Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Speech Enhancement using Deep Learning

Degree's Thesis
Audiovisual Systems Engineering

Author: Dan Mihai Badescu
Advisors: Antonio Bonafonte Cávez

Universitat Politècnica de Catalunya (UPC)
2016 - 2017

Abstract

This thesis explores the possibility to achieve enhancement on noisy speech signals using Deep Neural Networks. Signal enhancement is a classic problem in speech processing. In the last years, researches using deep learning has been used in many speech processing tasks since they have provided very satisfactory results.

As a first step, a *Signal Analysis Module* has been implemented in order to calculate the magnitude and phase of each audio file in the database. The signal is represented into its magnitude and its phase, where the magnitude is modified by the neural network, and then it is reconstructed with the original phase.

The implementation of the Neural Networks is divided into two stages.The first stage was the implementation of a *Speech Activity Detection Deep Neural Network* (SAD-DNN). The magnitude previously calculated, applied to the noisy data, will train the SAD-DNN in order to classify each frame in speech or non-speech. This classification is useful for the network that does the final cleaning. The *Speech Activity Detection Deep Neural Network* is followed by a *Denoising Auto-Encoder* (DAE). The magnitude and the label speech or non-speech will be the input of this second Deep Neural Network in charge of denoising the speech signal. The first stage is also optimized to be adequate for the final task in this second stage.

In order to do the training, Neural Networks require datasets. In this project the Timit corpus [9] has been used as dataset for the clean voice (target) and the QUT-NOISE TIMIT corpus[4] as noisy dataset (source).

Finally, *Signal Synthesis Module* reconstructs the clean speech signal from the enhanced magnitudes and the phase.

In the end, the results provided by the system have been analysed using both objective and subjective measures.

Resum

Aquesta tesis explora la possibilitat d'aconseguir millorar senyals de veu amb soroll, utilitzant Xarxes Neuronals Profundes. La millora de senyals és un problema clàssic del processatde senyal, però recentment s'està investigant amb deep learning, ja que són tècniques que han donat resultats molt satisfactoris en moltes tasques de processament de veu.

Com a primer pas, s'ha implementat un *Mòdul d'Anàlisi de Senyal* amb l'objectiu d'extreure el mòdul i la fase de cada arxiu d'àudio de la base de dades. El senyal es representa en mòdul i fase, on el mòdul es modifica amb la xarxa neuronal i posteriorment es reconstrueix amb la fase original.

La implementació de les Xarxes Neuronals consta de dues etapes. En la primera etapa es va implementar una Xarxa Neuronal de *Detecció d'Activitat de Veu*. El mòdul prèviament calculat, aplicat a les dades amb soroll, s'utilitza com entrada per entrenar aquesta xarxa, de manera que s'aconsegueix classificar cada trama en veu o no veu. Aquesta classificació és útil per la xarxa que fa la neteja final. A continuació de la Xarxa Neuronal de *Detecció d'Activitat de Veu* s'implementa una altra amb l'objectiu d'eliminar el soroll. El mòdul, juntament amb la etiqueta obtinguda en la xarxa anterior, seran l'entrada d'aquesta nova xarxa. En aquesta segona etapa també s'optimitza la primera per adaptar-se a la tasca final.

Les Xarxes Neuronals requereixen bases de dades per fer l'entrenament. En aquest projecte s'ha utilitzat el Timit corpus [9] com a base de dades de veu neta (objectiu) i el QUT-NOISE TIMIT[4] com a base de dades amb soroll (font).

A continuació, el *Mòdul de Síntesi de Senyal* reconstrueix el senyal de veu net a partir del mòdul netejat i la fase original.

Finalment, els resultats obtinguts del sistema van ser analitzats utilitzant mesures objectives i subjectives.

Resumen

Esta tesis explora la posibilidad de conseguir mejorar señales de voz con ruido utilizando Redes Neuronales Profundas. La mejora de señales es un problema clásico del procesado de señal, pero recientemente se esta investigando con deep learning, ya que son técnicas que han dado resultados muy satisfactorios en muchas tareas del procesado de señal.

Como primer paso, se ha implementado un *Módulo de Análisis de Señal* con el objetivo de extraer el módulo y fase de cada archivo de voz de la base de datos. La señal se representa en módulo y fase, donde el módulo se modifica con la red neuronal y posteriormente se reconstruye con la fase original.

La implementación de la Red Neuronal consta de dos etapas. En la primera etapa se implementó una Red Neuronal de *Detección de Actividad de Voz*. El módulo previamente calculado, aplicado a los datos con ruido, se utiliza como entrada para entrenar esta red, de manera que se consigue clasificar cada trama en voz o no voz. Esta clasificación es útil para la red que se encarga de hacer la limpieza. A continuación de la Red Neuronal de *Detección de Actividad de Voz* se implementa otra, con el objetivo de eliminar el ruido. El módulo junto con la etiqueta obtenida en la red anterior serán la entrada de esta nueva red. En esta segunda etapa también se optimiza la primera para adaptarse a la tarea final.

Las Redes Neuronales requieren bases de datos para el entrenamiento. En este proyecto se ha utilizado el Timit corpus [9] como base de datos de voz limpia (objetivo) y el QUT-NOISE TIMIT [4] como base de datos con ruido (fuente).

A continuación, el *Módulo de Síntesis de Señal* reconstruye la señal de voz limpia a partir del módulo sin ruido y la fase original.

Finalmente, los resultados obtenidos por el sistema fueron analizados utilizando medidas objetivas y subjetivas.

Acknowledgements

First of all, I want to thank my project supervisor, Antonio Bonafonte Cávez, for offering me the chance to collaborate in this project. Thank you for your dedication and your patience when guiding me and teaching me.

I would also like to thank Santiago Pascual for all the help he provided me on topics like Deep Learning. A special thank to my colleague Albert Aparicio for helping me in all my doubts.

To my family. Nu s-au inventat încа cuvinte care să descrie recunoștința pe care v-o port. Vă mulțumesc pentru educația și pentru această viață minunată pe care mi-ați oferit-o . Sunt conștient că nu a fost ușor și știu nenumăratele sacrificii pe care le-ați făcut pentru a-mi oferi șansa de a mă afla astăzi aici. Mama, Tata, Ioana, Tata Nelu, Mimi și Camelia vă iubesc!

To Graciela. Siempre estuviste a mi lado, apoyandome en lo bueno, pero sobre todo en los momentos complicados. Sé que estos últimos años los sufriste codo con codo conmigo, o incluso más. Eres el faro que me ilumina el camino día a día. Por todo ésto y más, gracias. Ahora se abrirá una nueva etápa en nuestras vidas y juntos la disfrutaremos.

To Adrià. Ay amigo! Al final el túnel se acabó. Parecía eterno, no? Te hablaría de todos los días juntos pero me extendería más que en la tesis. El túnel ha acabado pero el camino sigue y juntos lo seguieremos recorriendo. Para acabar, gracias por conocerte en ese Laboratorio de Física.

Revision history and approval record

Revision	Date	Purpose
0	05/12/2016	Document creation
1	27/12/2016	Document 1st revision
2	03/01/2017	Document revision
3	13/01/2017	Document approbation

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Dan Mihai Badescu	dan.mihai.badescu@alu-etsetb.upc.edu
Antonio Bonafonte Cávez	antonio.bonafonte@upc.edu

Written by:		Reviewed and approved by:	
Date	10/01/2017	**Date**	13/01/2017
Name	Dan Mihai Badescu	**Name**	Antonio Bonafonte Cávez
Position	Project Author	**Position**	Project Supervisor

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, firstly, we are going to present the statement of purpose of this thesis. Then, the requirements and specifications will be explained, along with the methods, procedures and work plan necessary to finish the project. Finally, the incidents and modifications during the development of the project will be mentioned.

1.1 Statement of purpose

Since the beginning of communication, the interfering noise signal has been one of the major problems that Engineering had to face with. In particular, in speech processing, the background noise can produce degradation for real-world applications, including speech communication, hearing aids or speech recognition [2, 13, 17].

Because an important number of speech processing task, such as speaker identification, speech recognition or voice activity detection systems require a good quality input signals, Speech Enhancement systems play a very important role in order to reduce the noise. Therefore, they are often employed as a pre-processing step [2, 13, 17].

Until very recently the main algorithms used to develop Speech Enhancement systems were broadly classified in four categories:

- Spectral subtractive algorithms.
- Statistical-model-based algorithms.
- Subspace algorithms.
- Binary Masking algorithms.

However, due to the new more powerful hardware which allows more power of calculation, the Neural Networks have become the new state of art in many fields, including speech processing. Their ability to model complex non-linear mapping functions makes them suitable for denoising tasks. Neural Network requires a big amount of data in order to learn. In particularly, in speech enhancement it is less difficult because it is possible to generate noise.

The goal of this thesis is to propose a solution that enhance noisy speech signals, taking advantage of state of the art deep learning techniques.Therefore, the project explores new techniques in order to develop a competitive framework to classify and then localize and eliminate noise signals, in order to achieve clean speech from noisy speech signals. The dataset used to fulfil this task is the QUT-NOISE-TIMIT corpus [5]. This database will be described in Section 3.1

In particular, the main contributions of this project are:

- Develop a module which prepares the data for training sessions.
- Develop and train a new Neural Network architecture model, in order to clean noisy speech signal.

This project has been developed as a contribution to to the research of the investigation group "VEU", which is part of the *Departament de Teoria del Senyal i Comunicacions* (TSC) at the *Universitat Politècnica de Catalunya* (UPC), during the Fall 2016 semester.

1.2 Requirements and specifications

This project has been developed with the goal to be a tool for other students and researchers to continue investigating in the area of Speech Processing.

The requirements of this project are the following:

- Develop a module to extract the Analysis and Synthesis spectrogram. The Analysis spectrogram will be processed by the Neural Network.
- Design and train a Neural Network to achieve Speech Enhancement.

Because the project was developed from scratch, the specifications of the project were decided considering the needs of the project and the available resources.

1.3 Methods and procedures

This project tries to provide a solution to background noise on speech signals. The proposed solution consists of three steps that can be observed in Figure1.1. The first stage consist of a module which extract the magnitude and phase from noisy wav files. Then a Neural Network will be implemented in order to map the noisy samples extracted from the magnitude previously calculated to their clean version. Next step consist in recreate the wav file from the clean magnitude and the phase.

All the tests were performed using the QUT-NOISE-TIMIT [5], which consists of 600 hours, with five different noise scenarios (CAFE, HOME, STREET, CAR and REVERB) recorded in ten unique locations.

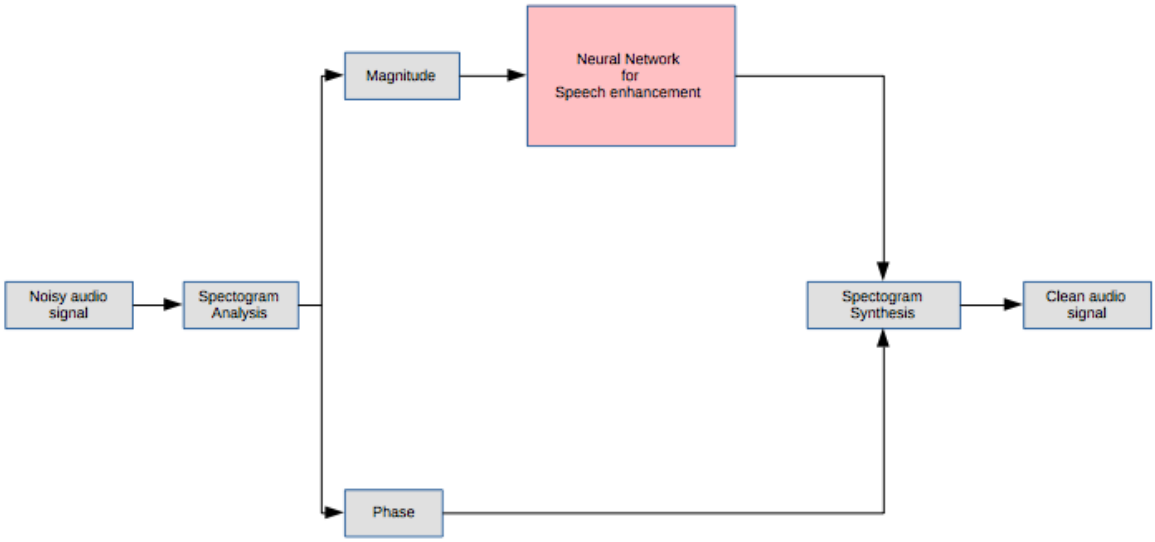

Figure 1.1: Global architecture of the solution

The development was basically done in *Python* with the *Numpy* library and using *Keras* as a framework in order to design and train our models. This Deep Learning wrapper can be used with two frameworks as a back-end: *Theano* and *TensoFlow*. Both frameworks support complex and high demanding computations over the CPU and GPU. For this project we decided to use *Theano* as a back-end because it is faster and more memory efficient. For the database creation, *Matlab* was used.

Additionally, specific hardware was required. In order to manage the high demanding computational resources needed to prepare the data and to train the models subsequently, a GPUs were required. They were provided by the "VEU" group at UPC.

1.4 Work Plan

The project planning was made in order to follow the packages detailed in this section, with the exceptions of several modifications explained in the Section 1.5.

1.4.1 Work Packages

- WP 1: Study of background knowledge
- WP 2: Development of the Signal Analysis and Signal Synthesis modules
- WP 3: Development of Neural Networks architectures
- WP 4: Evaluation
- WP 5: Documentation

The work packages will be detailed in the Gantt diagram, in Figure 1.2. There, it will be mentioned every task of each work package and the time necessary to fulfil it.

1.4.2 Gantt Diagram

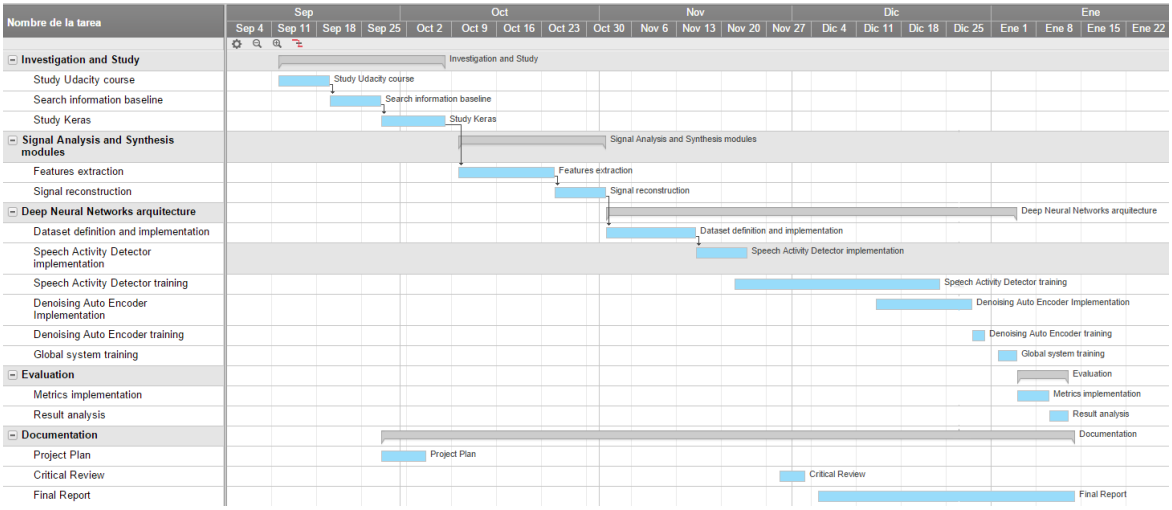

Figure 1.2: Gantt Diagram of the Degree Thesis

1.5 Incidents and Modification

This thesis is a complex project, so we had to face a number of incidences during these months. The first major incident was produced during the creation of the database. It had to be created from two different databases, QUT-NOISE [4] and Timit [9], and we found difficulties in running the code due to an error in the internal paths of the scripts.

The second incident, was also associated with the database. Because of the big amount of data, the time processing the entire database was larger than expected. The same issue, cause that the entire Neural Network architecture designed with the *"fit"* function, which will be explained in Section 3.3.1, be unable to train because of the memory overflow. This led to a redesign of the entire architecture, using *generators*, which are in charge to provide the network batches of data that are loaded directly from disk, in order to be able to train the model.

A third incident would be related to the complex implementation of the system where small errors are very difficult to find and it is very expensive in terms of time dedication.

Finally, the scope had to be readjusted due to the accumulative delays.

Once have made the introduction, in the next chapter we will talk about the **State of the art of Speech Enhancement and Deep learning techniques** currently used in audio processing. In the third chapter we will present the **Methodology** for Speech Enhancement based on Deep Learning, that was followed to do this project. Next, we will present the obtained **Results** in the forth chapter and the expected **Budget** required to implement this project in the fifth. Finally, in the sixth chapter we will present the **Conclusions** of the thesis.

Chapter 2

State of the art of Deep Learning techniques and Speech Enhancement

In this chapter we will do a brief explanation about why deep learning is so popular nowadays, the areas where it is implemented and some of the most used architectures. Then we will talk about speech enhancement and why it is important. In particular, we will mention some of the algorithms used before the implementing of Neural Networks and why Neural Networks fit very well in speech enhancement.

2.1 Deep Learning

In the recent years, Deep Learning has become very popular in areas as Image Processing, Video Processing, Machine Translation and many others. One of the reason that makes Deep Learning the state of the art of so many activities is the ability to model complex non-linear mapping functions [10]. Therefore, Speech Processing is another area that made Deep Learning the state of the art for many tasks, one them being Speech Enhancement.

To continue explaining how Deep Learning works, first we are going to define Deep Learning. "Deep Learning is a new area of Machine Learning research, which has been introduced with the objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence. Deep Learning is about learning multiple levels of representation and abstraction that help to make sense of data such as images, sound, and text"[6]. Deep Learning consists of Deep Neural Networks, which are in charge of learning from input train data. At the practice, depending on the task, there are several types of Deep Neural Networks, such as Convolutional Neural Networks, Recurrent Neural Networks etc..

The most basic Neural Network is the *fully-connected*, which is composed by a deep network of linear classifiers.

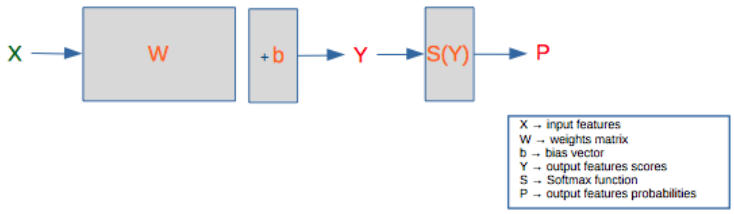

Figure 2.1: Logistic classifier diagram [14]

To understand better how a linear classifier works, Figure 2.1 represents its common architecture. Its equation is expressed in the following manner:

$$Y = WX + b \tag{2.1}$$

The weights and the bias are the key for the network to learn. The aim of the network is to learn the weights and the bias parameters that minimize the error from the training data [14]. The function that measures the error is called loss function. A common loss function is the *Cross-Entropy* or the *Mean Square Error* loss. The first one is more common in classification while the other is used more in regression. In order to minimize the error, an optimizer is requested. A classic optimizer is the *Gradient Descendent*, in particular the *Stochastic Gradient Descendent*. Linear models are stables but, very limited. Therefore, in order to keep the parameters in linear functions and make the entire model non-linear, a step further must be done and introduce non-linearities [14].

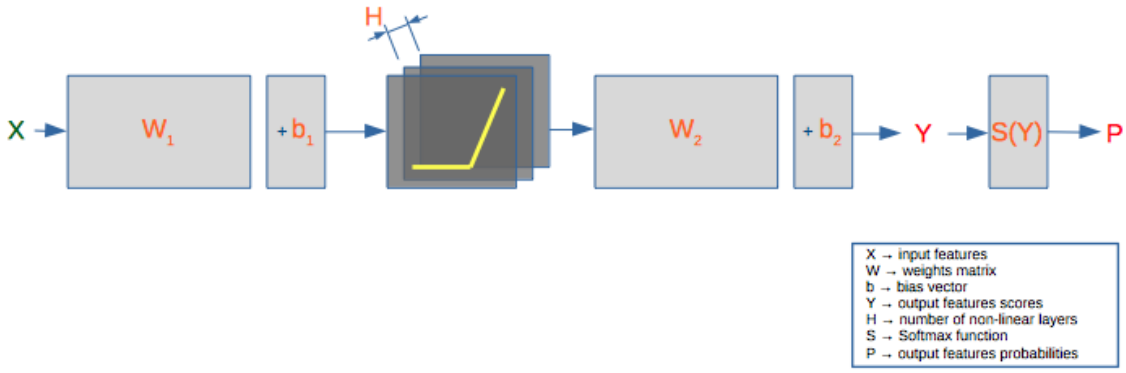

Figure 2.2: 2-layer fully-connected diagram [14]

In Figure 2.2 a fully-connected Neural Network diagram is represented. In order to introduce non-linearity to the model, a non-linear function must be inserted [14]. The most simple non-linear function, which can be observed in Figure 2.2, is the Rectified Linear Units function (ReLU). The number of non-linear functions units is the number of "hidden layers", which is another parameters to tune.

Finally, the concepts of *epoch*, batch size, *iteration*, *learning rate* and *overfitting* must be explained. The epoch is associated to the processing of the entire training dataset before the gradient is recalculated. The *batch*, on the other hand, is associated to calculate the gradient over smaller portions of the dataset, so several iterations will be required before an epoch is completed. This is an advantage, as the system is faster. Therefore, it is required to establish a batch size. As its name indicates, the *learning rate* sets the speed of learning. High *learning rates* increase the error while low *learning rates* increase the *overfitting*. When *overfitting* appears the train should stop. To identify it, the train loss and the validation loss must be observed. When the validation loss is increasing and the train loss is decreasing, it is a clear signal of *overfitting*.

Another class of Neural Networks is the Convolutional Neural Networks(CNN). They are design to exploit spatially-local correlation by enforcing a local connectivity pattern between neurons of adjacent layers [7]. In other words, the inputs of hidden units in layer m, are from a subset of units in layer m-1, units that have spatially contiguous receptive fields as we can see in the figure below:

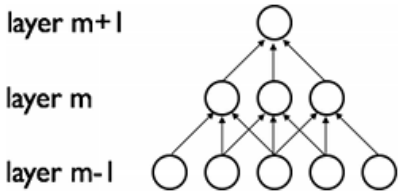

Figure 2.3: CNN exploiting spatially correlation [7]

In addition, each filter *h_i* is replicated across the entire layer. These replicated units share the same parametrization (weight vector and bias) and form a feature map. In Figure 2.4 a CNN feature map can be observed.

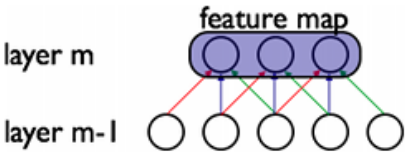

Figure 2.4: CNN feature map [7]

For these reasons, Convolutional Neural Networks are perfectly fit for image and video processing, but also for audio processing.

On the other hand, there are the Recurrent Neural Networks (RNN) which, unlike the Convolutional Neural Networks, are going to extract patterns over time instead of space. This networks are characterized by being trained with sequences of related data and having memory cells.For this reason they are perfect for speech processing task. To understand better how Recurrent Neural Networks works, the Figure 2.5 represent its common architecture.

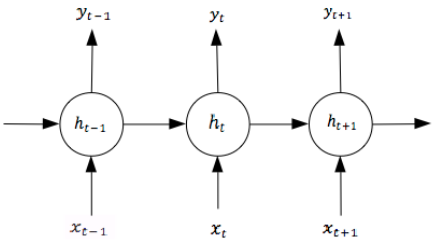

Figure 2.5: Recurrent Neural Network diagram

Given a input sequence $\mathbf{x} = (x_1, ..., x_T)$, a standard Recurrent Neural Network computes the hidden vector sequence $\mathbf{h} = (h_1, ..., h_T)$ and the output vector sequence $\mathbf{y} = (y_1, ..., y_T)$ by iterating the following equations [1]:

$$h_t = H(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{2.2}$$

$$y_t = H(W_{hy}h_t + b_y) \tag{2.3}$$

The W denotes the weight matrices, the b denotes the bias vectors and the H is the hidden layer function, which usually is a non-linearity such as *sigmoid*, *ReLU* or *tanh*.

One of the most efficient implementation of the Recurrent Neural Networks are the *Long Short-Term Memory* (LTSM). The LSTM contains special units called memory blocks in the recurrent hidden layer. As it can be observed in Figure 2.6, memory blocks contain memory cells with self-connections storing the temporal state of the network. In addition, they also contain special multiplicative units, called gates, to control the flow of information. Furthermore, an input and an output gate can be found in a memory block. The input gate controls the flow of input activations into the memory cell, while the output gate controls the output flow of cell activations into the rest of the network. In newer architectures a forget gate was added in order to prevent the LTSM from processing continuous input streams that are not segmented into sub-sequences. The forget gate scales the internal state of the cell before adding it as input to the cell through the self-recurrent connection of the cell, therefore adaptively forgetting or resetting the cell's memory. The latest LTSM architectures contain *peephole connections* which connect the internal cells to the gates in the same cell in order to learn the precise timing of the outputs [8].

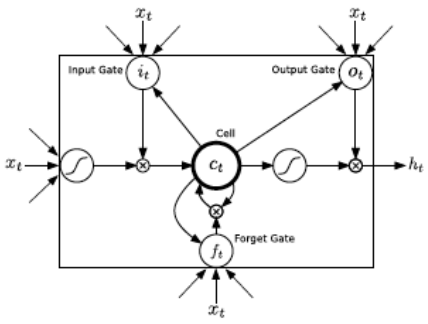

Figure 2.6: LTSM memory block [1]

2.2 Speech Enhancement

For several decades, Speech Enhancement has been the research to enhance noisy speech signals that are corrupted by additive noise, multiplicative noise or convolutional noise[2, 13, 17, 16] .After many years of study, it still a very challenging problem, because most papers rely on estimating the noise during the non-speech activity assuming that the background noise is uncorrelated, non-stationary and slowly varying [2, 13, 17, 16]. Therefore, the noise characteristics estimated in the absence of speech can be used subsequently in the presence of speech, whereas in real time environment such as assumptions do not hold for all the time. [11]

Until very recently the main algorithms used to develop Speech Enhancement systems were broadly classified in four categories:

- Spectral subtractive algorithms. The idea behind this algorithms is estimating the noise spectrum during non-speech regions and subtracting it from the speech.[10]
- Statistical-model-based algorithms. They are based on the principle of stochastic estimation

(Minimum Mean Square Error, Maximum Likelihood Estimation, etc.) of the magnitude of Speech Spectrum.[10]

- Subspace algorithms, assuming the clean signal to be a subspace of noisy signal.[10]
- Binary Masking algorithms. Their goal is to eliminate certain frequency bins by applying a threshold on the Signal-to-Noise Ratio (SNR).In order to achieve it, firstly, a binary mask must be applied to the time-frequency representation of the noisy signal.[10]

During the last years new approaches emerged, basing the algorithms on Deep Learning. In Speech Processing, the Neural Network architectures adopted for denoising are refereed to, as *Denoising Auto-Encoders* (DAE). A DAE is Neural Network which attempts to map noisy inputs to their clean versions [10].It has proved to be more robust and provide better results in terms of subjective and objective evaluations [10]. Recurrent Neural Networks are also used for denoising task. The architecture based on Recurrent Neural Networks are called *Recurrent Denoising Auto-Encoders* (RDAE) and they have shown significant performance exploiting the temporal context information in embedded signals . However, the best Deep Neural Networks applied for denoising task are the Long Short-Term Memory (LTSM) [10, 2, 13].

For this thesis we adopted a recent proposal [10] to enhance speech signals using Deep Learning. In the following chapter we will present the development required to implement a Deep Neural Network architecture capable of eliminating noise from speech signals.

Chapter 3

Speech Enhancement using Deep Learning development

This chapter describes the required stages to implement an architecture capable of mapping noisy speech signals to their clean versions, using Deep Learning. Firstly, we will explain the dataset used to train and test the models. Then the modules which prepare the data for the Neural Network will be introduced. Finally, we will present the proposed Neural Network architecture.

3.1 QUT-NOISE-TIMIT Dataset

In order to train and test the system, QUT-NOISE-TIMIT dataset has been used [5]. It is build from to different databases, QUT-NOISE[4] corpus and Timit corpus [9]. QUT-NOISE is used to provide the background noise, while Timit is providing the clean speech.

In order to provide a simulation of noisy speech in a wide variety of typical background noise conditions, QUT-NOISE corpus provides ten locations in five different scenarios:

- **CAFE:** The two locations of the CAFE scenario were a typical outdoor cafe environment (CAFE-CAFE) and a typical indoor shopping centre food-court (CAFE-FOODCOURTB) [5, 4].
- **HOME:** The two locations for the HOME scenario were in the kitchen (HOME-KITCHEN) and living-room (HOME-LIVINGB) during typical home activities [5, 4].
- **STREET:** The two locations for the STREET scenario were at the roadside near typical inner-city (STREET-CITY) and outer- city (STREET-KG) traffic-light controlled intersections [5, 4].
- **CAR:** For the CAR scenario the noise was measured in the car with the windows down (CAR-WINDOWNB) and with the windows up (CAR-WINUPB) [5, 4].
- **REVERB:** The two locations for the REVERB scenario were an enclosed indoor pool (REVERB-POOL) and an partially enclosed carpark (REVERB-CARPARK) [5, 4].

The duration of each file of noise can be observed in the following table:

Noise	Duration (min)
CAFE-CAFE 1	42:12
CAFE-CAFE 2	39:33
CAFE-FOODCOURTB 1	35:04
CAFE-FOODCOURTB 2	32:05
HOME-KITCHEN 1	33:10
HOME-KITCHEN 2	50:45
HOME-LIVINGB 1	40:51
HOME-LIVINGB 2	40:33
STREET-CITY 1	32:49
STREET-CITY 2	33:06
STREET-KG 1	56:58
STREET-KG 2	47:51
CAR-WINDOWNB 1	44:35
CAR-WINDOWNB 2	43:18
CAR-WINUPB 1	43:03
CAR-WINUPB 2	47:35
REVERB-POOL 1	38:22
REVERB-POOL 2	36:22
REVERB-CARPARK 1	42:10
REVERB-CARPARK 2	38:29

Table 3.1: Duration noise files in QUT-NOISE dataset

The final QUT-NOISE-TIMIT corpus consists of 600 hours of noisy speech sequences created by mixing 10 hours of noise with TIMIT clean speech [10]. It is classified into three categories based on the amount of noise:

- Low noise: SNRs of 15dB and 10dB
- Medium noise: SNR of 5dB and 0dB
- High noise: SNR of -5dB and -10dB

For the Neural Network architecture training, only the low noise subset is considered (15dB and 10dB).

3.2 Audio Analysis and Synthesis

The first stage of the project consists in implementing the Analysis and Synthesis modules which allows to extract the features necessary for training and testing the architecture.

The Analysis Module takes the wav files from database directory and extract their spectrogram magnitudes and phases. The log-power spectrum is used as a input for the Neural Networks

models, while the phase will be used in the Synthesis Module, in order to reconstruct the wav signal. The transform used for this task was the *Short Time Fourier Transform* (STFT). The formula for the STFT is the following:

$$X_l[k] = \sum_{n=-N/2}^{N/2-1} w[n]x[n+lH]e^{(-j2\pi kn/N)} \tag{3.1}$$

where l is the frame number, w is the analysis window and H is the hoop size. In order to implement the STFT the code provided by the "sms-tools" [12] has been used.

To windowing the signal, *Hamming* window has been used, with a length of 200 samples. The hoop has been established in 100 samples, and the sampling frequency in 16 kHz [12]. In Figure 3.1 it can be appreciated the windowing effect over an audio input signal.

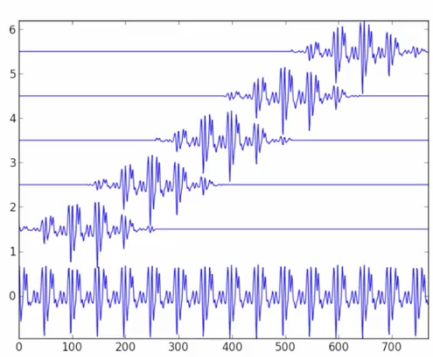

Figure 3.1: Windowing effect over the audio signal [12]

The entire analysis of a frame can be observed in Figure 3.2.

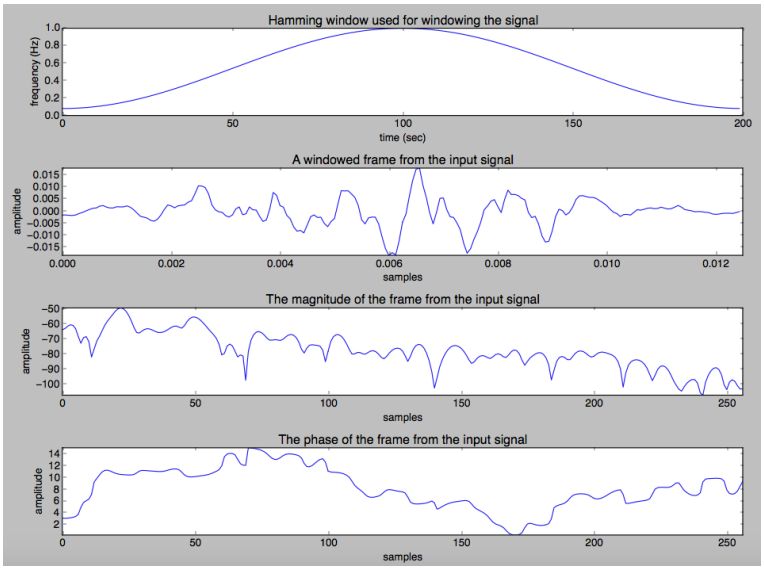

Figure 3.2: Analysis process of a frame

The Synthesis Module consists in reconstructing wav file from the clean speech magnitude obtained at the output of the Neural Network and the phase previously calculated in the Analysis Module [12].

In order to accomplish it the *Inverse Short Time Fourier Transform* (ISTFT) has been implemented. The formula for the ISTFT is the following:

$$y[n] = \sum_{l=0}^{L-1} Shift_{lH,n}[1/N \sum_{k=-N/2}^{N/2-1} X_l[k] e^{j2\pi kn/N}] \tag{3.2}$$

Shift operation must be implemented to undone the overlapping process during the windowing in the Analysis module. Therefore, the formula to obtain each output frame is:

$$yw_l[n] = x(n + lH)w[n] \tag{3.3}$$

and the output sound is:

$$y[n] = x[n] \sum_{l=0}^{L-1} w[n - lH] \tag{3.4}$$

To implement the ISTFT, the code provided by the "sms-tools"[12] has been reused.

3.3 Proposed Deep Neural Network architecture

This thesis implements a 2 stage Deep Neural Network architecture for denoising and speech enhancement in noisy environments. Previous studies have shown that providing speech detection along input have yielded performance benefits [10]. The architecture includes a Speech Activity Detection DNN (SAD-DNN) followed by a Denoising Auto Encoder (DAE). The Speech Activity Detection DNN will provide the speech and non-speech region markings, which enable the Denoising Auto Encoder to implicitly compute the underlying noise statistics over the nonspeech regions, and exploit them in order to improve the denoising process.

3.3.1 Keras

Keras is a high-level neural networks library, written in Python and capable of running on top of either TensorFlow or Theano[3]. It allows a fast and easy implementation of Neural Networks architecture due to its total modularity, minimalism, and extensibility[3]. Furthermore, it support convolutional networks, recurrent networks and combinations of both, including multi-input and multi-output training[3]. It can run seamlessly on CPU and GPU[3].

The first variable that affects the implementation with Keras is whether it has multi-input and multi-output or not. For complex models with multi-input and multi-output *Keras Functional API*

is the best choice to start implementing. Otherwise, for simple implementations the *Sequential* model should be enough. Besides Keras Functional API and the Sequential model use a different syntax, the implementation is similar.

The first step consists in defining the model and adding the layers, specifying the type (Dense, SimpleRNN, LTSM, GRU, etc.), the input and output dimensions and the activation function (relu, sigmoid, tanh, etc.) which introduce the non-linearity. For complex architectures, where the NN has at the input the output of another one, it is required a previous load of weights. To concatenate both models the *"merge"* function is a very useful tool[3].

The next step consists in compiling the model using the function *"compile"*. Here is where the optimizer and the loss function are declared[3].

Finally, we proceed to train the Neural Network. For small amount of data, *"fit"* [3] function is the easiest method to train. It takes as an argument the entire input data (source) and output data (target), which is loaded in RAM memory, the batch size, the number of epochs and the validation dataset (source and target). However, Deep Neural Networks require very big dataset which can not be loaded in RAM memory. The function provided by Keras in this situation is the *fit_generator* [3]. Unlike the *fit*, the *fit_generator* requires the implementation of a *Generator* for both, training and validation dataset. The aim of the *Generator* is to load batches directly from disk and feed the Neural Network. The output of the *Generator* is a tuple, corresponding to the source and target samples.

Once the Neural Network is trained, it is evaluated using a different dataset, the test dataset. To do the prediction, Keras provides the *predict* [3] function, in case the network was trained with *fit*. For big testing datasets, *predict_generator* [3] is more adequate. It requires the implementation of a *Generator* as well[3].

3.3.2 Speech Activity Detection DNN

As it is mentioned previously, the Deep Neural Networks is implemented in 2 stages. The first stage is comprised by a Speech Activity Detection DNN (SAD-DNN), which will predict 0 for the non-speech regions and 1 for the speech regions . It consists of a 2 layer fully connected Neural Network, as it can be observed in Figure 3.3 . In order to tune the parameters of the model, a *Rectified Linear Unit* (ReLU)[3] has been used as an activation function for the first hidden layer, with a dropout of 0.5. For the output layer, the activation functions has been set with a *Sigmoid* function. To measure the loss, the *Binary Crossentropy* [3] function has been set with an *Adam* optimizer [3]. Due to the dimensions of the database, the *fit_generator* and *predict_generator* functions have been used to train and test the model. They required the implementation of generator functions which were in charge to feed the network with batches of 257 features which corresponds to a audio frame.

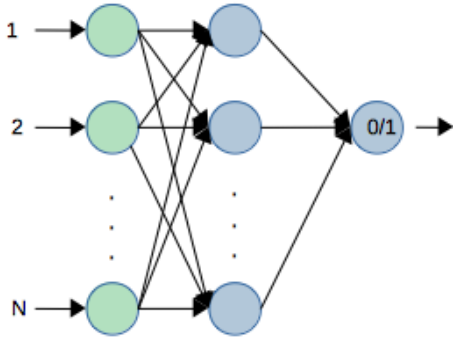

Figure 3.3: Speech Activity Detection NN diagram

The parameters used to implement the Speech Activity Detection DNN can be observed in the following table:

Parameter	**Value**
Dense 1 output_size	257
Dense 1 input_size	257
Dense1 Activation	relu
Dropout 2	0.5
Dense 1 output_size	1
Dense1 Activation	sigmoid
Optimizer 1	Adam
Learning rate 2	0,002
Epochs	50

Table 3.2: SAD-DNN configuration

In Figure 3.4 the evolution of the training session can be observed. Both, training loss and validation loss have an abrupt slope the first 5 epochs, which then tends to be constant. Analysing Figure 3.4, the training could had continued for several epochs before reaching overfitting.

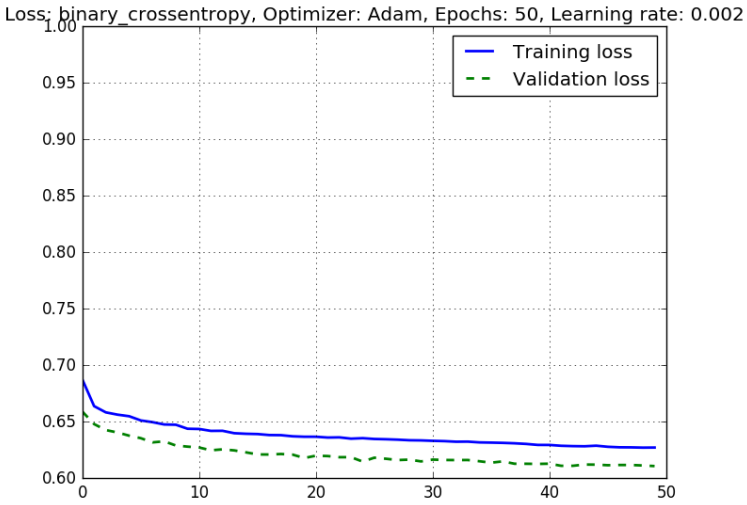

Figure 3.4: SAD training evolution

3.3.3 Denoising Auto Encoder

The second stage of the Neural Network architecture is comprised by a Denoising Auto Encoder (DAE), which attempts to map the noisy inputs to their clean versions. For this stage the output of the Speech Activity Detection DNN becomes an additional input to the denoisning DNN together with the noisy signal [10]. Furthermore, the loss function has also been modified, in this case the *Mean Square Error* (MSE) [3] is considered to be more adequate for this task. The other parameters remain the same as in the case of Speech Activity Detection DNN.

The parameters used to implement the Speech Activity Detection DNN can be observed in the following table:

Parameter	**Value**
Dense 1 output_size	258
Dense1 Activation	relu
Dropout 2	0.5
Dense 1 output_size	257
Dense1 Activation	linear
Optimizer 1	Adam
Learning rate 2	0,002
Epochs	60

Table 3.3: DAE-DNN configuration

3.3.4 Joint training

In order to train the full architecture, previously, the training of the SAD-DNN as an independent DNN has been required. In this manner the SAD-DNN is optimized for its task and then it

is concatenated and retrained jointly with the Denoising Auto Encoder. To concatenate the two architectures, the Functional API from Keras was a perfect tool. In Figure 3.5 the entire Neural Network architecture can be observed.

Finally, to validate the models, 25% from the training dataset has been saved for validation.

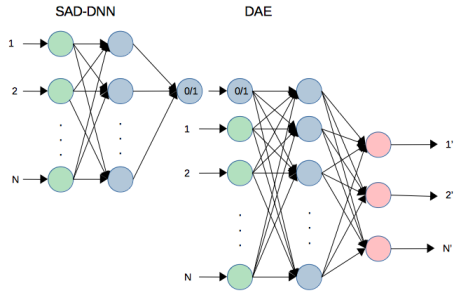

Figure 3.5: SAD-DNN and DAE joint diagram

The general system diagram can be observed in Figure 3.6 .

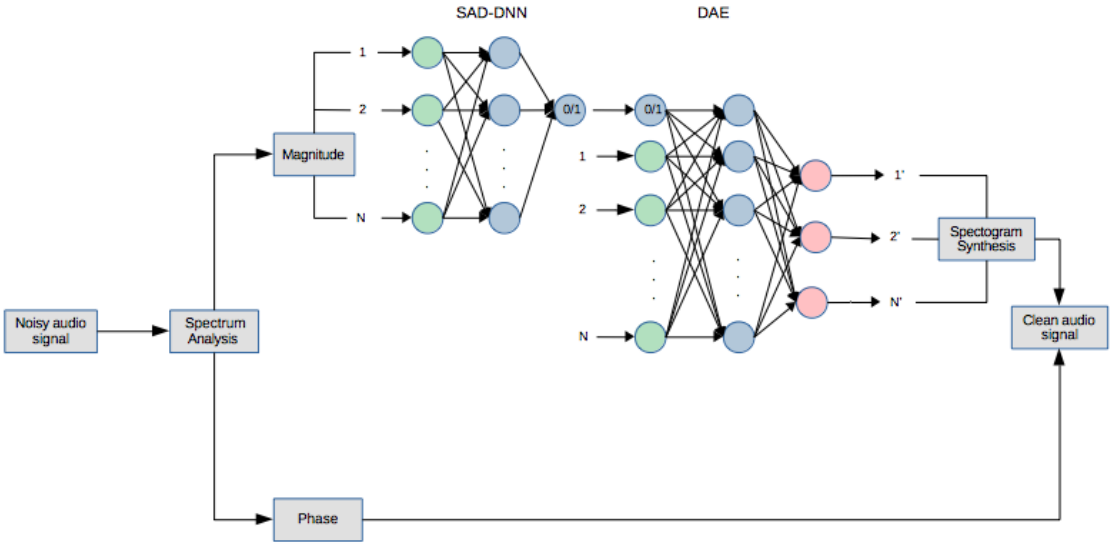

Figure 3.6: System diagram

In the following chapter we will explain the methods used to evaluate the system and present the obtained results.

Chapter 4

Results

To evaluate our system, objective and subjective measurements were necessary to implement. Recent researches use *Noise Reduction* (NR), *Speech Distortion* (SD) and *Perceptual Evaluation of Speech Quality* (PESQ) as measures[10, 2, 13]. Therefore, in this thesis we have adopted the same measurements in order to evaluate our system.

4.1 Evaluation Measures

As we stated previously, *Noise Reduction* (NR), *Speech Distortion* (SD) and *Perceptual Evaluation of Speech Quality* (PESQ) are all well adopted in the speech enhancement community[10].

The *Noise Reduction* (NR) measures the amount of noise which was eliminated from the input signal. It is expressed by the following equation:

$$NR = \frac{1}{N \times d} \sum_{i=1}^{N} |Y_i' - X_i| \tag{4.1}$$

The *N* denotes the number of testing samples, the *d* denotes the features dimensions, *Y'_i* is the enhanced output from the Neural Network architecture, while *X_i* represents the noisy input features.

Another measure we have implemented is the *Speech Distortion* (SD), which measures the distortion in the output signal and it is expressed by the following equation:

$$SD = \frac{1}{N \times d} \sum_{i=1}^{N} |Y_i' - Y_i| \tag{4.2}$$

The only difference between both formulas is in the argument that is subtracted from the enhanced output, Y_i'. Therefore, for the SD equation, the the noisy input features, X_i, are substituted by the clean speech features, Y_i, which are the target of the Neural Network.

Both measures are related in a way that increasing the Noise Reduction, increases the Speech Distortion as well. Big Speech Distortion is undesirable, therefore the goal is achieving the maximum Noise Reduction with a distortion that do not affect the clean voice signal.

Finally, the *Perceptual Evaluation of Speech Quality* (PESQ) measure has been used. It is a complex algorithm that provide a speech-quality score based on the comparison between speech signal and the signal under test. The PESQ value try to predict the Mean Opinion Score (MOS) value. It is measured with a value between the interval -0.5 and 4.5, where -0.5 corresponds to a poor quality, while 4.5 corresponds to a high quality[10].

4.2 Experimental Results and Discussions

To evaluate the system, low (SNR of -10dB and -5dB) and high (SNR of 10dB and 15dB) noise environments have been considered. All three measures previously explained, are presented in Table 4.1.

SNR	**Noise Reduction**	**Speech Distortion**	**PESQ**
-10dB	6.9648	3.3828	2.2530
-5dB	7.1887	3.3377	2.1920
+10dB	5.7338	2.8442	2.2833
+15dB	5.4672	2.7584	2.0994

Table 4.1: Performance evaluation of Speech Enhancement using Noise Reduction, Speech Distortion and PESQ

Although the Noise Reduction coefficient is higher for SNR of -5dB compared to SNR of -10dB, it can be observed that Noise Reduction coefficients increase their value as the noise increases in the speech signals. A possible explication could be the dimension of the test dataset used to do the measurements, which should be bigger. Analysing the Speech Distortion coefficients it can be confirmed that they have higher values as the Noise Reduction increases.

The PESQ coefficients on the other side, have provided unexpected values. PESQ coefficient for SNR of +15dB should be the highest as long as it has the lowest noise level. Considering that PESQ is measured between -0.5 and 4.5, all the obtained coefficients are above the average, but there is certainly room for improvident.

In terms of subjective measurements however, SNR of +15dB provides higher intelligibility than SNR of +10dB. Speech with SNR of -10dB and -5dB are not intelligible.

Once we have explained the obtained results, in the next chapter we will present the budget required to develop this project.

Chapter 5

Budget

This chapter is a guidance for the budget required to implement this project.

This project has been developed using the resources provided by the group VEU of UPC. For the development, computational resources and equipment were required. A laptop was needed to develop and a high performance GPU to train the Neural Networks.

The main costs of the project comes from the salaries of the researchers and the time spent on it. It will be considered that my position has been as Junior Engineer, while my project supervisor who was advising me have the salary of a Senior Engineer.

It will also be taken in account other expenses such as the local or the Internet.

It will be considered that the total duration of the project has been 18 weeks, as depicted in the Gantt diagram in Figure 1.2.

	Amount	**Wage/hour**	**Dedication**	**Total**
GeForce GTX TITAN	1	- €/h	18 weeks	1,500 €
Junior engineer	1	12,00 €/h	30 h/week	6,480 €
Senior engineer	1	80,00 €/h	3h/week	4,320 €
Computer	1	- €/h	18 weeks	600 €
Internet	1	- €/h	18 weeks	250 €
Local	1	- €/h	18 weeks	1350 €
			Total	14,500 €

Table 5.1: Budget of the project

Chapter 6

Conclusions and future development

The main objective of this project was to implement a system capable of enhance noisy speech signals using a Deep Neural Network architecture. As it is explained in Section 1.5, unfortunately, several incidents occurred during the development of the project that lead to obtain results which do not fulfil our expectations.

However, it is important to remark that the preparation of the datasets and the implementation of both, the Analysis and Synthesis Module have successfully been achieved. This should reduce the effort for a future development of these kind of tasks. Additionally, all the programs that load the training data, optimize the parameters, have been prepared. Future test and evaluation of more complex models will require relatively little effort in terms of development, however the time required for the parameter estimation will remain elevated.

As possible modification for a future development, in the original design implementation, it was added left and right context to the magnitude features which feed the Neural Network in order to obtain a better labelling of voice and unvoiced frames in the Speech Activity Detector, but also to help the Denoising Auto-Encoder with the denoising task [10]. Furthermore, replacing the fully-connected layers from the Denoising Auto-Encoder by a LTSM and a feed-forward neural network [10] could have been a good solution to improve the PESQ.

Another possible modification for a future development in order to improve the perceptual quality, may be the replacement of the *Mean Square Error* loss function, which is used in the Denoising Auto-Encoder, by a perceptually loss function [10]. As a last idea, our system can be used in conjunction with a *Weighted Denoising Auto-Encoder* [15].

Bibliography

[1] Abdel-rahman Mohamed Alex Graves and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. http://www.cs.toronto.edu/~fritz/absps/RNN13.pdf, 2013. Last accessed 7 January 2017.

[2] Dinei Florencio Anurag Kumar. Speech enhancement in multiple-noise conditions using deep neural networks. In *Proc.Interspeech Conference*, pages 3738–3742, 2016.

[3] Francois Chollet. Keras. https://github.com/fchollet/keras, 2015. Last access 11 January 2017.

[4] Robert Vogt Michael Mason David Dean, Sridha Sridharan. Qut-noise corpus. https://www.qut.edu.au/research/research-projects/speech-audio-image-and-video-technology-saivt, 2010. Last accessed 7 January 2017.

[5] Robert Vogt Michael Mason David Dean, Sridha Sridharan. The qut-noise-timit corpus for the evaluation of voice activity detection algorithms. https://eprints.qut.edu.au/38144/, 2010. Last accessed 7 January 2017.

[6] Li Deng and Dong Yu. Deep learning: Methods and applications. *Foundations and TrendsÂ® in Signal Processing*, 7(3-4):197–387, 2014.

[7] Laboratoire d'Informatique des Systèmes Adaptatifs. Convolutional neural networks. http://deeplearning.net/tutorial/lenet.html, 2016. Last accessed 7 January 2017.

[8] Andrew Senior Haşim Sak and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Proc. Interspeech Conference*, 2014.

[9] William Fisher Jonathan Fiscus-David Pallett Nancy Dahlgren Victor Zue John Garofolo, Lori Lamel. Timit corpus. https://catalog.ldc.upenn.edu/ldc93s1, 1993. Last accessed 7 January 2017.

[10] Panayiotis Georgiou Prashanth Guruntah Shivakumar. Perception optimized deep denoising autoencoders for speech enhancement. In *Proc.Interspeech Conference*, pages 3743–3747, 2016.

[11] R. Shantha Selva Kumari S. Selva Nidhyananthan and A. Arun Prakash. A review on speech enhancement algorithms and why to combine with environment classification. https://www.worldscientific.com/doi/pdf/10.1142/S0129183114300024, 2014. Last accessed 7 January 2017.

[12] Xavier Serra. Audio signal processing for music applications. https://www.coursera.org/learn/audio-signal-processing, 2016. Last accessed 7 January 2017.

[13] Li-Rong Dai Chin-Hui Lee Tian Gao, Jun Du. Snr-based progressive learning of deep neural network for speech enhancement. In *Proc.Interspeech Conference*, pages 3713–3717, 2016.

[14] Vincent Vanhoucke. Deep learning course. https://www.udacity.com, 2016. Last accessed 7 January 2017.

[15] B. Xia and C. Bao. Speech enhancement with weighted denoising auto-encoder. In *Proc.Interspeech Conference*, pages 3444–3448, 2013.

[16] Shigeki Matsuda Chiori Hori Xugang Lu, Yu Tsao. Speech enhancement based on deep denoising autoencoder. `https://www.researchgate.net/publication/283600839_Speech_enhancement_based_on_deep_denoising_autoencoder`, 2013. Last accessed 7 January 2017.

[17] Li-Rong Dai Yong Xu, Jun Du and Chin-Hui Lee. An experimental study on speech enhancement based on deep neural networks. In *IEEE SIGNAL PROCESSING LETTERS*, volume 21, pages 65–68, 2014.