng. Multiscale adaptation fusion networks for depth completion. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020.

[38] Shanshan Zhao, Mingming Gong, Huan Fu, and Dacheng Tao. Adaptive context-aware multi-modal network for depth completion. *IEEE Transactions on Image Processing*, 30:5264–5276, 2021.

[39] Yun Chen, Bin Yang, Ming Liang, and Raquel Urtasun. Learning joint 2d-3d representations for depth completion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10023–10032, 2019.

[40] L Liu, X Song, X Lyu, J Diao, M Wang, Y Liu, and L Zhang. Fcfr-net: Feature fusion based coarse-to-fine residual learning for depth completion. arxiv 2020. *arXiv preprint arXiv:2012.08270*, 2020.

[41] Yufan Zhu, Weisheng Dong, Leida Li, Jinjian Wu, Xin Li, and Guangming Shi. Robust depth completion with uncertainty-driven loss functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 3626–3634, 2022.

[42] Xinjing Cheng, Peng Wang, Chenye Guan, and Ruigang Yang. Cspn++: Learning context and resource aware convolutional spatial propagation networks for depth completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10615–10622, 2020.

[43] Mu Hu, Shuling Wang, Bin Li, Shiyu Ning, Li Fan, and Xiaojin Gong. Penet: Towards precise and efficient image guided depth completion. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13656–13662. IEEE, 2021.

[44] Philipos C Loizou. *Speech enhancement: theory and practice.* CRC press, 2007.

[45] Yariv Ephraim. Statistical-model-based speech enhancement systems. *Proceedings of the IEEE*, 80(10):1526–1555, 1992.

[46] Steven Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on acoustics, speech, and signal processing*, 27(2):113–120, 1979.

[47] Michael Berouti, Richard Schwartz, and John Makhoul. Enhancement of speech corrupted by acoustic noise. In *ICASSP'79. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 208–211. IEEE, 1979.

[48] Szu-Wei Fu, Ting-yao Hu, Yu Tsao, and Xugang Lu. Complex spectrogram enhancement by convolutional neural network with multi-metrics learning. In *2017 IEEE 27th international workshop on machine learning for signal processing (MLSP)*, pages 1–6. IEEE, 2017.

[49] Santiago Pascual, Antonio Bonafonte, and Joan Serra. Segan: Speech enhancement generative adversarial network. *arXiv preprint arXiv:1703.09452*, 2017.

[50] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

[51] KA Ramya, M Pushpa, J Kaur, and N Kaur. A survey on lossless and lossy data compression methods. *International Journal of Computer Science and Engineering Communications*, 4(1):1277–1280, 2016.

[52] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *arXiv preprint arXiv:2003.05991*, 2020.

[53] Srihari Kankanahalli. End-to-end optimized speech coding with deep neural networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2521–2525. IEEE, 2018.

[54] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2021.

[55] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438*, 2022.

[56] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[57] Daniel Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on acoustics, speech, and signal processing*, 32(2):236–243, 1984.

[58] Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 workshop on creating speech and language data with Amazon's Mechanical Turk*, pages 139–147, 2010.

[59] David Harwath and James Glass. Deep multimodal semantic embeddings for speech and images. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 237–244. IEEE, 2015.

[60] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[61] Brian McFee, Colin Raffel, Dawen Liang, Daniel P Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, pages 18–25, 2015.

[62] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[63] Antony W Rix, John G Beerends, Michael P Hollier, and Andries P Hekstra. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*, volume 2, pages 749–752. IEEE, 2001.

[64] Cees H Taal, Richard C Hendriks, Richard Heusdens, and Jesper Jensen. A short-time objective intelligibility measure for time-frequency weighted noisy

speech. In *2010 IEEE international conference on acoustics, speech and signal processing*, pages 4214–4217. IEEE, 2010.

[65] Schuyler R Quackenbush, Thomas P Barnwell, and Mark A Clements. *Objective measures of speech quality*. Prentice-Hall, 1988.

[66] Philipos C Loizou. Speech quality assessment. In *Multimedia analysis, processing and communications*, pages 623–654. Springer, 2011.

[67] John G Beerends, Christian Schmidmer, Jens Berger, Matthias Obermann, Raphael Ullmann, Joachim Pomy, and Michael Keyhl. Perceptual objective listening quality assessment (polqa), the third generation itu-t standard for end-to-end speech quality measurement part i—temporal alignment. *journal of the audio engineering society*, 61(6):366–384, 2013.

[68] Andrew Hines, Jan Skoglund, Anil C Kokaram, and Naomi Harte. Visqol: an objective speech quality model. *EURASIP Journal on Audio, Speech, and Music Processing*, 2015(1):1–18, 2015.

[69] Pranay Manocha, Adam Finkelstein, Richard Zhang, Nicholas J Bryan, Gautham J Mysore, and Zeyu Jin. A differentiable perceptual audio metric learned from just noticeable differences. *arXiv preprint arXiv:2001.04460*, 2020.

[70] Pranay Manocha, Zeyu Jin, Richard Zhang, and Adam Finkelstein. Cdpam: Contrastive learning for perceptual audio similarity. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 196–200. IEEE, 2021.

[71] Chandan KA Reddy, Vishak Gopal, and Ross Cutler. Dnsmos: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6493–6497. IEEE, 2021.

[72] Gabriel Mittag, Babak Naderi, Assmaa Chehadi, and Sebastian Möller. Nisqa: A deep cnn-self-attention model for multidimensional speech quality prediction with crowdsourced datasets. *arXiv preprint arXiv:2104.09494*, 2021.

[73] Pranay Manocha, Zeyu Jin, and Adam Finkelstein. Sqapp: No-reference speech quality assessment via pairwise preference. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 891–895. IEEE, 2022.

[74] Lindasalwa Muda, Mumtaj Begam, and Irraivan Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. *arXiv preprint arXiv:1003.4083*, 2010.

Ringraziamenti

Vorrei dedicare questo piccolo, grande, traguardo a tutte le persone che, in un modo e nell'altro, hanno fatto parte della mia vita e sono riuscite ad arricchirla.

Inizio con il ringraziare i miei relatori, prof. Pistellato e prof. Bergamasco, che mi hanno guidato e supportato durante la stesura della tesi e da cui ho potuto apprezzare la vera passione per la ricerca.

Spendo poi un paio di parole per le persone con cui ho stretto più di un semplice rapporto di amicizia tra i banchi. Con loro ho potuto condividere gioie e dolori, notti passate per finire i progetti e tutti gli esami preparati assieme. La squadra vincente per chi ci sta scommettendo, il team trios come piace chiamarci: Dario e Francesco.

Infine vorrei ringraziare la mia famiglia, mio padre, mia madre, mio fratello e la mia fidanzata Gloria, sempre presenti nella mia vita e in ogni momento di difficoltà.