Universidad de Alcalá
Escuela Politécnica Superior

Máster Universitario en Ingeniería de Telecomunicación

Trabajo Fin de Máster

Design, implementation and evaluation of an acoustic source localization system using *Deep Learning* techniques

Author: Juan Manuel Vera Díaz

Advisor: Daniel Pizarro Pérez

2019

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Máster Universitario en Ingeniería de Telecomunicación

Trabajo Fin de Máster

Design, implementation and evaluation of an acoustic source localization system using *Deep Learning* techniques

Author: Juan Manuel Vera Díaz

Advisor: Daniel Pizarro Pérez

Tribunal:

President: Javier Macías Guarasa

1st Vocal: David Fernández Barrero

2nd Vocal: Daniel Pizarro Pérez

Calification: .....

Date: .....

Deux ex machina

Acknowledgements

Este Trabajo de Fin de Máster ha sido fruto de muchas horas de dedicación, esfuerzo y constancia. No podría haberlo hecho sin ayuda de mi tutor Daniel Pizarro. A él le quiero agradecer todas las horas de charlas, ideas y consejos que me ha ido ofreciendo, las cuales me han permitido presentar este trabajo. Por otro lado, también me gustaría agradecer toda la ayuda prestada a Javier Macías, que siempre ha estado disponible para explicarme cualquier cosa que necesitase, ya fuese realicionada con el tema de este trabajo o no. He de reconocer que es muy fácil trabajar con vosotros dos.

También me gustaría agradecer a todos mis compañeros del *ISPACE* todo el apoyo que me han dado, aunque he de decir que muchos de ellos ya no son solamente compañeros y han pasado a ser amigos. En especial quiero mencionar a dos de ellos: A Fuentes, que es una enciclopedia andante sobre las CNN, gracias por estar siempre dispuesto a prestar ayuda, y a Casillas, por ser como es, hacer suyos los problemas que no lo son y ayudarme a resolverlos. Espero que pese a que te vayas dentro de poco como señor Doctor Casillas, eso no se convierta en un adiós.

Siguiendo con los agradecimientos, quiero agradecer a mi familia por creer siempre en mí. A mi padre, por intentar ayudarme siempre en todo lo que puede con sus consejos, que aunque a mí a veces me cueste aceptarlos, son los mejores. A mi madre, por ser la voz de mi conciencia y tener una paciencia infinita conmigo. Y a mi hermano, por interesarse por lo que hago e intentar comprenderlo, ¡a ver cuando me explicas tú cosas chulas de física! Gracias a los tres por apoyarme en todo y estar tan orgullosos de mí.

Por último, agradecer a Leti por estar siempre ahí. Por ser compañera, amiga y pareja, por aguantarme a las buenas y a las malas y por creer en mí, pase lo que pase. Una vez leí que una buena pareja no era aquella que se miraba a los ojos, sino la que miraba en la misma dirección. Contigo es muy fácil encontrar esa dirección. ¡Gracias por todo!

Seguramente me dejo a mucha gente en el tintero. A todos los que no he nombrado, pero que han participado de manera directa o indirecta en este trabajo... ¡MUCHAS GRACIAS A TODOS!

Abstract

This *Master Thesis* presents a novel approach for indoor acoustic source localization using microphone arrays, based on a Convolutional Neural Network (CNN) that we call the *ASLNet*. It directly estimates the three-dimensional position of a single acoustic source using as inputs the raw audio signals from a set of microphones. We use supervised learning methods to train our network end-to-end. The amount of labeled training data available for this problem is however small. This Thesis presents a training strategy based on two steps that mitigates this problem. We first train our network using semi-synthetic data generated from close talk speech recordings and a mathematical model for signal propagation from the source to the microphones. The amount of semi-synthetic data can be virtually as large as needed. We then fine tune the resulting network using a small amount of real data. Our experimental results, evaluated on a publicly available dataset recorded in a real room, show that this approach is able to improve existing localization methods based on SRP-PHAT strategies and also those presented in very recent proposals based on Convolutional Recurrent Neural Networks (CRNN). In addition, our experiments show that the performance of the *ASLNet* does not show a relevant dependency on the speaker's gender, nor on the size of the signal window being used. This work also investigates methods to improve the generalization properties of our network using only semi-synthetic data for training. This is a highly important objective due to the cost of labelling localization data. We proceed by including specific effects in the input signals to force the network to be insensitive to multipath, high noise and distortion likely to be present in real scenarios. We obtain promising results with this strategy although they still lack behind strategies based on fine-tuning.

Keywords: Acoustic source localization, microphone arrays, deep learning, convolutional neural networks.

Contents

List of Figures

List of Tables

List of Acronyms

ANN	Artificial Neural Network.
ANNs	Artificial Neural Networks.
ASL	Acoustic Source Localization.
CNN	Convolutional Neural Networks.
CNNs	Convolutional Neural Networks.
DOA	Direction of Arrival.
FIR	Finite Impulse Response.
GCC	Generalized Cross-Correlation.
IFFT	Inverse Fast Fourier Transform.
MOTP	Multiple Object Tracking Precision.
MUSIC	Multiple Signal Classification.
MVDR	Minimum Variance Distortionless Response.
PHAT	Phase Transform.
RIR	Room Impulse Response.
SGD	Stochastic Gradient Descent.
SNR	Signal to Noise Ratio.
SRC	Stochastic Region Contraction.
SRP	Steered Response Power.
SRP-PHAT	Steered Response Power Phase Transform.
TDOA	Time Difference Of Arrivals.

Chapter 1

Introduction

The only way to do great work is to love what you do. If you haven't found it yet, keep looking. Don't settle.

Steve Jobs

One of the main tasks in the field of human-machine interaction, with a tremendous impact in other applied fields, is to know the position of human beings that interact within an environment [3–5]. Non-invasive technologies are preferred in this context, avoiding the need of wearable devices for localization. The two main non-invasive techniques used in indoor localization systems are those based on video and acoustic sensors. These modalities can be used together or separately to perform this task. Figure 1.1 shows a non-invasive localization system based on video and audio sensors. In particular, there is a vast

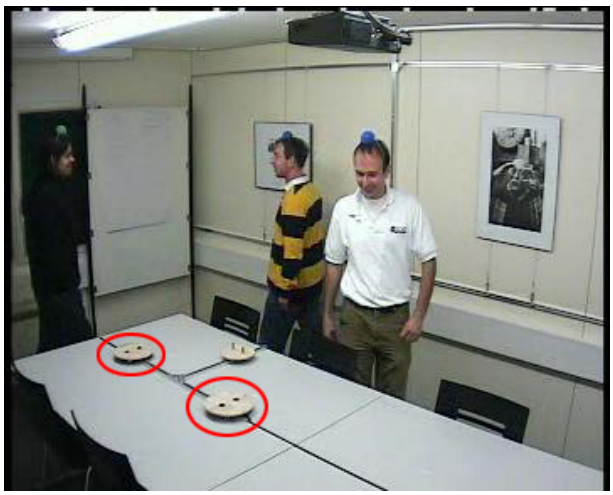

Figure 1.1: Example of a multimodal localization system [1]. Note that the red circles shows where the microphone arrays are located.

amount of research focused on obtaining the position of any acoustic source that appears in a scene from a set of microphones placed in the environment. This is known as the Acoustic Source Localization (ASL) problem and it is very challenging, mainly due to the presence of reverberation and noise. Most of existing ASL methods try to estimate the Direction of Arrival (DOA) of the source. Other works obtain the source elevation and azimuth and just a few of them get the exact position of the acoustic source in space. The latter is the object of interest in this Thesis. The ASL problem has also a high impact on many domains, such as speech enhancement, beam-forming techniques or people indoor tracking.

Nowadays, the vast majority of ASL works focus on locating human beings or other acoustic sources in indoor environments using two main approaches: *i) model-based* methods, which use mathematical models to relate the audio data with the source position and *ii) learning-based* methods, based on learning a regression function to obtain the source position directly from the raw audio data. Outdoor ASL remains an open problem, where most of the methods fail due to the non-linear effects of outdoor sound propagation, which is caused by the presence of wind, high noise and distortion. In particular, this Master Thesis focuses on indoor ASL and proposes a method based on modern Deep Learning techniques. The main goals of this Thesis are detailed next.

1.1 Goals

This Thesis focuses on methods for indoor ASL using a set of microphone arrays, which will record the acoustic signals that occur in a known room. The main objective of this work is to develop a learning-based localization method capable of improving the accuracy of current state-of-the-art methods. To this end we propose a Convolutional Neural Networks (CNN) architecture, hereinafter *ASLNet*, which is trained end-to-end to solve the localization problem. The inputs are the raw audio signals from the set of microphones and the output is the position of the speaker in Cartesian coordinates with respect the room's coordinate origin. A simple scheme of the proposed system is shown in figure 1.2.

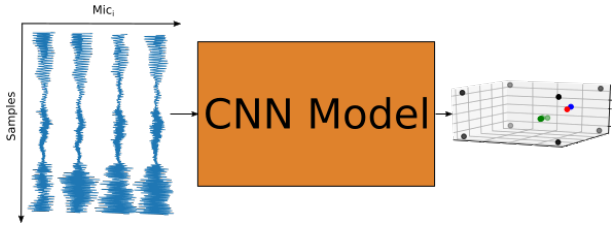

Figure 1.2: Simple scheme of the proposed system.

We train our network in two stages. In the first stage we use a phonetic corpus *Albayzin* [6] and an audio propagation model to create a large semi-synthetic dataset for training our network. In the second stage we use *domain adaptation* [7,8] techniques to adapt our network to a real data dataset [1] recorded at *IDIAP* research institute. These two stages are necessary due to the scarcity of the real data available for this task. We also study how well our network is capable to generalize only training from semi-synthetic data, which can be made virtually as larger as needed, to work under real data conditions. To this end this Thesis also include several data augmentation methods and simulation techniques in order to improve the performance of our network with real data when only synthetic data is used for training. In all our experiments we use as baseline the results obtained with the Steered Response Power (SRP) [9] method, which is one of the most popular within the field of ASL, as well as with a state-of-the-art method proposed in [10].

1.2 Document organization

The rest of this document is organized in the following chapters:

- **State-of-the-art**: it describes previous works in ASL methods. It begins by defining the concept of Time Difference Of Arrivals (TDOA), on which most conventional methods are based, to finally introduce the concept of deep learning and the Artificial Neural Network (ANN).

- **Proposed system**: this chapter details the topology of the designed network, the training strategies that will be carried out, as well as the different processes of synthetic data generation that have been implemented.
- **Experimental work and results**: it shows the experiments that have been carried out to verify the performance of the proposed method.
- **Conclusions and future work**: this chapter shows the conclusions and the future lines of research.
- **Appendices**: the document includes two additional appendices that reflect the budget necessary to carry out this work as well as the necessary tools and resources.

Chapter 2

State-of-the-art

Tell me and I forget. Teach me and I remember.
Involve me and I learn.

Benjamin Franklin

In the existing scientific literature there are several proposals that address the ASL problem. We distinguish between *model-based* methods and *learning-based* methods. Model-based methods, also known as classic methods, use signal propagation mathematical models, signal processing techniques, and optimization methods to recover the source localization information from the audio signals. Examples of these techniques are the methods based on estimating time delays [11–15], beamforming techniques [9, 16–18] and high-resolution spectral-estimation [11, 17]. Learning-based methods estimate an unknown mathematical function from training data that directly relates the solution of the ASL problem with the audio signals [19, 20]. Recently, deep learning methods have been proposed to solve this problem showing very promising results [2, 21].

2.1 Model-based methods

Inside this category there are three main approaches to solve the ASL problem: time delay estimation methods, beamforming methods and high-resolution spetral-estimation methods. We describe them in detail in the following subsections.

2.1.1 Time delay

Methods based on time delay estimate the location of the acoustic source that has emitted the signal by estimating the phase shift between the recorded signals. The phase shift is directly related with the TDOA [11–15] between signals arriving to different microphones. Given the TDOA, there exist hyperbolic trilateration methods which obtain the three-dimensional position of an acoustic source. Time delay methods are also referred to as indirect methods or two-step localization methods because they first obtain the TDOA and then compute the position of the source via trilateration. Next, we explain in detail how to estimate position of an acoustic source from the TDOA of several pairs of microphones, as well as one of the most used methods for estimating TDOA, known as GCC [22].

2.1.1.1 Time Difference of Arrival

Localizing a source given the TDOA measurements between pairs of microphones has been extensively used in the field of ASL. Given an unknown punctual acoustic source at position $s = (x_s, y_s, z_s)^\top$ and N microphones at known positions $m_n = (x_n, y_n, z_n)^\top$ with $n = 1 \ldots, N$, the TDOA between the signal emitted by the source and received in a pair of microphones is defined as

$$\Delta\tau(s, m_n, m_k) = \frac{1}{c}(\parallel m_n - s \parallel - \parallel m_k - s \parallel), \tag{2.1}$$

where $\parallel \cdot \parallel$ denotes the Euclidean norm, c is the sound propagation velocity (343 $\frac{m}{s}$ at 20 $°C$) and $\Delta\tau(s, m_n, m_k)$ is the TDOA between position s and the microphone pair, measured in seconds. From equation 2.1, we can deduce the possible locations of the acoustic source s that correspond to the same TDOA, which are arranged in the form of hyperboloid of revolution as it is shown in figure 2.1.

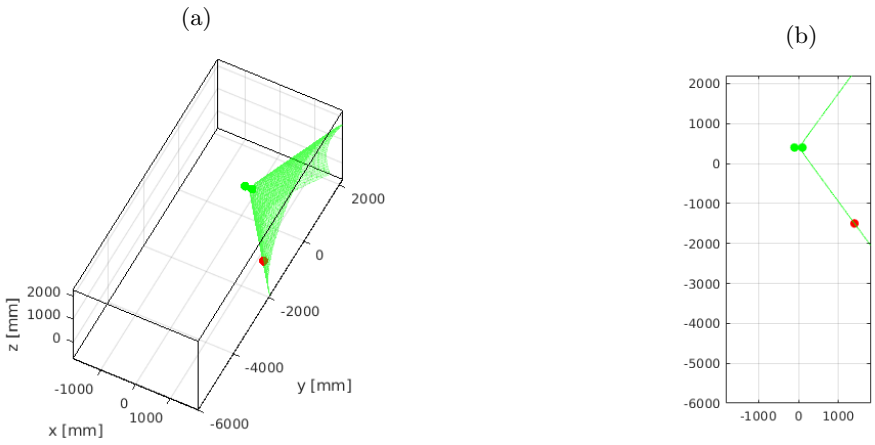

Figure 2.1: Geometry of the constant TDOA measurements for a source s and a given microphone pair (a) 3-D hyperboloid, (b) 2-D hyperbola (obtained by intersecting the hyperboloid with a plane).

$$\begin{cases} \frac{x^2}{a^2} - \frac{y^2}{b^2} - \frac{z^2}{b^2} = 1 \\ a = c\frac{\Delta\tau(s, m_n, m_k)}{2} \\ b = \sqrt{f^2 - a^2} \end{cases} \tag{2.2}$$

Each additional pair of microphones generates a new hyperboloid that passes through the acoustic source position, as shown in figure 2.2. Therefore, using additional pairs can be used to obtain the current position of the acoustic source, that lies at the intersection of all the hyperboloids.

2.1.1.2 Generalized Cross-Correlation

The GCC method is one of the most used to estimate the TDOA of two signals recorded from a given pair of microphones. Basically it is based on the fact that when correlating the signals recorded by two microphones from the same source, the maximum of the correlation is at the TDOA between the microphones. Formally, given an unknown source s, the signal recorded by a microphone can be expressed in the frequency domain as

$$M_n = H_n\, S + N_n, \tag{2.3}$$

where H_n is the Room Impulse Response (RIR) between the source and the n^{th} microphone, S is the signal emited by the source and N_n is a white gausian noise included in the signal by the microphone.

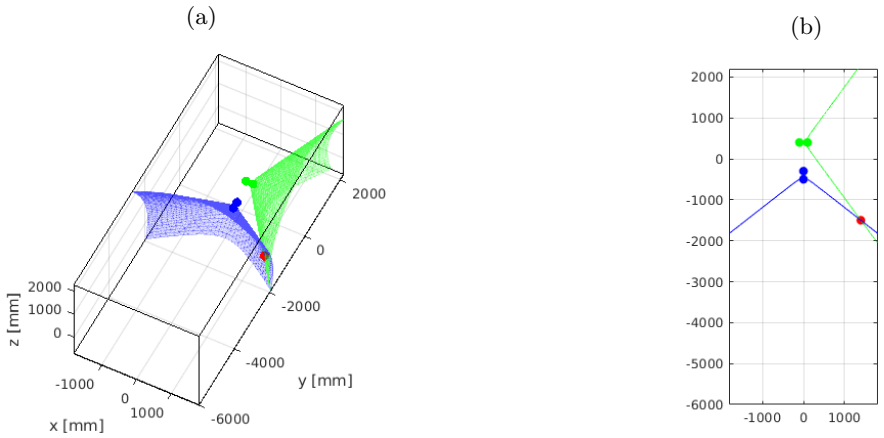

Figure 2.2: Geometry of the constant TDOA measurements for a source s and two given microphone pairs (a) 3-D hyperboloid, (b) 2-D hyperbola (obtained by intersecting the hyperboloid with a plane).

Taking into account only the effects due to propagation, avoiding the reverberation and other undesired effects of the environment, the signals recorded by a pair of microphones are related by a phase shift. Therefore the signals could be expressed as

$$\begin{cases} M_n = S + N_n \\ M_k = S\, e^{-j\omega\Delta\tau(s,m_n,m_k)} + N_m. \end{cases} \tag{2.4}$$

The GCC of two signals is given by the expression

$$R_{M_n,M_k}(\Delta\tau) = \mathcal{F}^{-1}\left\{M_n\, M_k^*\right\}, \tag{2.5}$$

where $\mathcal{F}^{-1}$ is the inverse of the Fourier Transform. Assuming that N_m and N_n are mutually uncorrelated and white, the result of equation 2.5 is approximately a shifted version of the self-correlation of S by the amount of TDOA that exists between both signals (see figure 2.3). Therefore, in general the TDOA can be estimated based on where the maximum of the GCC function is located:

$$\widehat{\Delta\tau}(s, m_n, m_k) = \underset{\Delta\tau}{\operatorname{argmax}}\, R_{M_n,M_k}(\Delta\tau). \tag{2.6}$$

For instance, given two signals where one is a distorted and delayed version of the other, they are received at the microphones with a TDOA around the $2.2ms$. We perform the GCC function and the result is shown in figure 2.3. We observe that the maximum peak is located at $\widehat{\Delta\tau}(s, m_n, m_k) \approx 2.2ms$.

A variant of this algorithm widely used in the state-of-the-art is the so-called GCC-PHAT algorithm, which performs the same process as GCC with the difference that a Phase Transform (PHAT) [22] filter is applied to the correlated signals. The PHAT filter normalizes the modulus of the signal spectrum, leaving the phase untouched.

2.1.1.3 Trilateration from TDOA

Hyperbolic trilateration algorithms estimate the source position $\widehat{s} = (x_{\hat{s}}, y_{\hat{s}}, z_{\hat{s}})^\top$ from a set of TDOA measurements in different microphone pairs. Usually this goal is achieved by minimizing the mean squared

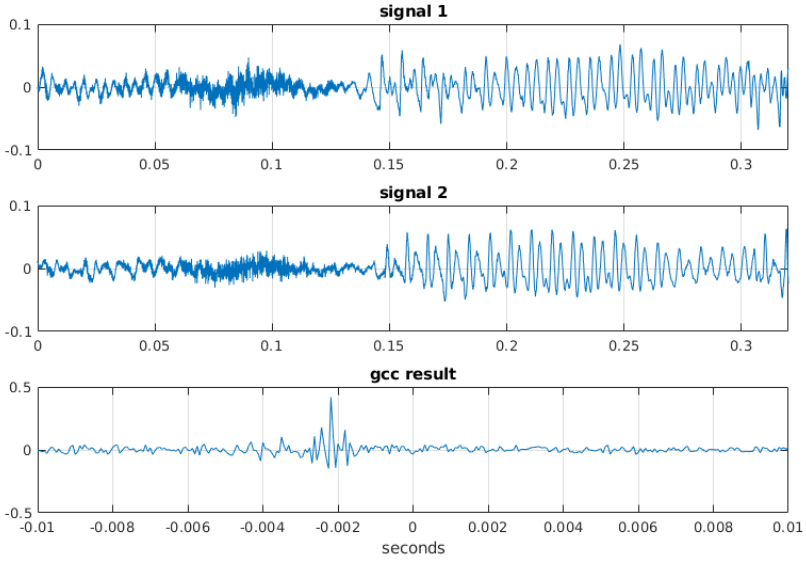

Figure 2.3: Result obtained by GCC for the two signals shown.

error between the estimated TDOA and its predicted value given the source and microphone positions.

$$\widehat{s} = arg \min_{s}(\sum_{n=1}^{N_m} \sum_{k=i+1}^{N_m} (\| \widehat{\Delta\tau}_{n,k} - \Delta\tau(s, m_n, m_k) \|)), \tag{2.7}$$

where $\widehat{\Delta\tau}_{n,k}$ is the estimated TDOA from the (m, n) pair of microphones. Popular algorithms for minimizing the cost equation 2.7 are based on gradient descend methods, such as *Gauss-Newtown* [23] or *Levenberg Marquardt* [24]. This type of techniques do not achieve good performance in highly noisy and reverberant environments, due to the distortion that occurs during TDOA estimation. Another phenomenon to be taken into account is that to solve the TDOA equations for the three Cartesian coordinates, at least three measurements of TDOA are needed, which means that at least three pairs of microphones are needed.

2.1.2 Beamforming

Beamforming techniques [16] estimate the position of the source by sampling a set of possible spatial locations and computing a beamforming function at each location. This approach chooses the source location that maximizes a statistic, that is maximum when the target position matches the source location.

Formally, consider an unknown acoustic source at the position $s = (x_s, y_s, z_s)^\top$ and a set of N microphones, where the n^{th} microphone is located at $m_n = (x_n, y_n, z_n)^\top$. A signal recorded by the n^{th} microphone can be expressed in frequency domain as

$$R_n = H_n \, S + N_n, \tag{2.8}$$

where S is the signal emited by the acoustic source s, H_n is the RIR between the source and the n^{th} microphone and N_n is the additive non-correlated noise signal for this microphone.

We denote by $bf(q, r, m)$ the beamforming function. It takes as arguments the set of signals $r = [R_1, R_2, \cdots, R_N]^\top$ received by the microphones, the set of microphone positions $m = [m_1, m_2, \cdots, m_N]^\top$, and a virtual source position $q = (x, y, z)$. The value of $bf(q, r, m)$ is a scalar that represents the *coherence*

of the virtual position q with respect to the received signals and microphone positions, being maximal if q is equal to the source true position s. The output of the beamforming function is also usually identified as some sort of acoustic power of the virtual source. Based on the beamforming function we can estimate the position of an acoustic source $\hat{s}$ as

$$\hat{s} = \underset{q}{\operatorname{argmax}}\, bf(q, r, m). \tag{2.9}$$

The most commonly used beamforming functions are the SRP and the Minimum Variance Distortionless Response (MVDR) which are described below. A simple example of this kind of techniques is given in figure 2.4 where the beamforming function is sampled at a uniform grid of two-dimensional positions, usually called the acoustic map, using two pairs of microphones (seen as blue and green points in the map). We can see that the ideal hyperbolas that correspond to the acoustic source (seen as dashed blue and green lines) correspond to ridges in the beamforming function. The estimation of the acoustic source will be the area with higher acoustic power. In this example it is identified as a red circle. Note that it is also the place where the two hyperbolas intersect.

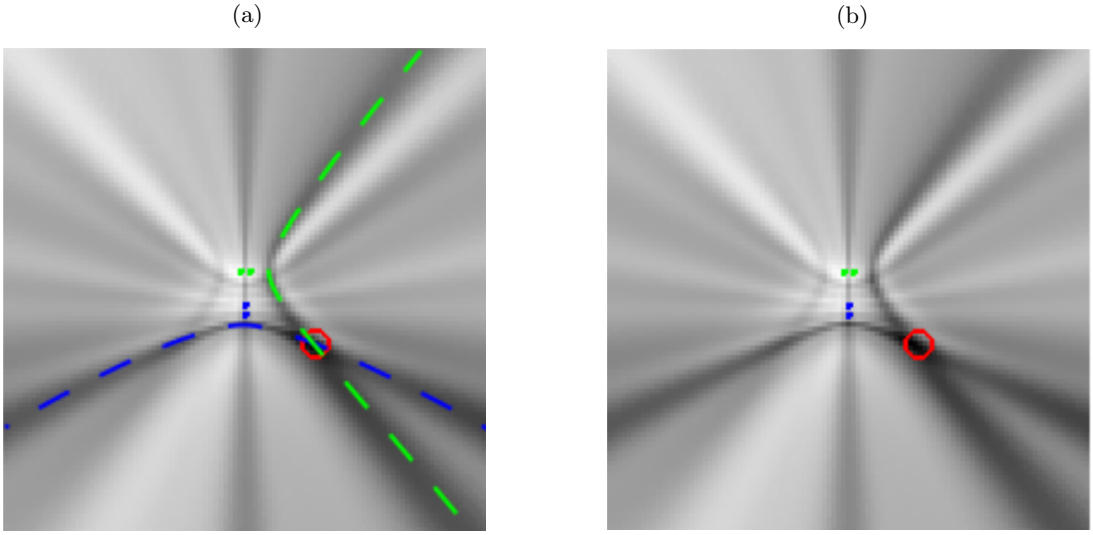

Figure 2.4: Example of an acoustic map for a source s and two given microphone pairs. (a) Difference between the ideal hyperbolas, (b) Estimation of acoustic source position.

The maximization of the beamforming function can be based on different strategies. The most popular method is to uniformly sample the beamforming function where each point of the grid corresponds to a position where the beamforming is steered at. The source position is then obtain by a simple search mechanism. This method is the simplest to implement, but also the one with the highest computational cost and its accuracy directly depends on the grid size. Another widely used strategy is the so-called Stochastic Region Contraction (SRC) [25], where several random points are defined within a search region. Depending on where the greatest power is located, the region is reduced iteratively until finding the acoustic source. This search method is fast, but its convergence to an accurate solution is not guaranteed.

2.1.2.1 Steered Response Power

The SRP technique [13, 16, 17, 26], which is the simplest beamforming method, uses the signal power received when the microphone array is steered in the direction of a specific location. This acoustic power at one point is calculated as the sum of the GCC value of the signals along all the microphones, according

to the TDOA existing between them. SRP-PHAT[1] is a widely used algorithm for speaker localization based on SRP. It was first proposed as such in [9], and is a beamforming based method which combines the robustness of the steered beamforming methods with the insensitivity to signal conditions afforded by the PHAT filtering, which is used to weight the incoming signals. The advantage of using PHAT [22] is its resistance to reverberation and room conditions [27].

Following [28], the value of the acoustic power at point q, according to the SRP-PHAT beamforming, is defined by the expression

$$srp(q,r,m) = \sum_{n=1}^{N}\sum_{m=1}^{N} \mathcal{F}^{-1}\left\{\frac{R_n\ R_m^*}{\mid R_n \mid \mid R_m^* \mid}\right\}(\Delta\tau(q_k, m_n, m_m)), \tag{2.10}$$

where R_n and R_m are the received signals at the microphones m_n and m_m in the frequency domain. The the operator $(\cdot)^*$ means the conjugated of $(\cdot)$ and thus the term $R_nR_m^*$ is the GCC function in frequency domain of the signals arriving at the microphones. $\Delta\tau(q_k, m_n, m_m)$ denotes the TDOA between the position q_k and the microphones m_n and m_m. The expression $(\mid R_{m_j} \mid \mid R_{m'_j}^* \mid)^{-1}$ corresponds to the PHAT filtering used as a weighting function by the SRP algorithm, and it is usually applied so that the algorithm only uses the phase information to determine the location of the source [10, 18, 29, 30].

2.1.2.2 Minimum Variance Distorsionless Response

The MVDR beamformer [20, 31, 32] is a well-known beamforming tecnique which is aimed at minimizing the energy of noise and sources coming from different directions, while keeping a fixed gain on the desired position. This method is more robust against noise if compared to other algorithims as SRP, but in the other hand, it exhibits problems when facing reverberant environments. The MVDR beamforming function is defined as

$$mvdr(q,r,m) = \omega(q,m)f(q,r,m), \tag{2.11}$$

where $\omega(q,m)$ is the weighting function and $f(q,r,m)$ is the beamformer itself. As in the case of SRP, the best performance is obtained when the GCC function is used as $f(q,r,m)$. On the other hand, $\omega(q,m)$ is a weighting function which, in addition, is capable of minimizing noise with respect to the signal that contains the information.

As shown in [31], the optimal value of the weighting function to guarantee the minimization of the noise in the signal, is that which satisfies that $\omega(q,m)$ is an steering vector of the form $\omega(q,m) = \left(\frac{\|m_m - q\|}{\|m_n - q\|}\right)^2$, being the $\|\cdot\|$ operator the Euclidean norm. According to the above, the expression 2.11 of the MVDR beamformer can be rewritten as

$$mvdr(q,r,m) = \sum_{n=1}^{N}\sum_{m=1}^{N} \mathcal{F}^{-1}\left\{\left(\frac{\|m_m - q\|}{\|m_n - q\|}\right)^2 (R_n\ R_m^*)\right\} / \Delta\tau(q, m_n, m_m)). \tag{2.12}$$

Comparing SRP-PHAT with MVDR, the former is more robust to reverberation while the latter has more immunity to environmental noise.

2.1.3 High-resolution spetral-estimation

Methods based on spectral estimation of the signal, like the popular Multiple Signal Classification (MUSIC) algorithm [33–36], exploit the spectral decomposition of the covariance matrix of the incoming signals to improve the spatial resolution of the algorithm in the case of multiple sources. The basic idea

[1] From now on, SRP-PHAT and SRP will be used interchangeably with the same meaning

of this algorithm is to find the steering vector orthogonal to the null space of the spatial spectral density matrix of the signals recorded by the Nm microphones. Following equation 2.8, the self-correlation of the received signal R_n is defined as

$$C_r = E[R_n\, R_n^H] = \sum_r (\lambda_r V_r V_r^H), \tag{2.13}$$

where λ_r is the r^{th} eigenvalue and V_r is its corresponding eigenvector. The MUSIC spectral density value at position q is given by the following expression

$$S_{MUSIC}(q) = \frac{1}{a^H(q)\left(\sum_{r=Ns}^{Nm} V_r V_r^H\right) a(q)}. \tag{2.14}$$

In order to accurately know the spectral density value at a given position, a steering vector $a(q)$ is needed of the form $a(q) = \left(1, e^{-j\frac{\omega_s q}{cN}}, e^{-j\frac{2\omega_s q}{cN}}, \cdots, e^{-j\frac{(N-1)\omega_s q}{cN}}\right)$, with ω_s as the sampling pulsation, c as the sound propagation velocity and N as the total length of the recorded signal. According to this, an steering power map can be built in a similar way as SRP map. However, unlike SRP and similar approaches, MUSIC assumes the signals to be incoherent. Unfortunately, in real scenarios with speech sources and reverberation effects, the incoherence condition is not fulfilled, making the subspace-based techniques problematic in practice.

2.2 Learning-based methods

In the past few years, deep learning approaches [37] have taken the lead in different signal processing and machine learning fields, such as computer vision [38,39] and speech recognition [40–42], and, in general, in any area in which complex relationships between observed signals and the underlying processes generating them need to be discovered.

The idea of using neural networks for ASL is not new. Since the early nineties, works such as [43–45] proposed the use of neural network techniques in this area. However an evaluation on realistic and extensive data sets was not viable at this time, and the proposals were somehow limited in scope. With the advent and huge increase on applications of deep neural networks in all areas of science, and mainly due to the sophisticated capabilities and more careful implementation details of network architectures and the availability of advanced hardware architectures with increased computational capacity, promising works have been proposed also for ASL [2,21,46–51]. The main differences between the different proposals using neural networks for ASL reside in the architectures, input features, the network output (target), and the experimental setup (using real or simulated data).

In the following sub-sections the bases of the Artificial Neural Networks (ANNs) will be explained, going into detail in the Convolutional Neural Networks (CNNs), as well as some applications for the task of ASL that make use of them.

2.2.1 Fundamentals of neural networks

The operation of a classical neuron is shown in figure 2.5. It takes an input vector and perform the weighted sum of its components, adding a bias to the result. The result is passed through a non-linear function or activation function. The parameters of the neuron are the weights of the weighted sum and the bias parameter. There are different types of activation functions such as the sigmoidal, hyperbolic

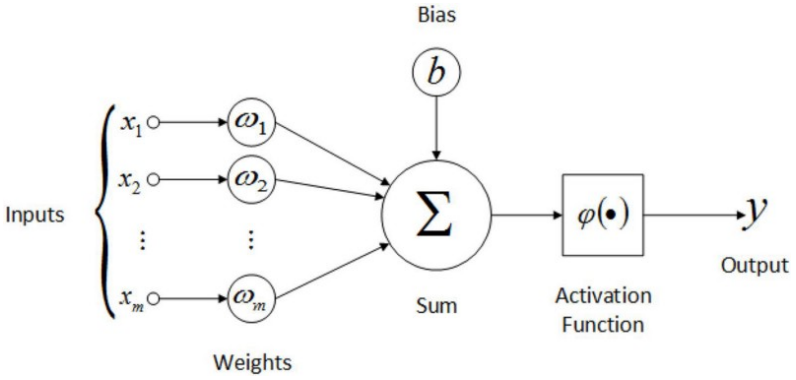

Figure 2.5: Scheme of operation of an artificial neuron.

tangential, ReLU, LeakyRelu or softmax functions, but without a doubt the most common today, within the neural networks, is the ReLU acivation. Figure 2.6 shows some of the most common activations.

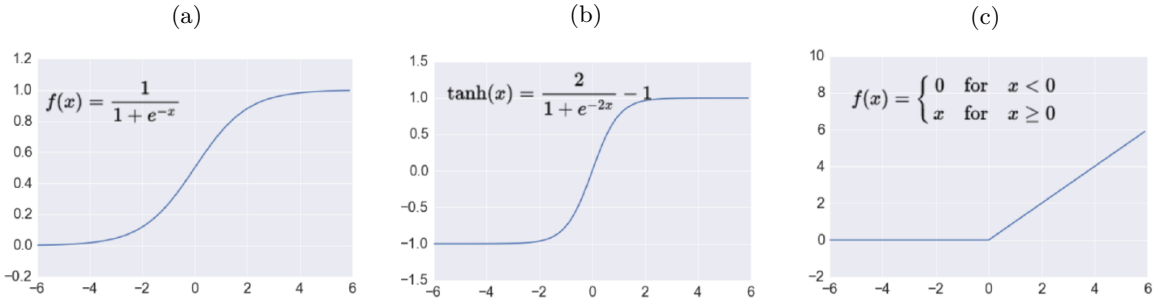

Figure 2.6: Examples of activation functions, sigmoidal (a), hiperbolic tangential (b) and ReLU (c).

An ANN is formed by connecting multiple neurons, all at different levels or layers, with a predefined number of neurons per layer. Likewise, there are three main types of layers based on the function they perform.

- **Input layer:** they are used to receive the input data, which is usually normalized.
- **Hidden layers:** they are located between the input layer and the output layer. There is no fixed number of these layers in a neural network, so they can vary between no layer up to a large number of layers. They are formed by different neurons, all of them interconnected. The connection between neurons is the topology of the network.
- **Output layer:** provides the data corresponding to the solution estimated by the network, which has been processed by the previous layers.

In figure 2.7 a possible scheme of an ANN is shown as well as where the three types of layers mentioned above would be located. As has been said, each of the layers that form the neural network are composed of a set of neurons. The most appropriate number for each case is not a trivial decision, since the combination of hundreds or millions of them can make it easier to solve more complex problems, but it also entails the need for more processing time and resources to obtain more solution.

Once defined the basic functions that are developed in the ANN, it is important to mention the concept of training. This process is used to adjust the weights of each of the neurons that forms the network so that it performs a specific function. In the case of supervised learning, it is necessary to define a data set that is representative of the problem to be solved, which is called the training set. This set

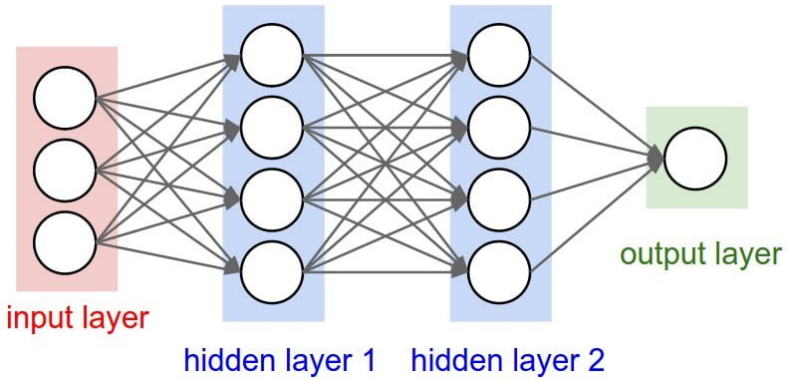

Figure 2.7: Scheme of a neural network.

can be of any nature (image, audio, ...). All the data used for training must be accompanied by its target outputs the network has to reproduce.

One of the most important concepts when training a neural network is the loss function (mean squared error, mean absolute error, binary cross entropy, ...) that calculates the error that exists between the output estimated by the network and the targets associated with the input data. The learning algorithm is based on minimizing the loss function in terms of the network weights. The most common optimization strategy is based on gradient descent via back propagation [52, 53].

2.2.2 Convolutional neuronal networks

The CNNs have acquired in the last years great importance in several fields of science. These networks, whose origin dates back to 1989, differ from the classic networks in the inclusion of layers of a particular type of neuron, whose mathematical function is based on discrete convolution with a set of Finite Impulse Response (FIR) filters.

Convolution operations are especially well suited for processing signals with different temporal or spatial dimensions, such as audio signals or images. In addition, and in comparison with traditional networks, the number of parameters required by convolutional layers does not grow with the temporal dimensions of the input. The restricted number of parameters and convolution characteristics contribute to the construction of deep neural networks that operate more easily with signals of any dimension.

Figure 2.8 shows the differences in the layers used with respect to the classic networks, shown in figure 2.7. In this case, the CNNs are mainly formed by three different types of layers: the convolutional layers, the pooling layers and the fully connected layers, which are detailed below.

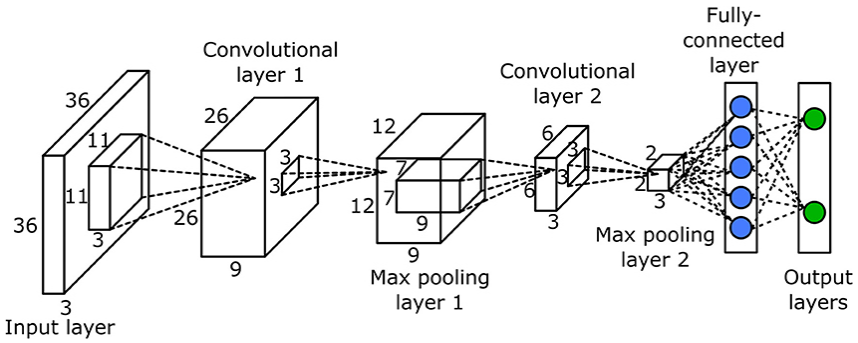

Figure 2.8: Scheme of a convolutional neural network.

2.2.2.1 Convolutional layer

This layer is fundamental within the CNNs. In these layers, the input is composed of a tensor where temporal dimensions (duration of the audio sample) or spatial (height and width of an image) and the depth dimension (channels) are distinguished. The filters that are defined in those layers consist of kernels of a size generally lower than the one of the input which match in terms of the depth dimension. The output depth of the convolutional layer coincides with the number of filters. As a simple example, given an image, represented by the two-dimensional matrix $\mathbf{I}$ and a $\mathbf{K}$ kernel with dimensions N and M, the convolution operation is expressed as

$$\mathbf{S}(i,j) = \sum_{\forall m \in M} \sum_{\forall n \in N} \mathbf{I}(i-m, j-n)\, \mathbf{K}(m,n). \tag{2.15}$$

This example is graphically represented in figure 2.9. Convolutional layers have a lower number of

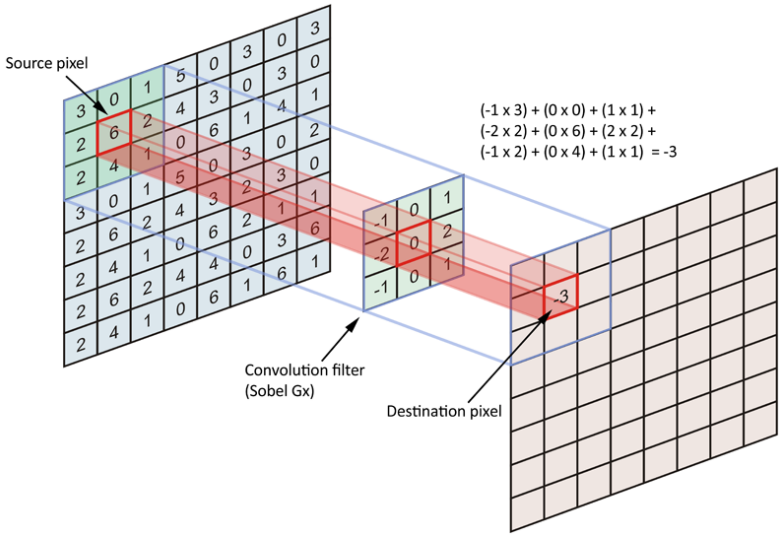

Figure 2.9: Convolution layer example.

parameters than those required by classical neurons for the same input and output dimensions. In addition these layers learn characteristics of the input data regardless of the time or place in which they appear in the input. This idea arises from the fact that if a specific characteristic is exploited to give a type of output, this characteristic can always be used regardless of where it occurs.

Finally, it is important to detail the important hyperparameters which are responsible for determining the output volume of the layer:

- **Depth:** it represents the number of filters used in that layer. Normally this paramenter is represented by D.
- **Filter size:** it defines the dimensions of the filter used to perform the convolution. In the case of images, the filters are two-dimensional, while in the case of audio they are usually one-dimensional. This hyperparameter is represented as F.
- **Stride:** it specifies the displacement that the filter used has to make throughout the input samples. If the stride takes the value of $S = 1$, this implies that the filter slides sample to sample along the entire input, if it had a value of $S = 2$, the filter would be applied every two samples and so on for each stride value.

- **Zero-padding:** it is used to fill the input data with zeros. This technique allows the convolutional layer to control input and output dimensions. It is usually represented by P.

Knowing the dimensions of the input data and the value of the aforementioned hyperparameters, the output data dimension W_{out} are given by the expression

$$W_{out} = \frac{W_{in} - F - 2P}{S} + 1, \tag{2.16}$$

where W_{in} represents the input data dimensions.

2.2.2.2 Pooling layer

The pooling layers perform various types of operations (maximum, average, L2 norm, ...), by which a subset of the input data of the layer is passed to a single value. This layer operates in the manner shown in figure 2.10, where it slides along the input data and performs the operation for the selected set (in this case, obtain the maximum) and enters it into the output data of the layer. This layer has only

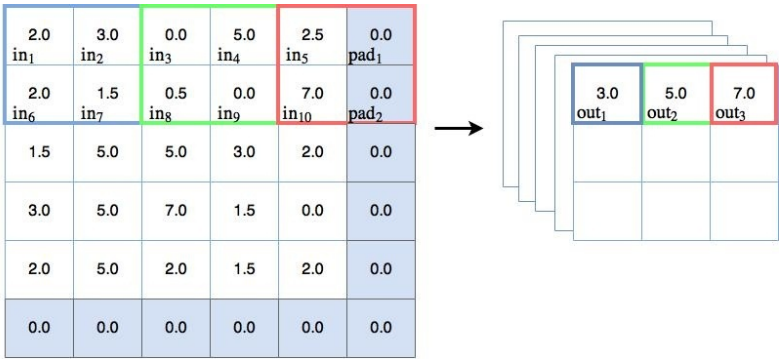

Figure 2.10: Max-pooling layer example.

the size of the pooling (equivalent to the filter size of the convolutional layer) and the applied stride as hyperparameters. The depth of the exit remains the same as that of the entrance and the output dimensions are defined as

$$W_{out} = \frac{W_{in} - F}{S} + 1, \tag{2.17}$$

In which W_{in} and W_{out} represent the dimensions of the input and output data of the layer respectively.

2.2.2.3 Fully connected layer

This type of layers perform an operation similar to the layers of classical neurons. They are characterized mainly by having connections to each and every activation of the previous layer. Figure 2.11 shows a graphical example of a fully connected layer. There are other types of layers that are used when designing the architecture of a network and we have only covered a few of them.

2.2.3 Tipical optimizers

The training of a CNN is based on updating the model weights proportionally to a learning rate by computing its gradient according to a loss function (*Mean Absolute Error*, *Mean Square Error*, *Cross Entropy*, ...) that relates the error committed between the output data and the desired one, and then back-propagate it from the last layer to the previous ones until reach the first layer of the model. This

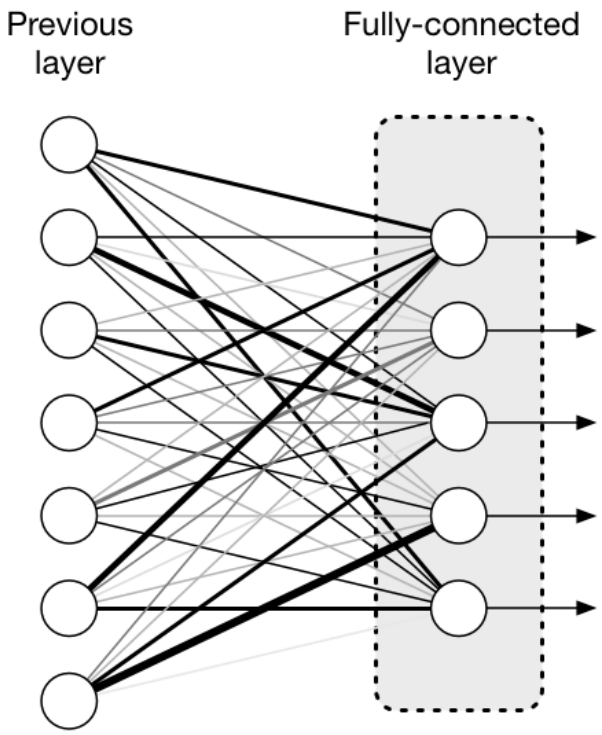

Figure 2.11: Fully conected layer example.

process is handled by what is known as optimizer and in the state-of-the-art there are several of them. In this subsection only the Stochastic Gradient Descent (SGD) and the ADAM optimizer are described since they are the most used by the scientiic community.

2.2.3.1 SGD

The SGD optimizer [52] is the simplest one in the state-of-the-art and is based on updating the model weigths of the iteration t according to the gradient of used loss function $\mathcal{L}(\Theta_i)$ for the i^{th} layer of the model. As an example, using the *Squared Error* as the loss function, the gradient of the i^{th} layer will be computed as $\nabla\mathcal{L}(\Theta_i) = \nabla(\Theta_i^\top x_i - y_i)^2$ where $\nabla(\cdot)$ is the gradient operator, y_i is the desired output data of the i^{th} layer, Θ_i are the weights of the i^{th} layer and x_i is the input data, so $\Theta_i^\top x_i$ defines the estimated output data of the i^{th} layer. The simplified updating weight process of the SGD optimizer is defined in equation 2.18 with η as a fixed learning rate, t the actual iteration and $t-1$ the previous one.

$$\Theta_{t,i} = \Theta_{t-1,i} - \eta\,\nabla\mathcal{L}(\Theta_{t-1,i}) \tag{2.18}$$

The value of the fixed learning rate is a critical hyperparameter of the model due to a large value of it can make the model to not converge properly and a short value can make it to converge in a local minimum, being this a non-optimal solution. Althougth this is a robust method and widly used by the scientific community, it has its inconvenients. It is a slow algorithm in terms of convergence and has and strong dependence on the learning rate.

2.2.3.2 ADAM

The *ADAM* optimizer [53] is the natural evolution of the SGD optimizer. As it is commented above, one of the inconvenients of the SGD algorithm is the dependence on the learning rate value. The following

expression defines the *ADAM* algorithm

$$\Theta_{t,i} = \Theta_{t-1,i} - \eta_t f\left(\mathcal{L}(\Theta_{t,i})\right), \tag{2.19}$$

which is identical to the equation 2.18 with the exception of the η parameter and how they compute the loss function gradient ($f\left(\mathcal{L}(\Theta_i)\right)$). The *ADAM* optimizer dynamically updates the learning rate at each iteration. The process of updating the learning rate is described in equation 2.20 where $\beta_1, \beta_2 \in [0, 1)$ are two hyperparameters. Usual values are $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

$$\eta_t = \eta_0 \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t} \tag{2.20}$$

In *ADAM* the first and the second moment of the loss function are computed and then used to update the weights. This function is defined in equation 2.21 where $m_{t,i}$ is the first moment of the loss function at the t^{th} iteration of the i^{th} layer, $v_{t,i}$ is the second moment of the loss function at the t^{th} iteration of the i^{th} layer.

$$\begin{cases} m_{t,i} = \beta_1\, m_{t-1,i} + (1-\beta_1)\, \nabla\mathcal{L}(\Theta_{t,i}) \\ v_{t,i} = \beta_2\, v_{t-1,i} + (1-\beta_2)\, \nabla^2\mathcal{L}(\Theta_{t,i}) \\ f\left(\mathcal{L}(\Theta_{t,i})\right) = \frac{m_{t,i}}{\sqrt{v_{t,i}}} \end{cases} \tag{2.21}$$

The *ADAM* algorithm is well suited for problems with a large amount of data or parameters and is also suitable for non-stationary problems due to its adaptive condition. It is also faster in terms of convergence than SGD.

Chapter 3

Proposed System

Believe and act as if it were impossible to fail.

Goethe

3.1 Introduction

This chapter describes the proposed solution to the ASL problem using deep learning methods, the databases used for training and the training strategy.

3.2 Problem Statement

The proposed system obtains the position of an acoustic source from the audio signals recorded by a set of arrays of microphones formed entirely by M microphones. Given a coordinate origin as a reference, the position of the source is defined by the three-dimensional coordinate vector $s = (x_s, y_s, z_s)^\top$. The positions of the microphones are known and are defined by the coordinate vector $m_i = (x_i, y_i, z_i)^\top$ with $i = 1, \ldots, M$. The audio signal captured by the i^{th} microphone is denoted by $r_i(t)$. The signal is discretized with a sampling frequency f_s, which is defined as $r_i[n]$. For simplicity it is assumed that $r_i[n]$ is composed of N samples. This corresponds to an audio window of duration $W = \frac{N}{f_s}$, which is one of the system's hyper-parameters.

The problem to be solved is to find the following regression function f:

$$\hat{s} = f(r_1, r_2, \ldots, r_M, m_1, m_2, \ldots, m_M), \tag{3.1}$$

that obtains the position of the speaker given the signals recorded by the microphones, where $\hat{s}$ is the estimation of the speaker position and $r_i = (r_i[0], r_i[1], \ldots, r_i[N-1])^\top$ is the vector with all the samples recorded by the i^{th} microphone. In classical approaches, f is found assuming that the signals received by the different microphones principally differ by a delay that depends on the relative position of the source with respect to the microphones. In any case, this assumption is not met in environments where the signal suffers from random noise and distortions, such as multi-path or reverberations. Due to the aforementioned effects, and the random nature of the audio signal, the regression function of the equation 3.1 can not be estimated analytically, so this work presents a learning approach to obtain directly f using deep learning tecniques. f can be represented using a CNN which is learned end-to-end from the signals

of the microphones. In the system it is assumed that the position of the microphones is fixed, so the condition of knowing them from equation 3.1 will implicitly be learned by the network by means of the following regression problem:

$$\hat{s} = f_{net}(r_1, r_2, \ldots, r_M), \tag{3.2}$$

where f_{net} denotes the function to be represented using the CNN, whose topology is described in next section.

3.3 Network topology

The topology of the proposed neural network is shown in figure 3.1. This network follows the structure of a five-layer convolutional encoder connected to a fully-connected blocks. According to the equation 3.2, the inputs of the network are the set of the windowed signals of the microphones and the output of the network corresponds with the estimated position of the acoustic source. Due to the purpose of the network to perform an acoustic source localization, this will be called from now on as ASLNet (Acoustic Source Localization Network). Table 3.1 shows the number of filters used in the proposed topology, as

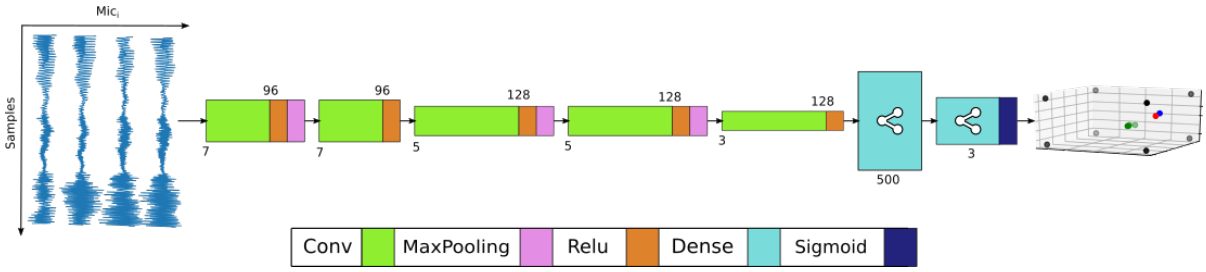

Figure 3.1: ASLNet topology.

well as their size. For the first two layers, 96 filters of size 7 are used. For the rest of the layers 128 filters are used in each layer, where in layers 3 and 4 each filter is of size 5 and in the last one they are size 3. As shown in figure 3.1, some of the layers are followed by a *max-pooling* layer with the same pool size as the filters of its corresponding convolutional layer. The last two layers are *fully-connected*, the first of which is a hidden layer with 500 nodes and the second is an output layer with 3 nodes, corresponding to the x, y and z coordinates provided by the network. All the convolutional layers are followed by a *ReLU* activation whereas activation function of the output layer is a sigmoid function. During training, the *fully-connected* hidden layer has a dropout factor of 50% to prevent overfitting. The size of the input

Layer	# Filters	Filter Size
Convolutional Layer 1	96	7
Convolutional Layer 2	96	7
Convolutional Layer 3	128	5
Convolutional Layer 4	128	5
Convolutional Layer 5	128	3

Table 3.1: Used convolutional layers summary.

varies depending on the time window used (80 ms, 160 ms or 320 ms). Therefore, three different models are generated where the only change is the dimensions of the input layer, leaving the rest of the layers unchanged.

3.4 Datasets

In the proposed work, two databases have been used. The first of them consists of a series of recordings inside an anechoic room and will be used to generate a semi-synthetic dataset. The second database is used both for testing and for fine-tuning and it is recorded in a realistic indoor environment. Both databases are explained below in more detail.

3.4.1 The semi-synthetic dataset: *Albayzin Phonetic Corpus*

The *Albayzin Phonetic Corpus* [54] consists of 3 sub-corpora of 16-bit signals recorded at 16 kHz by 204 Spanish speakers in a professional recording studio using high quality near-speech microphones. This dataset is used to generate semi-synthetic data as described in section 3.5.1.1. Of the 3 sub-corpora, only the so-called phonetic corpus [6] is used, consisting of 6800 fully balanced audio files, where each of the 204 different voices appear. Of all those files, 4800 (164 balanced voices) are intended for training while 2000 (40 balanced voices) are used as validation as is shown in table 3.2. The characteristic of the fonetical balance converts this dataset into the ideal one to generate the semi-synthetic data.

	# Files		**# Voices**	
	Train	**Validation**	**Train**	**Validation**
Female	2400	1000	82	20
Male	2400	1000	82	20
Total	4800	2000	164	40

Table 3.2: *Albayzin phonetic corpus* dataset summary.

3.4.2 The real dataset: IDIAP AV16.3 Corpus

The *IDIAP AV16.3* [1] dataset is an audio-visual corpus recorded in the *Smart Meeting Room* of the *IDIAP* research institute in Switzerland. This database is composed of several audio and video sequences with a wide range of experimental conditions and sampled synchronously at 16 kHz. The sequences are assigned an annotation file that contains the real positions (3D coordinates) of the speaker mouth, which will be taken as ground truth at the time of testing or performing the fine-tunning. The main characteristics of the sequences used are shown in table 3.3.

As mentioned before, all the sequences used have been recorded in *IDIAP Smart Meeting Room*, which according to [55] is a rectangular room of dimensions $3.6m \times 8.2m \times 2.4m$ with a rectangular table placed in the center of it with dimensions $4.8m \times 1.2m \times 0.73m$. On the surface of the table there are two circular arrays of microphones of 0.1m radius. Each of them is composed of 8 microphones uniformly distributed, as shown in figure 3.2c. The centers of both arrays are separated by a distance of 0.8m and the midpoint between them is considered as the origin of system coordinates. The layout of the room shown in figure 3.2 is taken into account not only when generating the semi-synthetic dataset detailed in section 3.5.1.1, but also when it comes to representing the results. For all the experiments carried out in this thesis, we use the microphone set up shown in figure 3.2c.

[1] Height refered to the speaker mouth.
[2] For windows of 320 ms, 160 ms and 80 ms respectively.

Sequence	Average speaker height (cm) [1]	Duration (seconds)	Number of GT frames [2]	Description
seq01-1p-0000	54.3	208	279/556/1113	A single indian male speaker, static while speaking, at each of 16 locations. The speaker is facing the microphone arrays
seq02-1p-0000	62.5	171	301/601/1202	A single european female speaker, static while speaking, at each of 16 locations. The speaker is facing the microphone arrays.
seq03-1p-0000	70.03	220	325/650/1300	A single european male speaker, static while speaking, at each of 16 locations. The speaker is facing the microphone arrays.
seq11-1p-0100	53.5	33	542/1083/2165	A single european male speaker, making random movements while speaking, and facing the arrays.
seq15-1p-0100	79.5	36	491/981/1962	A single european male speaker, walking around while alternating speech and long silences. No constraints.

Table 3.3: *IDIAP AV16.3 corpus* used sub-dataset summary.

3.5 Training strategy

In the state-of-the-art there are very few databases with exact labeled locations and those that do exist have very little data available. Normally the DOA or elevation and azimuth of the acoustic source is labeled. Due to this, one of the main objectives of this line of research is to achieve a CNN based method that is able to estimate the position of the acoustic source having been trained only with semi-synthetic data, that is, with audio sequences that have not been recorded in the room in which the method is being applied.

In the specific case of this work, the amount of data available in *IDIAP*'s *AV16.3* database [1] to properly train the CNN model is limited. To solve this problem it has been decided to carry out two different training strategies. The first one is based on a two-phase training, where the following two steps are defined:

- **Step 1**: The CNN model is trained with semi-synthetic data. The speech recordings of the *Albayzin Corpus* dataset [54] are used to generate simulated acoustic sources in a random position within a virtual environment that corresponds to the geometry of the *IDIAP Smart Meeting Room* [55]. The acoustic characteristics of the environment (specific types of noise, noise levels, etc.) have also been taken into account when generating the semi-synthetic dataset.
- **Step 2**: The CNN model is fine-tuned with data recorded in a realistic environment. Starting from the model weights obtained in step 1 as initialization, the network is trained with a reduced set of data from the *IDIAP AV16.3* database [1].

The second proposed training strategy is to train the network only with semi-synthetic data in order not to depend on any real database. To do this, the Albayzin Corpus dataset is again used to generate the virtual signals recorded by the microphones. In this case, the generation process will be more elaborated than in the first strategy.

(a) (b) (c)

Figure 3.2: *IDIAP Smart Meeting Room* Layout. (a) Simplified top view of the *IDIAP Smart Meeting Room*, (b) A real picture of the room extracted from a video frame, (c) Microphone setup used in this proposal.

The loss function used to train the network, in any of the two defined strategies, is the mean square error between the real 3D position labeled (s_i) and the estimate given by the CNN model ($\hat{s_i}$), following the expression:

$$\mathcal{L}(\Theta) = \frac{1}{N_s} \sum_{i=0}^{Ns-1} \mid \hat{s_i} - s_i \mid^2, \tag{3.3}$$

where Θ represents the weights of the network. The equation 3.3 is minimized as a function of unknown weights using iterative optimization based on the SGD algorithm [56]. The solution weights ($\theta \in \Theta$) are finally obtained once a stop criterion is reached in the optimization.

3.5.1 Training strategy 1

As mentioned brefore, the first proposed training strategy consists of two phases. A first phase based on semi-synthetic data followed by a fine-tuning stage using real data.

3.5.1.1 Semi-Synthetic Dataset Generation

In this training phase, the audio signals of the *Albayzin Phonetic Corpus* [54] are used to generate a large semi-synthetic dataset. Note that many other phonetic corpora are available for this task (distributed free or commercially). With this method we generate N_s random position vectors uniformly distributed to cover all the physical space of the room. Each generated position is associated with an audio signal, which is synthetically propagated to each of the microphones. In order to realistically simulate the signals received in the microphones from a given source position, two main issues have to be considered:

- **Signal propagation considerations**: this is affected by the impulse response of the target room. Different alternatives can be used to simulate this effect, such as convolving the anechoic signals with real room impulse responses, such as in [47], which can be difficult to acquire for general positions in big environments, or by using room response simulation methods, such as the image method [57] used in [18] for this purpose.
- **The acoustic noise conditions of the room and the recording process conditions**: these can result from additional equipment (computers, fans, air conditioning systems, etc.) present in the room, and from problems in the signal acquisition setup. They can be addressed by assuming additive noise conditions and selecting the noise type and acoustic effects that should be preferably estimated in the target room.

With respect to the first issue we use a simple model delaying the signals according to the distance between the source and the microphones. We use a sub-sample delay algorithm based on phase shifting the Fourier Transform of each signal. This simulation model does not consider other environmental effects such as reverberations and multi-path echos. Due to this, it is not necessary to know any specific characteristics of the room (position of the walls, materials of them, furniture, etc) except for the position where the microphones are located. The process described above for the signal received at the i^{th} microphone is defined as follows:

$$\begin{cases} r_i = g_i \, \mathcal{F}^{-1}\{X \odot D_i\} \\ D_i = \left(1, e^{-j\frac{\omega_s d_i}{cN}}, e^{-j\frac{2\omega_s d_i}{cN}}, \cdots, e^{-j\frac{(N-1)\omega_s d_i}{cN}}\right), \end{cases} \tag{3.4}$$

where $i \in [0, N_m - 1]$, $\mathcal{F}^{-1}$ denotes the Inverse Fast Fourier Transform (IFFT), $\odot$ operator is the element-wise product operator, $\omega_s = 2\pi f_s$ being f_s the sample frequency used to record the signals, c is the sound propagation velocity ($343\frac{m}{s}$ at $20°C$), d_i is the Euclidean distance between the acoustic source and the i^{th} microphone, N is the signal total length and $g_i \in [0.01, 0.03]$ is a gain factor applied to the i^{th} microphone signal. Factor g_i creates random gain fluctuations between the microphones. This is important so that the network learns to focus its attention at the phase information of the signals rather than the amplitude. It was intended that these random amplitudes would also help to tackle the effects of the non-isotropic directionality of the microphones in the real case.

Regarding the second issue, the noise and disturbances in the signals arriving to the microphones have been simulated so that the signal-to-noise ratio and the spectral content of the signals were as similar as possible to those found in the real data. In the *IDIAP Smart Meeting Room*, a spectrogram-based analysis showed that the recordings were contaminated with a tone at around $25Hz$ in the spectrum which does not appear in anechoic conditions. This was probably caused by some electrical noise captured by the recording hardware. The frequency of this tone actually varied in a range between $20Hz$ and $30Hz$.

Therefore, the signals has been contaminated with an additive tone of a random frequency in this established range, and they also have been contaminated with white Gaussian noise:

$$r_{i_\eta} = r_i + k_s \, sin(\frac{2\pi f_o}{f_s} n + \phi_o) + k_\eta \, \eta_{wgn}, \tag{3.5}$$

where k_s is a scaling factor for the contaminating tone signal (similar to the tone amplitude found in the target room recordings, 0.1 in the specific case), $f_o \in [20, 30]\, Hz$, $\phi_o \in [0, \pi]\, rad$, η_{wgn} is a white Gaussian noise signal, k_η is a noise scaling factor to generate signals with a similar Signal to Noise Ratio (SNR) to that found in the target room recordings ($SNR \in [1, 30]\, dB$) and $n \in [0, N-1]]$ is the sample index of the microphone signal. After this procedure was applied, signals similar to the shown in figure 3.3 are generated and the semi-synthetic signal data set is ready to be used in the neural network training procedure.

3.5.1.2 Fine Tuning Procedure

The previous step took care of reproducing the simple acoustic characteristics of the microphone array configuration and the presence of specific types and levels of additive noises, but there are other phenomena, like the presence of diffuse noise and multi-path propagation and reverberation, which are more complex to simulate as it is said before. In order to introduce these acoustic characteristics from the target physical environment, we carried out a fine tuning procedure on the network model, using a short amount of real recorded data in the target room.

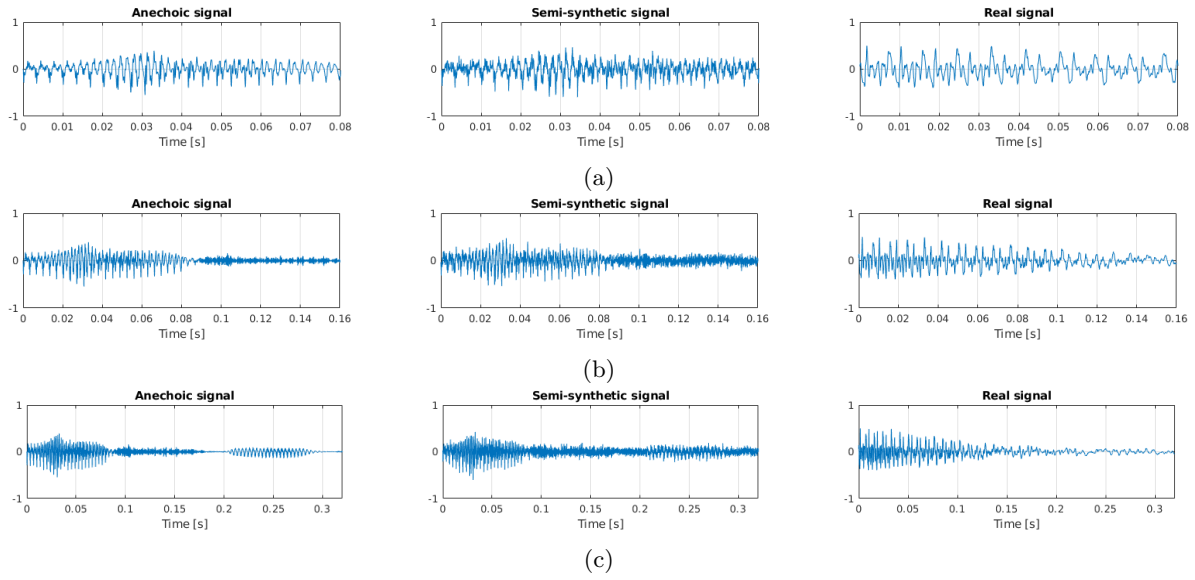

Figure 3.3: Comparison of the selected anechoic signal, the semi-synthetic generated signal and a real signal captured by a microphone for the temporal window sizes (a) 80 ms, (b) 160 ms and (c) 320 ms.

The fine tuning procedure consisted of training the network with some of the sequences recorded in the real environment after initializing the network weights obtained from the training phase with the semi-synthetic data. This is now a standard procedure in the deep learning literature.

3.5.2 Training strategy 2

The second strategy tries to remove the fine-tunning step in order to depend as little as possible on real labeled data, which involves a high cost. It also allows any geometric configuration to be trained using synthetic data. To accomplish this objective the generation of the semi-synthetic dataset has to be more realistic in terms of the signal propagation considerations than the one explained in section 3.5.1.1. Several data generation methods are proposed, which can be used in a complementary way with each other.

3.5.2.1 NOISE method

The noise generated so far has been white gaussian noise. However, this type of noise is not a realistic noise, since the background noise that appears in a recording can contain information about the environment in which the microphones are located as well as their response. This is why we can expect that by introducing realistic noise in the generation of semi-synthetic data, the performance of the *ASLNet* model will improve.

To correctly add the real noise to the anechoic signal, firstly, noise sequences have been generated by selecting fragments of the sequences not used for tests where only background noise appears. Randomly, fragments of noise sequences are selected and added to the anechoic signal so that a specific SNR is met. Starting from equation 3.4, this procedure is defined as

$$r_{i_{NOISE}} = r_i + k_\eta * \eta_{real}, \tag{3.6}$$

where η_{real} is the background noise fragment extracted sequence and k_η is a multiplicative constant that ensures a $SNR \in [1, 30]\, dB$. An example of the signals that are generated with this method is shown in figure 3.4 where the result is also compared with a similar real signal.

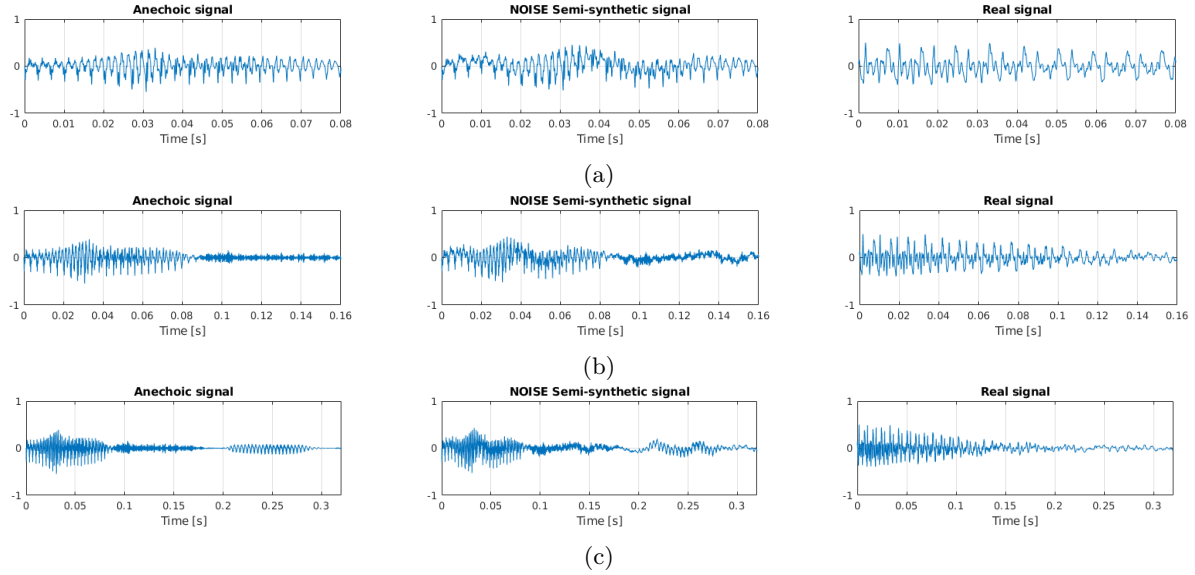

Figure 3.4: Comparison of the selected anechoic signal, the semi-synthetic generated signal and a real signal captured by a microphone for the temporal window sizes (a) 80 ms, (b) 160 ms and (c) 320 ms.

3.5.2.2 ECHO method

Reverberations that occur within a closed environment are one of the main acoustic effects that affect the performance of ASL methods. This phenomenon can be precisely simulated by using geometric algorithms, such as the mirror source [58], or by RIR as in method [59]. However, this process is dependent on the structure of the room where the acoustic signal is propagated, which is far from our objective of not depending on a specific room.

Our proposal is to generate data so that the *ASLNet* network learns to obviate the reverberations. To achieve this goal, we propose to include several echoes at different instants of time in a random manner within a specific time frame, so that the virtual distance that the simulated acoustic wave travels is within realistic ranges. The equation that is used to implement this method is

$$r_{i_{ECHO}} = r_i + \sum_{e=1}^{N_E} \frac{1}{c\,\tau_{e_{rnd}}} \mathcal{F}^{-1}\left(R_i[n]\, e^{-j\frac{\omega_s\, \tau_{e_{rnd}}\, n}{N}}\right), \tag{3.7}$$

where $N_E \in [3, 5]$ is the number of echoes added to the i^{th} microphone, $\tau_{e_{rnd}} \in [6, 60]\, ms$ is the random delay that suffers the e^{th} echo which is associated to a propagatión distance $c\,\tau_{e_{rnd}} \in [2, 20]\, m$ with $c = 343 m/s$ (at 20^oC), R_i is the signal recorded by the i^{th} microphone in the Fourier space, ω_s is the sampling pulsation and N is the total length of the signal.

Finally, an example of a simulated signal next to a real signal is shown in figure 3.5.

3.5.2.3 PHASE method

Another effect to take into account is the phase distortion that occurs in the signal as it propagates through the environment. Also keep in mind that this phase distortion is more pronounced the higher

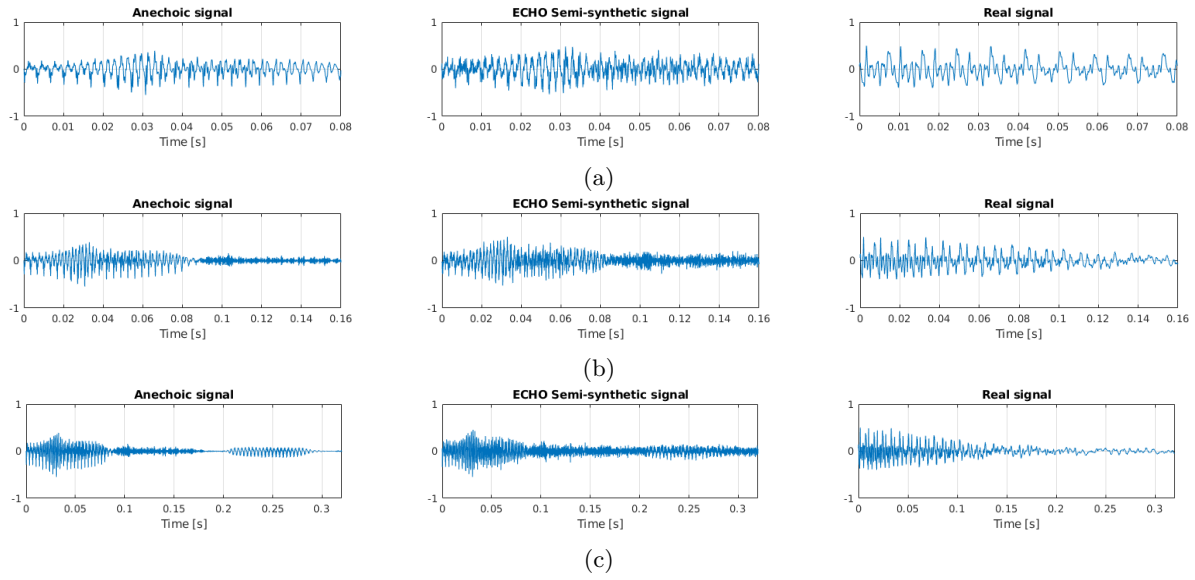

Figure 3.5: Comparison of the selected anechoic signal, the semi-synthetic generated signal and a real signal captured by a microphone for the temporal window sizes (a) 80 ms, (b) 160 ms and (c) 320 ms.

the frequency of the signal. To simulate this phenomenon, a statistical study of the *IDIAP AV16.3* database has been carried out, where the variances have been estimated for each one of the $1kHz$ frequency bins analyzed in such a way that these phase variances are applied to the anechoic signals of the *Albayzin Corpus* databaset. Physically, this process is equivalent to adding an uncertainty to the position from which the acoustic signal is virtually propagated, due to the non-linear processes that occur in this process. The proposed method is formally defined as follows

$$r_{i_{PHASE}} = \mathcal{F}^{-1}\left(R_i[n] \odot e^{-j\omega_\phi[n]}\right), \tag{3.8}$$

where R_i is the signal recorded by the microphone i in the Fourier space, $[\odot]$ is the element-wise operator and ω_ϕ is the vector with each frequency bin pulsation variance.

An example of the result of applying this method to an anechoic signal is shown in figure 3.6 as well as the shape of a signal recorded in the real room.

3.5.2.4 PHAT method

This method is based on PHAT filtering the input signals of the network. The PHAT filter is defined in [22] and involves normalization of the signal spectrum modulus. Starting from equation 3.4, this process is formally defined as

$$r_{i_{PHAT}} = \mathcal{F}^{-1}\left(\frac{R_i}{|\,R_i\,|}\right) + k_\eta\,\eta_{wgn}, \tag{3.9}$$

where R_i is the Fourier Transform of the signal recorded by th i^{th} microphone only taking into account the signlas delays due to propagation, $|\cdot|$ is the module operator, η_{wgn} is a white Gaussian noise signal and k_η is a noise scaling factor to generate signals with a similar SNR to that found in the target room recordings ($SNR \in [1, 30]\,dB$).

After applying this technique, the results obtained in figure 3.7 can be observed, as well as a comparison with a real signal. In this case, it is observed that the PHAT filter produces semi-synthetic signals that are closer to the filtered real signals.

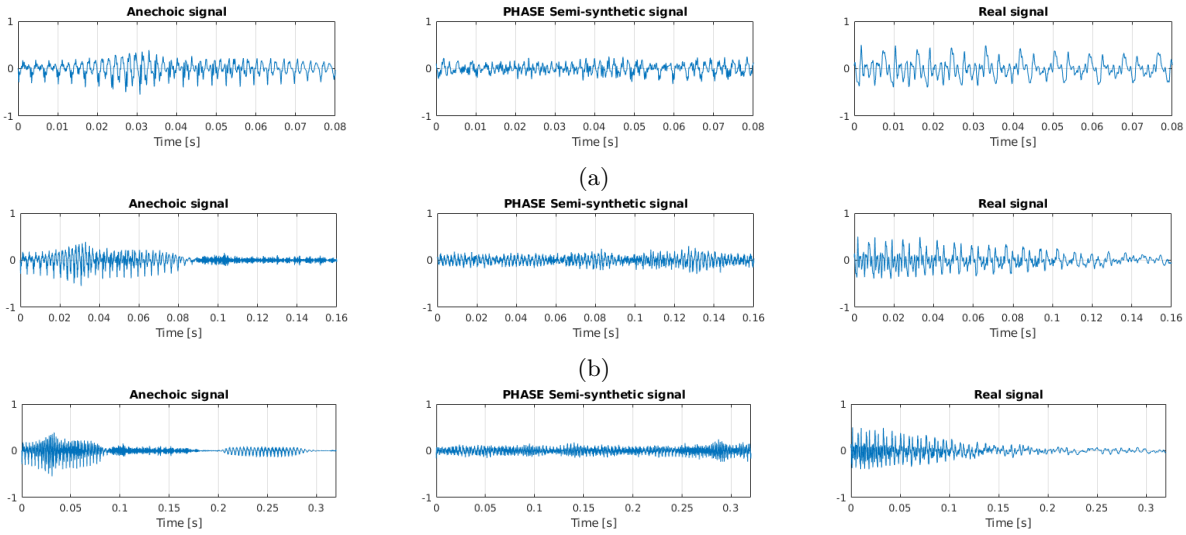

Figure 3.6: Comparison of the selected anechoic signal, the semi-synthetic generated signal and a real signal captured by a microphone for the temporal window sizes (a) 80 ms, (b) 160 ms and (c) 320 ms.

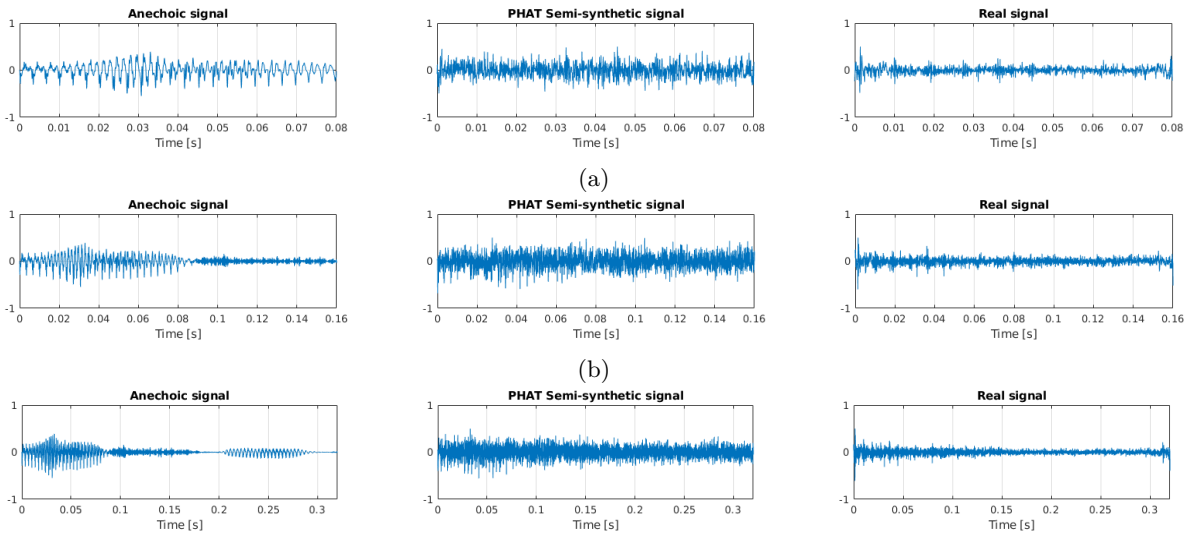

Figure 3.7: Comparison of the selected anechoic signal, the semi-synthetic generated signal and a real signal captured by a microphone for the temporal window sizes (a) 80 ms, (b) 160 ms and (c) 320 ms.

Chapter 4

Experimental work and Results

If you can dream it, you can do it.

Walt Disney

4.1 Introduction

This chapter details the conditions on which the two proposed training strategies have been tested. It also describes the general characteristics for the experimental set up as well as the error metrics used when comparing the proposal of this work with other state-of-the-art methods. Finally, the experimental results obtained will be presented.

4.2 Training and Fine Tuning Details

As described in section 3.5, two different training strategies have been defined. The first one involves a two step process where semi-synthetic data is used for training followed by fine tuning with real data. The second one involves only semi-synthetic data for training and includes some effects in the data generation process that improve generalization of the network.

Semi-synthetic data generation has been described in subsections 3.5.1.1 and 4.5.5 depending on the strategy. In both cases a set of three-dimensional positions q_i are randomly generated with uniformly distributed values in the intervals $q_{i_x} \in [0, 3.6]\, m$, $q_{i_y} \in [0, 8.2]\, m$ and $q_{i_z} \in [0.92, 1.53]\, m$ that correspond to the possible distribution of the position of a speaker's mouth in the *IDIAP* room [1].

Regarding the optimization strategy of the loss function defined in the equation 3.3, the ADAM optimizer [53] was used during 200 epochs with a batch size of 100 samples. The learning rate of the ADAM optimizer has been set to $\alpha = 10^{-3}$ while all other parameters have been set following the recommendations of [53] ($\beta_1 = 0.9$, $\beta_2 = 0.99$ and $\epsilon = 10^{-8}$). A total of 7200 different frames of input data per epoch have been generated randomly during this phase. Another 800 samples data has also been generated as validation data with which to establish a stop condition.

The experiments are carried out with three different lengths of time windows ($80ms$, $160ms$ and $320ms$) and the training phase in strategy one is run once per window length, obtaining three different network models. In each training 200 audio recordings are chosen randomly where 40 windows are extracted from each audio, also randomly. In the same way, 200 acoustic source position vectors (q_i)

randomly generated, so that each position generates 40 audio windows, all extracted from the same recording. The training phase in strategy two follows the same procedure but it is only carried out with the $320ms$ time window. For the fine-tuning phase, detailed in subsection 3.5.1.2, sequences $seq11$ and $seq15$ are mainly used, which are characterized by the appearance of a single speaker moving throughout the room while speaking. The *ADAM* optimizer is also used in this case to perform the fine-tuning, setting the learning rate to a value $\alpha = 10^{-4}$ while the rest of the parameters are configured according to the recommended values.

As will be described in section 4.5, experiments will also be carried out by performing fine-tuning with the sequences $seq01$, $seq02$ and $seq03$ in a complementary way after the fine-tuning performed with the $seq11$ and $seq15$ sequences.

4.3 Experimental Setup

In this work a simple configuration of arrays of microphones is used to evaluate our proposal in a restricted environment, as was done in [10]. We use the microphones 1, 5, 11 and 15 of the 16 available in the *IDIAP* dataset, grouped into two pairs of microphones. This configuration is the same as the one selected in [10] to provide two pairs of orthogonal microphones. This configuration is shown in the figure 3.2c in which, the microphones with the same color belong to the same pair. The result related to the length of the acoustic frame has been evaluated for $80ms$, $160ms$ and $320ms$ to accurately assess the impact of acoustic time resolution on the localization results.

We evaluate whether our *ASLNet* can be competitive compared to other traditional localization methods. In particular we use for comparison a standard SRP-PHAT method and the strategy proposed in [10], which will be referred to as GMBF. This GMBF method is based on fitting a generative model to the GCC-PHAT signals using sparse constrains, achieving a significant improvement over the *IDIAP* dataset [10,60]. The fitting procedure of the GMBF method does not require training, as opposed to the ASLNet approach.

The experiments carried out in this work are divided into 5 phases, where the first 4 correspond to the results obtained after the first strategy of training described in section 3.5.1 and the fifth phase corresponds to the second strategy described in section 4.5.5. Each of the phases is summarized below:

- **First experiment**: the ASLNet network is trained only with semi-synthetic data from scratch. The results obtained are compared with those provided by SRP-PHAT and GMBF, which are taken as baseline.
- **Second experiment**: the improvement of performance with respect to the baseline is evaluated when a single sequence is used for the fine-tuning procedure where only one speaker in movement appears throughout the room.
- **Third experiment**: the impact it has on the baseline is evaluated by using two different sequences where a single speaker appears in the fine-tuning process of the network.
- **Fourth experiment**: the performance is evaluated, with respect to the baseline, that has the fine-tuning process in the network when using sequences where a single speaker appears both in motion and in static.
- **Fifth experiment**: each of the proposed methods for generating semi-synthetic data are compared, in terms of error, against that used for the training strategy one.

Finally, we provide a comparison between our proposal and that described in [2], for which the authors kindly provided the source code [61].

4.4 Evaluation Metrics

The *ASLNet* approach obtains a set of spatial coordinates $\hat{s}_k = \left(\hat{s}_{k_x}, \hat{s}_{k_y}, \hat{s}_{k_z}\right)^\top$ that are estimates of the current position of the speaker's mouth at the time instant k. These positions are compared, in terms of Euclidean distance, with the actual position of the speaker's mouth, $s_k = \left(s_{k_x}, s_{k_y}, s_{k_z}\right)^\top$ (ground truth).

All the experiments are evaluated adopting the same metric used in [10] and developed under the CHIL project [62]. It is known as MOTP and is defined as

$$MOTP = \frac{\sum_{k=1}^{N_p} \mid s_k - \hat{s}_k \mid^2}{N_p} \, [m] \,, \tag{4.1}$$

where N_p denotes the total number of position estimations along time, $\hat{s}_k$ the estimated position vector, and s_k is the labeled ground truth position vector. Both the results obtained by the *ASLNet* approach as well as by the GMBF method will be compared with SRP-PHAT through the metric of the relative improvement of MOTP, which is defined below:

$$\Delta_r^{MOTP} = 100 \times \frac{MOTP_{SRP-PHAT} - MOTP_{proposal}}{MOTP_{SRP-PHAT}} \, [\%] \tag{4.2}$$

4.5 Results and Discussion

In this section all the results obtained for each of the four experiments proposed will be exposed according to the defined metrics. In addition, specific conclusions of them will be given.

The first experiment carried out, as mentioned above, consists in checking the performance of *ASLNet* after having been trained only with semi-synthetic data. It will be compared against the results obtained by the SRP-PHAT and GMBF algorithms. Finally, all the results obtained in this experiment will be considered as the baseline to quantify the improvement of the method proposed in the successive experiments.

4.5.1 Experiment 1: Baseline Results

The baseline results for the sequences *seq*01, *seq*02 and *seq*03 are shown in table 4.1, where the best results for each of the analyzed window sizes are highlighted in **bold font**. The table shows the results obtained by the standard algorithm of SRP-PHAT (columan SRP), the alternative proposed by [10] (column GMBF) and the proposal of this work after training only with semi-synthetic data (column *ASLNet*).

The main conclusions obtained after the baseline results are the following:

1. The MOTP values improve as the frame size increases, as expected. The larger the length of the time window, the better the correlation values will be. The best average values of MOTP for the SRP-PHAT method are around 76*cm*, and for the GMBF method around 59*cm*.

		80ms			**160**ms			**320**ms		
		SRP	GMBF	*ASLNet*	SRP	GMBF	*ASLNet*	SRP	GMBF	*ASLNet*
seq01	$MOTP(m)$	1.020	**0.795**	1.615	0.910	**0.686**	1.526	0.830	**0.588**	1.464
	Δ_r^{MOTP}		**22.1**%	−58.3%		**24.6**%	−67.7%		**29.1**%	−76.4%
seq02	$MOTP(m)$	0.960	**0.864**	2.124	0.840	**0.759**	1.508	0.770	**0.694**	1.318
	Δ_r^{MOTP}		**10.0**%	−121.3%		**9.6**%	−79.5%		**9.9**%	−71.2%
seq03	$MOTP(m)$	0.900	**0.686**	1.559	0.770	**0.563**	1.419	0.690	**0.484**	1.379
	Δ_r^{MOTP}		**23.8**%	−73.2%		**26.9**%	−84.3%		**29.9**%	−99.9%
Average	$MOTP(m)$	0.957	**0.778**	1.763	0.836	**0.666**	1.481	0.760	**0.585**	1.385
	Δ_r^{MOTP}		**18.7**%	−84.3%		**20.4**%	−77.1%		**22.9**%	−82.3%

Table 4.1: Baseline results.

2. The GMBF method achieves very relevant improvements with respect to SRP-PHAT, on the other hand, the *ASLNet* proposal (which has only been trained with semi-synthetic data so far) shows a behavior that is far from reaching the performance of SRP-PHAT or GMBF.

The results obtained reflect that there are other effects in the signal that are only present in the real data, such as reverberations, which may be affecting the network since they are not considered in the generation of training data. This can be approached by introducing realistic simulations of the propagation of the acoustic wave or by indirect techniques that induce the network to ignore or take into account these effects.

4.5.2 Experiment 2: Fine-tuning with a moving sequence

The second experiment uses additional training data from the *IDIAP* room (different from those used for testing) to fine tune the *ASLNet* weights trained with the semi-synthetic dataset. Specifically, to carry out this first experiment, sequence 15 of the *IDIAP* dataset has been used for fine-tuning. Table 4.2 shows the results obtained for the GMBF method as well as for the *ASLNet* network (column *ASLNet*-ft15).

		80ms		**160**ms		**320**ms	
		GMBF	*ASLNet*-ft15	GMBF	*ASLNet*-ft15	GMBF	*ASLNet*-ft15
seq01	$MOTP(m)$	**0.795**	0.875	**0.686**	0.833	**0.588**	0.777
	Δ_r^{MOTP}	**22.1**%	14.2%	**24.6**%	8.5%	**29.1**%	6.4%
seq02	$MOTP(m)$	0.864	**0.839**	**0.759**	0.801	**0.694**	0.731
	Δ_r^{MOTP}	10.0%	**12.6**%	**9.6**%	4.6%	**9.9**%	5.1%
seq03	$MOTP(m)$	**0.686**	0.835	**0.563**	0.806	**0.484**	0.734
	Δ_r^{MOTP}	**23.8**%	7.2%	**26.9**%	-4.7%	**29.9**%	-6.4%
Average	$MOTP(m)$	**0.778**	0.849	**0.666**	0.813	**0.585**	0.746
	Δ_r^{MOTP}	**18.7**%	11.3%	**20.4**%	2.8%	**22.9**%	1.8%

Table 4.2: Results for the fine tuned with sequence *seq15*.

These results show that the *ASLNet* network behaves similarly to the SRP-PHAT algorithm, with the network being, for the most part, slightly better. In any case, *ASLNet* is still behind the GMBF method in all but one case.

The main conclusion of this experiment is that the fine-tuning process is able to satisfactorily complement the previous training of the network with semi-synthetic data, getting the network to obtain a comparable and slightly better performance to SRP-PHAT. In particular, the following points are relevant.

1. The amount of data used for the fine-tuning process has been reduced, consisting only of a sequence of 36 seconds.

2. The speaker that appears in the scene used moves while speaking. This is contrary to the sequences used in the test, in which the speaker is static. It follows that, from this sequence, the network is able to extract enough information from the environment which favors a better estimation of the acoustic source in the test process.

3. The improvements obtained in the *ASLNet* network decrease as the size of the time window used increases. This suggests that the speed at which the speaker moves has an effect on the training, adding an uncertainty to the estimate the greater the time window used.

4. The speaker that appears in the sequence used for the fine-tuning is a male. The sequences used for the test show considerable diversity in terms of nationality and gender. Despite this, the results obtained show homogeneity, so it seems that the network does not show any dependence on these factors, unlike some other methods.

In spite of the relevant improvements with the fine-tuning approach, they are still far from making this method suitable for further competitive exploitation in the ASL scenario (provided we have the GMBF alternative), so the next experiment aimed to increase the amount of fine tuning material. Finally, as a justification of the fine-tuning process, the test has been carried out to train the network from scratch with the same sequence as that used in fine-tuning. The results obtained are shown in table 4.3 where it can be seen how the training process with semi-synthetic data and fine-tuning with a real sequence gets a better performance.

		80*ms*		**160***ms*		**320***ms*	
		ASLNet-t15	*ASLNet*-f15	*ASLNet*-t15	*ASLNet*-f15	*ASLNet*-t15	*ASLNet*-f15
seq01	$MOTP(m)$	1.009	**0.875**	0.949	**0.833**	1.0009	**0.777**
	Δ_r^{MOTP}	1.1%	**14.2**%	−4.3%	**8.5**%	−21.6%	**6.4**%
seq02	$MOTP(m)$	**0.807**	0.839	**0.767**	0.801	0.807	**0.731**
	Δ_r^{MOTP}	**15.9**%	12.6%	**8.7**%	4.6%	−4.8%	**5.1**%
seq03	$MOTP(m)$	0.935	**0.835**	0.911	**0.806**	0.936	**0.734**
	Δ_r^{MOTP}	−3.9%	**7.2**%	−18.3%	**−4.7**%	−35.7%	**−6.4**%
Average	$MOTP(m)$	0.915	**0.849**	0.875	**0.813**	0.916	**0.746**
	Δ_r^{MOTP}	4.3%	**11.3**%	−4.6%	**2.8**%	−20.6%	**1.8**%

Table 4.3: Results for the *ASLNet* proposal, either trained from scratch with sequence *seq15* (columns *ASLNet*-t15) or fine tuned with sequence *seq15* (columns *ASLNet*-f15).

4.5.3 Experiment 3: Fine-tuning with two moving sequences

The third experiment carried out only is based on increasing the amount of data used for the fine-tuning process of the *ASLNet* network. To do this, two sequences of the same nature will be used (*seq15* and *seq11*) where a single speaker will appear moving throughout the room while speaking. These sequences differ from each other in that the speakers that appear are different, the paths they follow and the speed at which they move.

Table 4.4 shows the results obtained for the GMBF method and the ASLNet network after performing the fine-tuning process.

In this case, further improvements have been achieved with respect to using only the sequence 15 for the fine-tuning process. There is only one case in which *ASLNet* does not exceed the performance of SRP-PHAT. In any case, the conclusions drawn in this experiment, in addition to the additional improvement, are similar to those in experiment two.

		80*ms*		**160***ms*		**320***ms*	
		GMBF	*ASLNet*-f15+11	GMBF	*ASLNet*-f15+11	GMBF	*ASLNet*-f15+11
seq01	$MOTP(m)$	**0.795**	0.805	**0.686**	0.750	**0.588**	0.706
	Δ_r^{MOTP}	**22.1**%	21.1%	**24.6**%	17.6%	**29.1**%	14.9%
seq02	$MOTP(m)$	0.864	**0.809**	0.759	**0.716**	**0.694**	0.712
	Δ_r^{MOTP}	10.0%	**15.7**%	9.6%	**14.8**%	**9.9**%	7.5%
seq03	$MOTP(m)$	**0.686**	0.792	**0.563**	0.732	**0.484**	0.692
	Δ_r^{MOTP}	**23.8**%	12.0%	**26.9**%	4.9%	**29.9**%	−0.3%
Average	$MOTP(m)$	**0.778**	0.802	**0.666**	0.732	**0.585**	0.703
	Δ_r^{MOTP}	**18.7**%	16.2%	**20.4**%	12.4%	**22.9**%	7.5%

Table 4.4: Relative improvements over SRP-PHAT for method GMBF and the *ASLNet* fine tuned with sequences *seq15* and *seq11*

4.5.4 Experiment 4: Fine-tuning with moving and static sequences

The last proposed experiment consists in adding to the fine-tuning process of the experiment network three sequences where a single static speaker appears. To ensure that the network uses different data at the time of training (including fine-tuning) and testing, the sequences for test and fine-tuning have been distributed as shown in table 4.5, while in table 4.8 the results obtained after the training phase according to the commented method are shown.

Test sequence	**Fine tuning sequences**
seq01	*seq15 + seq11 + seq02 + seq03*
seq02	*seq15 + seq11 + seq01 + seq03*
seq03	*seq15 + seq11 + seq01 + seq02*

Table 4.5: Fine tuning material used in the experiment corresponding to table 4.8

		80*ms*		**160***ms*		**320***ms*	
		GMBF	*ASLNet*-f15+11+st	GMBF	*ASLNet*-f15+11+st	GMBF	*ASLNet*-f15+11+st
seq01	$MOTP(m)$	0.795	**0.607**	0.686	**0.540**	0.588	**0.485**
	Δ_r^{MOTP}	22.1%	**40.5**%	24.6%	**40.7**%	29.1%	**41.6**%
seq02	$MOTP(m)$	0.864	**0.669**	0.759	**0.579**	0.694	**0.545**
	Δ_r^{MOTP}	10.0%	**30.3**%	9.6%	**31.1**%	9.9%	**29.2**%
seq03	$MOTP(m)$	**0.686**	0.707	**0.563**	0.617	**0.484**	0.501
	Δ_r^{MOTP}	**23.8**%	21.4%	**26.9**%	19.9%	**29.9**%	27.4%
Average	$MOTP(m)$	0.778	**0.664**	0.666	**0.581**	0.585	**0.511**
	Δ_r^{MOTP}	18.7%	**30.3**%	20.4%	**30.0**%	22.9%	**32.4**%

Table 4.6: Relative improvements over SRP-PHAT for the GMBF method and the *ASLNet* fine tuned with the sequences described in table 4.5

The main conclusions obtained from the results shown in table 4.8 are the following:

1. The *ASLNet* network exhibits much better behavior than the GMBF method for all window sizes tested. The average absolute improvement over SRP-PHAT for the *ASLNet* proposal is 10 points higher than GMBF, reaching 32.8% in the case of the neural network and 22.9% in the case of GMBF.

2. The effect of adding static sequences has proven to be beneficial, as expected. This improvement is mainly due to the fact that the positions where each of the sequences is generated are similar (although not identical) where the height, position and gender of the speaker are varied. This has led the network to focus on the key acoustic characteristics of these regions, thus improving the location estimate.

3. The improvement obtained is significant and is obtained through the cost of adding new real sequences to the fine-tuning process. In any case, this extra cost is still reasonable, since all the material used for the fine-tuning process has a reduced duration of 400 seconds on average.

Finally, to summarize, figure 4.1 shows a comparison of the average error of MOTP for SRP-PHAT, GMBF and the proposal of this work called *ASLNet*.

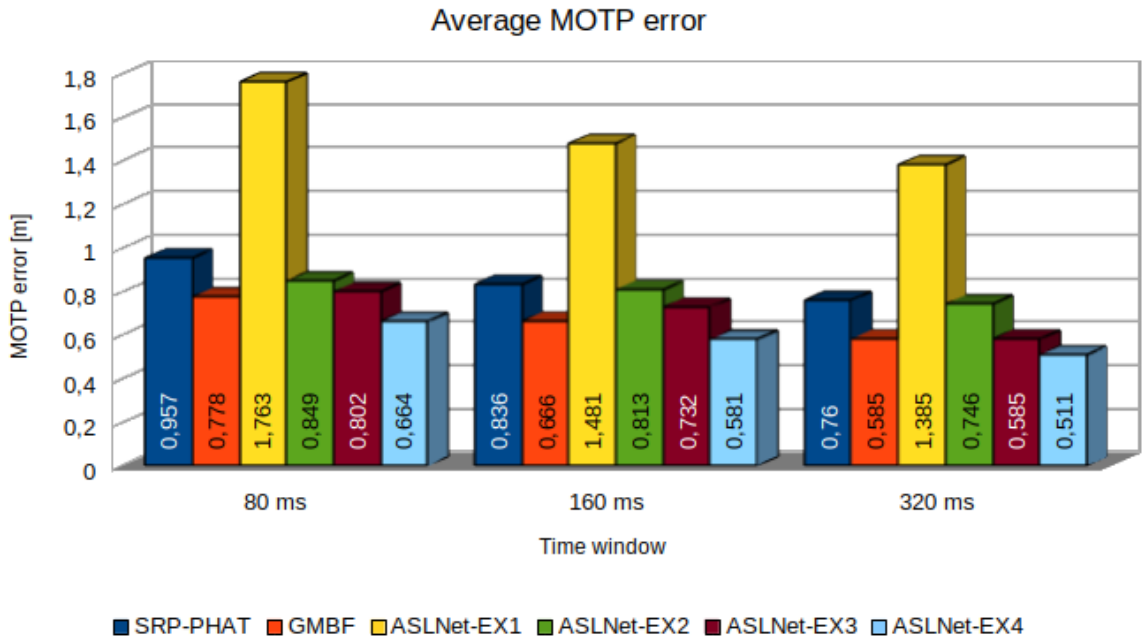

Figure 4.1: MOTP error results for SRP-PHAT, GMBF and *ASLNet* for experiments 1, 2, 3 and 4 (for all window sizes).

According to the results obtained by the proposed network, clearly the greatest contribution in terms of improvement is due to the use of fine-tuning sequences with real data (experiments 2, 3 and 4). It should be noted that these improvements are independent of the height and gender of the speaker that appears in the scene. On the other hand, it can be assumed that this proposal makes a contribution to the current state of the art of the ASL task.

4.5.5 Experiment 5: Comparison between proposed methods of semi-synthetic data generation

As it is commented in section 3.5.1.1, the dataset generator process used in the previous experiments is very simplistic. We propose in this experiment a much richer semi-synthetic data generator using all effects detailed for training strategy 2 in section . We refer to them as **NOISE**, **ECHO**, **PHASE** and **PHAT**. After performing the corresponding training for each one of the semi-synthetic data generation methods proposed with a time window of $320ms$, the results shown in table 4.7 have been obtained where the best one for each sequence is highlighted in **bold font** as well as the best average MOTP error.

It can be observed that in all the proposed methods, the performance of the *ASLNet* trained with the semi-synthetic generation method described in strategy one are significantly improved. We obtain an average an improvement of 24 cm for the worst case, corresponding to the *ECHO* method and of 46 cm for the best, corresponding to the *PHASE* method. Figure 4.2 shows a comparative graph of the error in

Test sequence	ASLNet	$\text{ASLNet}_{\text{NOISE}}$	$\text{ASLNet}_{\text{ECHO}}$	$\text{ASLNet}_{\text{PHASE}}$	$\text{ASLNet}_{\text{PHAT}}$
seq01	1.464	1.056	1.072	**0.936**	1.010
seq02	1.318	1.043	1.193	**0.938**	0.982
seq03	1.379	1.025	1.164	**0.906**	0.973
Average	1.385	1.041	1.145	**0.926**	0.987

Table 4.7: Comparison of the MOTP $[m]$ errors obtained by training only with semi-synthetic data generated with the methods proposed in 3.5 for a $320ms$ used time window.

the estimation of the location of the acoustic source for all semi-synthetic data generation methods, as well as for the SRP-PHAT algorithm and for GMBF described in [10].

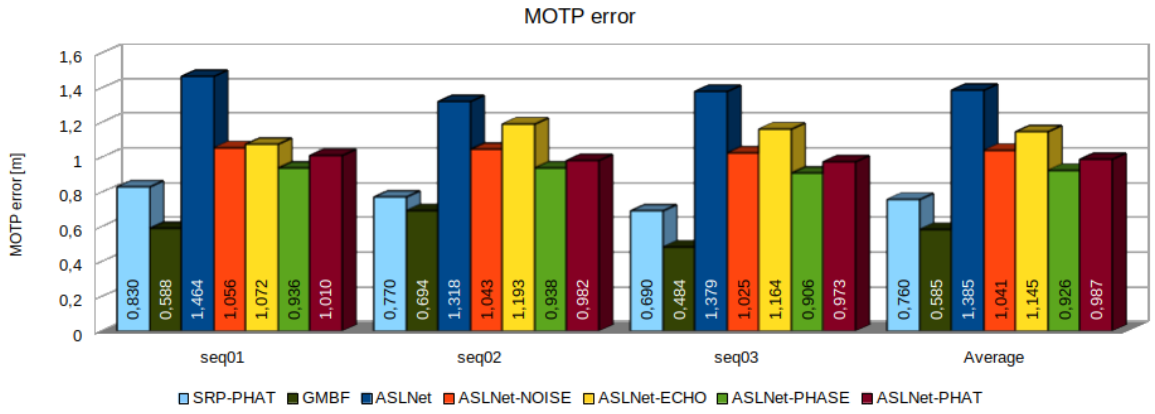

Figure 4.2: MOTP error results for SRP-PHAT, GMBF and *ASLNet* for experiment 5 for a $320ms$ time window.

In this case it can be observed how none of the proposed methods improve neither the SRP-PHAT algorithm nor the GMBF. However, an improvement is observed when using more complex data generation methods, especially in those that act on the phase of the signals generated, instituting a possible future line focused on this particular aspect.

4.5.6 Comparison with other deep learning methods

In this section, we also provide a comparison between our proposal and a recent deep learning ASL method known as *SELDnet* [2], for which the source code is available in [61]. *SELDnet* is a Convolutional Recurrent Neural Networks (CRNN) architecture that uses the signal spectrum of the audio signals as inputs (phase and magnitude components of the spectrogram calculated on each audio channel) and is able to deal with multiple overlapping sound events.

SELDNet generates two different outputs:

- **Classification output**: The first output of the *SELDnet* is able to classify the sound events among a list of classes for each consecutive frame in the input audio signals.

- **Regression output**: The second output estimates the DOA vector detected on each of the consecutive frames in the audio input. This vector is parametrized as the x, y, and z axis coordinates of the DOA on a unit sphere around the microphone, which is claimed to lead to a network that learns better than one that uses a parametrization based on angles.

As suggested by the authors, we used the default values of the *SELDnet* design parameters regarding the feature extraction, network model, and training process, and in order to carry out the comparison with our method, the following issues were taken into account:

- *SELDnet* uses an audio window for each microphone and extracts consecutive overlapped frames to compute the spectral components that are used as inputs. To compare this with our network, we performed experiments with different windows sizes: $80ms$, $160ms$ and $320ms$.
- Due to the fact that we used sequences of audio where only a single speaker appeared simultaneously, we assigned the same label ("*speech*") to all the audio windows used for training.
- We needed *SELDnet* to infer the x, y, z coordinates of the target source, instead of the DOA vector. This only required us to change the target output during training, as the network model does not change it at all. Our spatial coordinates were also normalized in the interval $[?1, 1]$ which is compatible with the regression output of the *SELDnet*. The final output coordinates were again denormalized back to metric coordinates to proceed with the MOTP calculations.
- We followed the same experimental procedure as in our proposal (initial semi-synthetic training followed by fine tuning) in a resource-restricted scenario using only two microphone pairs. The experimental conditions were those for which we got the best performance (included in Table 4.8), that is, using the testing and fine tuning sequences described in Table 4.5.

Table 9 shows the relative improvements of the proposal in [2] (column *SELDnet*) and our *ASLNet* approach (column *ASLNet*-f15+11+st) over SRP-PHAT.

		80ms		**160**ms		**320**ms	
		SELDNet	*ASLNet*-f15+11+st	*SELDNet*	*ASLNet*-f15+11+st	*SELDNet*	*ASLNet*-f15+11+st
seq01	$MOTP(m)$	1.037	**0.607**	1.039	**0.540**	1.076	**0.485**
	Δ_r^{MOTP}	−1.7%	**40.5**%	−14.2%	**40.7**%	−29.6%	**41.6**%
seq02	$MOTP(m)$	1.035	**0.669**	1.003	**0.579**	0.981	**0.545**
	Δ_r^{MOTP}	−7.8%	**30.3**%	−19.4%	**31.1**%	−27.4%	**29.2**%
seq03	$MOTP(m)$	1.017	**0.707**	0.991	**0.617**	1.020	**0.501**
	Δ_r^{MOTP}	−13.0%	**21.4**%	−28.7%	**19.9**%	−47.8%	**27.4**%
Average	$MOTP(m)$	1.029	**0.664**	1.010	**0.581**	1.024	**0.511**
	Δ_r^{MOTP}	−7.6%	**30.3**%	−20.7%	**30.0**%	−34.9%	**32.4**%

Table 4.8: Relative improvements over SRP-PHAT for the strategy in [2] (column *SELDNet* and the *ASLNet* fine tuned with the sequences described in table 4.5

Chapter 5

Conclusions and future work

In this work we have presented the *ASLNet*, a CNN model for solving the ASL problem from the signals captured in a set of microphones. It has been shown that this method is quite promising. Regarding the training strategy, we have proposed a two step approach. We first train the network using semi-synthetic data and then we follow by fine tuning the network using real data. The results show that our method greatly improves the state-of-the-art performance, specifically in comparison with SRP-PHAT [9] and GMBF [10] algorithms. We have shown that this is only possible if enough and rich training data is available. Due to the fact that labelling localization data is a complex and costly process, we have also explored the possibility of only using semi-synthetic data for training. We propose strategies to improve the way we generate the data, including reverberation effects, distortion and realistic noise. Although we have not improved with this strategy the state-of-the-art [9, 10] methods, we have obtained promising results.

The following future lines derived from this work are proposed:

- Implementation of more complex networks that are capable of exploiting information more efficiently than our *ASLNet* network. This network consists of only five convolutional layers and a fully-connected block. It is expected that a network including more sophisticated layers, such as residual or recurrent networks, could obtain better performance.
- Discretization the search space in which the network is applied by returning an occupation map where it is assigned greater likelihood to bounded regions close to the acoustic source. This is similar to how the SRP algorithm operates.
- Generation of our own database, which would be adapted to the specifications required for the ASL task.
- Development of a model that is capable of performing the task of ASL in rooms with different geometry and acoustic properties. This task entails some difficulties, since rooms with different geometry imply different reverberation effects.
- Development of more realistic and complex semi-synthetic data generation methods. At this point there is a compromise with the previous point because the simulated signal will be more realistic the more specific knowledge we have from the target room.
- Implementation of a method that is capable of performing the task of localization in scenarios with multiple speakers. This topic is very popular today and is very challenging.

Bibliography

[1] G. Lathoud, J.-M. Odobez, and D. Gatica-Perez, "Av16.3: An audio-visual corpus for speaker localization and tracking," in *Proceedings of the MLMI*, ser. Lecture Notes in Computer Science, S. Bengio and H. Bourlard, Eds., vol. 3361. Springer-Verlag, 2004, pp. 182–195.

[2] S. Adavanne, A. Politis, J. Nikunen, and T. Virtanen, "Sound event localization and detection of overlapping sources using convolutional recurrent neural networks," *CoRR*, vol. abs/1807.00129, 2018. [Online]. Available: http://arxiv.org/abs/1807.00129

[3] J. Torres-Solis, T. H. Falk, and T. Chau, "A review of indoor localization technologies: towards navigational assistance for topographical disorientation," in *Ambient Intelligence*, F. J. V. Molina, Ed. Rijeka: IntechOpen, 2010, ch. 3. [Online]. Available: https://doi.org/10.5772/8678

[4] T. Ruiz-López, J. L. Garrido, K. Benghazi, and L. Chung, "A survey on indoor positioning systems: Foreseeing a quality design," in *Distributed Computing and Artificial Intelligence*. Springer Berlin Heidelberg, 2010, pp. 373–380.

[5] L. Mainetti, L. Patrono, and I. Sergi, "A survey on indoor positioning systems," in *2014 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE, 2014, pp. 111–120.

[6] A. Moreno, D. Poch, A. Bonafonte, E. Lleida, J. Llisterri, J. B. Mariño, and C. Nadeu, "Albayzin speech database: design of the phonetic corpus." in *EUROSPEECH*. ISCA, 1993. [Online]. Available: http://dblp.uni-trier.de/db/conf/interspeech/eurospeech1993.html#MorenoPBLLMN93

[7] H. D. III and D. Marcu, "Domain adaptation for statistical classifiers," *CoRR*, vol. abs/1109.6341, 2011. [Online]. Available: http://arxiv.org/abs/1109.6341

[8] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3320–3328. [Online]. Available: http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf

[9] J. DiBiase, "A high-accuracy, low-latency technique for talker localization in reverberant environments using microphone arrays," Ph.D. dissertation, Brown University, 2000.

[10] J. Velasco, D. Pizarro, and J. Macias-Guarasa, "Source localization with acoustic sensor arrays using generative model based fitting with sparse constraints," *Sensors*, vol. 12, pp. 13 781–13 812, 10/2012 2012.

[11] M. S. Brandstein and H. F. Silverman, "A practical methodology for speech source localization with microphone arrays," *Computer Speech & Language*, vol. 11, no. 2, pp. 91–126, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0885230896900248

[12] M. Gillette and H. Silverman, "A linear closed-form algorithm for source localization from time-differences of arrival," *Signal Processing Letters, IEEE*, vol. 15, pp. 1 –4, 2008.

[13] A. Marti, M. Cobos, J. J. Lopez, and J. Escolano, "A steered response power iterative method for high-accuracy acoustic source localization," *The Journal of the Acoustical Society of America*, vol. 134, no. 4, pp. 2627–2630, 2013. [Online]. Available: https://doi.org/10.1121/1.4820885

[14] M. Compagnoni, A. Pini, A. Canclini, P. Bestagini, F. Antonacci, S. Tubaro, and A. Sarti, "A geometrical-statistical approach to outlier removal for tdoa measurements," *IEEE Transactions on Signal Processing*, vol. 65, no. 15, pp. 3960–3975, Aug 2017.

[15] J. Velasco, D. Pizarro, J. Macias-Guarasa, and A. Asaei, "TDOA matrices: Algebraic properties and their application to robust denoising with missing data," *IEEE Transactions on Signal Processing*, vol. 64, no. 20, pp. 5242–5254, Oct 2016.

[16] J. P. Dmochowski and J. Benesty, "Steered beamforming approaches for acoustic source localization," in *Speech Processing in Modern Communication*, ser. Springer Topics in Signal Processing, I. Cohen, J. Benesty, and S. Gannot, Eds. Springer Berlin Heidelberg, 2010, vol. 3, pp. 307–337, 10.1007/978-3-642-11130-3_12. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-11130-3_12

[17] J. DiBiase, H. Silverman, and M. Brandstein, "Robust localization in reverberant rooms," *Microphone Arrays*, pp. 157–180, 2001.

[18] J. Velasco, C. J. Martin-Arguedas, J. Macias-Guarasa, D. Pizarro, and M. Mazo, "Proposal and validation of an analytical generative model of SRP-PHAT power maps in reverberant scenarios," *Signal Processing*, vol. 119, pp. 209 – 228, 2016.

[19] A. Deleforge, "Acoustic Space Mapping: A Machine Learning Approach to Sound Source Separation and Localization," Theses, Université de Grenoble, November 2013. [Online]. Available: https://tel.archives-ouvertes.fr/tel-00913965

[20] D. Salvati, C. Drioli, and G. L. Foresti, "On the use of machine learning in microphone array beamforming for far-field sound source localization," in *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, Sept 2016, pp. 1–6.

[21] J. M. Vera-Diaz, D. Pizarro, and J. Macias-Guarasa, "Towards end-to-end acoustic localization using deep learning: From audio signals to source position coordinates," *Sensors*, vol. 18, no. 10, 2018. [Online]. Available: http://www.mdpi.com/1424-8220/18/10/3418

[22] C. Knapp and G. Carter, "The generalized correlation method for estimation of time delay," *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 24, no. 4, pp. 320 – 327, aug 1976.

[23] M. T. H. F. Dan Foresee, "Gauss-newton approximation to bayesian learning," in *Proceedings of International Conference on Neural Networks (ICNN'97)*, vol. 3, June 1997, pp. 1930–1935 vol.3.

[24] J. J. Moré, "The levenberg-marquardt algorithm: Implementation and theory," in *Numerical Analysis*, G. A. Watson, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1978, pp. 105–116.

[25] H. Do, H. F. Silverman, and Y. Yu, "A real-time srp-phat source location implementation using stochastic region contraction(src) on a large-aperture microphone array," *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, vol. 1, pp. I–121–I–124, 2007.

[26] M. Cobos, M. G. Pineda, and M. Arevalillo-Herráez, "Steered response power localization of acoustic passband signals," *IEEE Signal Process. Lett.*, vol. 24, no. 5, pp. 717–721, 2017. [Online]. Available: https://doi.org/10.1109/LSP.2017.2690306

[27] M. Omologo and P. Svaizer, "Use of the crosspower-spectrum phase in acoustic event location," *IEEE Transactions on Speech and Audio Processing*, vol. 5, no. 3, pp. 288–292, May 1997.

[28] M. Cobos, A. Marti, and J. J. Lopez, "A modified srp-phat functional for robust real-time sound source localization with scalable spatial sampling," *IEEE Signal Processing Letters*, vol. 18, no. 1, pp. 71–74, Jan 2011.

[29] P. Pertilä and E. Cakir, "Robust direction estimation with convolutional neural networks based steered response power," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 6125–6129.

[30] M. Cobos, M. García-Pineda, and M. Arevalillo-Herráez, "Steered response power localization of acoustic passband signals," *IEEE Signal Processing Letters*, vol. 24, no. 5, pp. 717–721, May 2017.

[31] D. Salvati, C. Drioli, and G. L. Foresti, "A weighted mvdr beamformer based on svm learning for sound source localization," *Pattern Recognition Letters*, vol. 84, pp. 15 – 21, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865516301659

[32] E. A. P. Habets, J. Benesty, S. Gannot, and I. Cohen, *The MVDR Beamformer for Speech Enhancement*, ser. Springer Topics in Signal Processing. Springer Berlin Heidelberg, 2010, vol. 3, pp. 225–254. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-11130-3_9

[33] R. Schmidt, "Multiple emitter location and signal parameter estimation," *Antennas and Propagation, IEEE Transactions on*, vol. 34, no. 3, pp. 276 – 280, mar 1986.

[34] A. Wang, K. Yao, R. E. Hudson, D. Korompis, F. Lorenzelli, S. F. Soli, and S. Gao, "A high performance microphone array system for hearing aid applications," in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 6, May 1996, pp. 3197–3200 vol. 6.

[35] H. Wang and M. Kaveh, "Coherent signal-subspace processing for the detection and estimation of angles of arrival of multiple wide-band sources," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 33, pp. 823 – 831, 09 1985.

[36] T. L. Tung, K. Yao, D. Chen, R. E. Hudson, and C. W. Reed, "Source localization and spatial filtering using wideband music and maximum power beamforming for multimedia applications," in *1999 IEEE Workshop on Signal Processing Systems. SiPS 99. Design and Implementation (Cat. No.99TH8461)*, Oct 1999, pp. 625–634.

[37] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, http://www.deeplearningbook.org.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. USA: Curran Associates Inc., 2012, pp. 1097–1105. [Online]. Available: http://dl.acm.org/citation.cfm?id=2999134.2999257

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[40] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, Nov 2012.

[41] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 32, no. 2. Bejing, China: PMLR, 22–24 Jun 2014, pp. 1764–1772, http://proceedings.mlr.press/v32/graves14.html.

[42] L. Deng and J. C. Platt, "Ensemble deep learning for speech recognition," in *INTERSPEECH*, 2014.

[43] J. C. Murray, H. R. Erwin, and S. Wermter, "Robotic sound-source localisation architecture using cross-correlation and recurrent neural networks," *Neural Networks*, vol. 22, no. 2, pp. 173 – 189, 2009, what it Means to Communicate. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608009000136

[44] B. Z. Steinberg, M. J. Beran, S. H. Chin, and J. H. Howard, "A neural network approach to source localization," *The Journal of the Acoustical Society of America*, vol. 90, no. 4, pp. 2081–2090, 1991. [Online]. Available: https://doi.org/10.1121/1.401635

[45] M. S. Datum, F. Palmieri, and A. Moiseff, "An artificial neural network for sound localization using binaural cues," *The Journal of the Acoustical Society of America*, vol. 100, no. 1, pp. 372–383, 1996. [Online]. Available: https://doi.org/10.1121/1.415854

[46] X. Xiao, S. Zhao, X. Zhong, D. L. Jones, E. S. Chng, and H. Li, "A learning-based approach to direction of arrival estimation in noisy and reverberant environments," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 2814–2818.

[47] R. Takeda and K. Komatani, "Discriminative multiple sound source localization based on deep neural networks using independent location model," in *2016 IEEE Spoken Language Technology Workshop (SLT)*, Dec 2016, pp. 603–609.

[48] ——, "Sound source localization based on deep neural networks with directional activate function exploiting phase information," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 405–409.

[49] Y. Sun, J. Chen, C. Yuen, and S. Rahardja, "Indoor sound source localization with probabilistic neural network," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 8, pp. 6403–6413, Aug 2018.

[50] S. Chakrabarty and E. A. P. Habets, "Multi-speaker localization using convolutional neural network trained with noise," *CoRR*, vol. abs/1712.04276, 2017. [Online]. Available: http://arxiv.org/abs/1712.04276

[51] D. Salvati, C. Drioli, and G. L. Foresti, "Exploiting cnns for improving acoustic source localization in noisy and reverberant conditions," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 103–116, April 2018.

[52] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, Y. Lechevallier and G. Saporta, Eds. Heidelberg: Physica-Verlag HD, 2010, pp. 177–186.

[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[54] E. E. L. R. Association, "Albayzin corpus," http://catalogue.elra.info/en-us/repository/browse/albayzin-corpus/b50c9628a9dd11e7a093ac9e1701ca0253c876277d534e7ca4aca155a5611535/. [Online]. Available: http://catalogue.elra.info/en-us/repository/browse/albayzin-corpus/b50c9628a9dd11e7a093ac9e1701ca0253c876277d534e7ca4aca155a5611535/

[55] D. C. Moore, "The idiap smart meeting room," IDIAP Research Institute, Switzerland, Tech. Rep., November 2004. [Online]. Available: http://publications.idiap.ch/downloads/reports/2002/com02-07.pdf

[56] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ser. ICML'11. USA: Omnipress, 2011, pp. 265–272. [Online]. Available: http://dl.acm.org/citation.cfm?id=3104482.3104516

[57] J. B. Allen and D. A. Berkley, "dd," *The Journal of the Acoustical Society of America*, vol. 65, no. 4, pp. 943–950, 1979. [Online]. Available: https://doi.org/10.1121/1.382599

[58] M. Vorlander, "Simulation of the transient and steady-state sound propagation in rooms using a new combined ray-tracing/image-source algorithm," *The Journal of the Acoustical Society of America*, vol. 86, no. 1, pp. 172–178, 1989. [Online]. Available: https://doi.org/10.1121/1.398336

[59] E. A. Lehmann and A. M. Johansson, "Diffuse reverberation model for efficient image-source simulation of room impulse responses," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1429–1439, Aug 2010.

[60] J. F. Velasco-Cerpa, "Mathematical modelling and optimization strategies for acoustic source localization in reverberant environments," Ph.D. dissertation, Escuela Politécnica Superior. University of Alcalá (Spain), March 2017.

[61] S. Adavanne, A. Politis, J. Nikunen, and T. Virtanen, "Sound event localization and detection of overlapping sources using convolutional recurrent neural network (seldnet)," https://github.com/sharathadavanne/seld-net, august 2018.

[62] D. Mostefa, M. Garcia, K. Bernardin, R. Stiefelhagen, J. McDonough, M. Voit, M. Omologo, F. Marques, H. Ekenel, and A. Pnevmatikakis, "Clear evaluation plan, document chil-clear-v1.1 2006-02-21," http://www.clear-evaluation.org/clear06/downloads/chil-clear-v1.1-2006-02-21.pdf (accessed on 11 october 2012), 2006. [Online]. Available: http://www.clear-evaluation.org/clear06/downloads/chil-clear-v1.1-2006-02-21.pdf

Appendix A

Tools and Resources

The necessary tools and resources for the successful development of this work have been:

- Computer compatible with at least the following features:
  - Intel Core i7 CPU
  - 4 GB RAM
  - 50 GB free on the hard drive
  - NVIDIA GeForce GTX 980 GPU
- Operating system based on GNU/Linux
- PyCharm development environment
- Python 2.7.12 version with at least the following packages:
  - Tensorflow-gpu 1.4.1
  - Keras 2.1.5
  - Numpy 1.14.2
  - SoundFile 0.10.2
  - matplotlib 2.2.2
  - scipy 1.0.1
- LaTeXdocument preparation system
- GIT version control

Appendix B

Budget

Once the tools and resources necessary for the development of this work have been defined, the budget necessary to carry out the proposed project can be deducted. In this budget the costs due to hardware, software and labor will be taken into account.

- **Hardware budget**

Concept	Quantity	Unit cost	Subtotal
Compatible PC	1	1000 €	1000 €
NVIDIA GeForce 980 GTX GPU	1	437 €	437 €
SubTotal			1437 €

- **Software budget**

Concept	Quantity	Unit cost	Subtotal
Ubuntu 16.04 LTS OS	1	0 €	0 €
PyCharm IDE	1	0 €	0 €
Python libraries	some	0 €	0 €
LaTeXIDE	1	0 €	0 €
Git version control	1	0 €	0 €
SubTotal			0 €

- **Labor budget**

Concept	Hours	€/ hour	Subtotal
Software development	640	50 €	32000 €
Document writing	200	10 €	2000 €
SubTotal			34000 €

- **Total budget**

Concept	Subtotal
Hardware budget	1437 €
Software budget	0 €
Labor budget	34000 €
Total budget	35437 €