UNIVERSIDAD VERACRUZANA

FACULTAD DE LA CONTRUCCIÓN Y EL HÁBITAT

REGIÓN VERACRUZ

“AN ARTIFICIAL NEURAL NETWORK FOR UNDERWATER ACOUSTIC CLASSIFICATION ON MARINE VESSELS”

TESIS

PARA OBTENER EL TÍTULO DE

MAESTRO EN INGENIERIA APLICADA

PRESENTA

JOSUE MOISÉS AGUILAR GARRIDO

DIRECTOR

DR. LUIS FELIPE MARÍN URIAS

Boca del Río, Veracruz 2019

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

ABSTRACT

Classification using different kind of machine learning techniques is a trend in the computer science area, and this is due to the increasing accuracy and speed for recognizing patterns the computers are gaining, and the huge practical applications those algorithms may have.

For this thesis work, it will be shown the implementation and the results of an artificial neural network for classification purposes, by using the audio signals of four different kinds of maritime vessels, that were obtained with the use of a particular underwater acoustic acquisition system that was proposed in this project also.

The implementation methodology was the following, first to collect acoustic data of four classes of maritime vessels, later preprocess all the data in order to get a better dataset for training the neural network (it was proposed using the spectrogram and the mean of the spectrogram of the audios), then train the neural network with the backpropagation algorithm with the preprocessed audios, and finally evaluating and comparing the results of the classificator, for the different neural network topologies proposed.

A special emphasis in this methodology is on the underwater acoustic acquisition system, and the preprocessing of the audio combined with the robustness of the artificial neural network implementation, which have leaded to a successful classification rate of 100%, for all dataset proposed in this work.

Keywords: Underwater Acoustics, SONAR, Machine Learning, Power Spectral Density, Artificial Neural Network, Classification, Digital Signal Processing.

1 INTRODUCTION

Sonar is a technique/system based on sound propagation that is used to navigate, communicate with or detect other vessels, and usually the research work in that area focuses on transducers, acoustic signal processing, acoustical oceanography, bioacoustics, micro-computers, and human/machine engineering. The sonar system basically consists in a hydrophone array, preamplifiers, sound emission (for active sonar), analogue to digital converters, beamformer processor, and a console for displaying the sonar plots. For this thesis work, it was proposed a new methodology to extract the acoustic signature from maritime vessels using a single cylinder-shaped hydrophone (instead of a complete array) combined with an own-designed acquisition system, and it was proposed also an implementation of a computational neural network in order to classify four kind of maritime vessels to finally observe the results for the different proposed topologies and give some conclusion of the whole process.

The acoustic signature is basically the noise a vessel creates, it is a distinctive characteristic that could be used to classify a vessel. One common tool for the underwater maritime surveillance is the passive sonar, with that tool it is possible to know the direction of arrival of the acoustic waves a marine vessel may generate, while they are on operation, and also it is possible to get and listen the particular acoustic signature for every kind of vessels. Even though, for this project the collecting task of the acoustic signals was not made with a conventional sonar system, the results of this project can be

extended/implemented in a proper sonar system, because the classifications algorithms are based in a single acoustic signature, and the proposed acquisition system for this work is able to get a single acoustic signature. So, there is no need of all the data that a conventional sonar is able to provide (at least for this thesis project), and the acoustic data retrieved from a single hydrophone for this case is enough in order to design/test the artificial neural networks proposed.

1.1 Project Justification

Detection, tracking and classifying marine vessels of all sizes, approaching ports and harbors and others special geographical areas, is an imperative aspect to the security of any country who has access to the sea. As can be seen see in the section in 2.1, there have been a lot of efforts studying this problem in different countries and with different approaches. It is not hard to see the importance for the law enforcement departments and defense institutions worldwide, to assure surveillance, not just in air and earth and on the surface of the seas, even in underwater environments, and so far, the Sonar is the best tool to detect, track and classify vessels for underwater vessels.

Marine transportation plays a vital role in the global economic viability of Mexico. As a maritime nation, Mexico depends on a strong commercial maritime industry that is tied to maritime security and its stability. Ports and affiliated transportation are all part of a complex system and are potential targets, with widescale disaster implications. A need to detect, track and classify vessels of all sizes approaching our ports and harbors is imperative for the security of this country and its complex maritime systems (Miguel Alvarado, 2017).

This work is an application of the passive acoustic method for vessel classification using an artificial neural network. The analysis of noise radiated by passing vessels close to the hydrophone (acoustic sensor), provides sound signatures and specific acoustic features. Those features are then implemented into a decision-making algorithm used for final classification, with the chief end of enhance the sonar operators' capabilities, to decide if there is a potential threat in the area they are performing the detection.

1.2 Hypothesis

It is possible to create an acquisition system for underwater acoustic signals, a classifier algorithm using those acoustic signals for training an artificial neural network, for four kinds of maritime vessels purposes, and verify the accuracy of the results for

different kinds of topologies, which are on function of the input and hidden layers of the neural network.

1.3 Research Objectives

1.3.1 General Objective

Implement an acquisition system and a classifier powered by a multilayer artificial neural network, in order to process correctly the underwater acoustic signals for four different kind of maritime vessels, to finally provide relevant classification information and metrics of the results.

1.3.2 Specific Objectives

•Perform a research related to the state of art of different kind of classifier for maritime vessels, and verify the advantages and disadvantages.

•Develop the hardware and software needed for acquire the underwater acoustic signals. The acquisition must have capabilities of receive data on both, on real-time and with audios files. And must be able to save the audio files into a computer hard disc with the pre-processed audios.

•Start collecting as many acoustic signatures as possible of the following vessels: a mini-submarine (ROV), merchant ships, fishing boats, and motorboats with outboard engines, in order to train and test the proposed artificial neural networks. And this tool also has to be able to create the spectrogram and mean of spectrogram data using the audio signals.

•Develop a software tool that can be used to generate a database with four kind of classes, based on the preprocessed acoustic signature captured previously.

•Develop a software tool that can be used to load a database with the four kind of classes and train an artificial neural network implemented with the backpropagation algorithm, for different kinds of network morphologies and with a customable stop condition for the training process.

•Develop a software tool that can be used to test a trained neural network, and verify the accuracy of the classifications and performance, to finally report conclusions.

2 BACKGROUND

2.1 Previous Works

A lot of efforts have been applied to audio classification focused for human speech recognition like Pocketsphinx, Google's Speech API, and many others. Such applications and services, recognize speech to text with pretty good quality, but none of them can determine different sounds like an urban sound or for this case an acoustic signature of a maritime vessel.

There have been different approaches in this research area, for example one technique that have been used for automatic ship detection is based on the extraction of features in the frequency domain by using the algorithm of Fast Fourier Transform (J-C Di Martino, S. Tabbone, 1994), where a Lofar (Low Frequency Analysis and Recording), was used, in order to get a narrow-band plot of the signals, providing an image of frequency power versus time, more commonly referred to as "lofargram", and finally extract lines from images using statistical likelihood tests. The main application of this technique it is to improve the signal to noise ratio, as an enhancement of the input Lofar image.

An approach for classification of vessels involves techniques that contains different kind of preprocessing algorithms for the audio signal, like DEMON (Detection of Envelope Modulation on Noise). That consists in the principle of noise radiated by a ship, that is modulated at a rate dictated by parameters of the propeller and the engine (number of blades and rotational speed). And a classification algorithm like "decision trees", to study the contrast between two or more probability vectors, with effectiveness. [3] (Talmor Meir, Mikhail Tsionskiy, Dr. Alexander Sutin, 2012)

Another approaches for classification algorithms have been implemented based on Artificial Neural Networks (ANN) and Support Vector Machines (SVM) for vessel recognition, both methods used as input, acoustical signature in the frequency domain, and was achieved an accuracy of 92%. [4] (N.Leal, E. Leal and G. Sanchez 2015).

But not all efforts have been exclusive focused on classification in the frequency domain, in the article [5] (Hamed Komari, Hassan Farsi), presents a hybrid method for process the sound in time and frequency domain, with a Bayesian classificator.

In another research work was compared the performance of four kinds of neural networks classifiers (hyperplane based classifier MLP, kernel based classifier AKC, probabilistic based classifier PNN, exemplar based classifier LVQ). And for the preprocessing and feature extraction stage, was used Two-Pass Split-Windows (TPSW) algorithm, used to extract tonal features from the average power spectral density of the input data. [6] (Chin-Hsing Chen, Jiann-Der Lee).

And there are also many different kinds of patents related to methods of classification of maritime vessels, developed by companies involved in the sonar and defense industry, like the DE102012015638A1, assigned to Atlas Elektronik Co, which describes a "Method and apparatus for classifying vessels" and the DK414786A assigned to Krupp Gmbh which describes a "Method and apparatus for detecting a vessel".

It is clear that a lot of efforts have been applied to this research area worldwide, especially in the defense industry, but so far, there is not a standard way to implement a classifier for maritime vessels, every company and research center have their own approach, and they are always trying to improve the accuracy of the classification, either testing a new classifier algorithm or topology or a new preprocessing method for the audio signals.

2.2 Underwater Acoustic Theory

2.2.1 Sound Propagation

So far, the most efficient way to propagate a physical signal in the ocean or any underwater environment is a sound wave, which can propagate long distances compared with radiofrequency or light signals which are attenuated very fast in the same environment. The behavior of the sound in underwater environments is very important for sonar design and for to know what can be expected when acoustic measurements are performed.

The basic theory of acoustics involves the study of vibration, waves, and their propagation, (Qihu Li, 2011). When an underwater acoustic wave is propagated, some pressure changes are applied to the structure of the water medium, with the result of the spread of the sound.

The propagation velocity *c* of sound in the sea can be derived from the following adiabatic equation

$$c = \left(\frac{\partial p}{\partial \rho} \right)_s = \frac{1}{\rho K_a} \tag{2.1}$$

Where *p* is the sound pressure, ρ is the density of water, and K_a is the adiabatic compression coefficient. In sea water, since ρK_a is a function of temperature, salinity and pressure, the sound of speed in sea water is also a function of temperature, salinity and pressure, but temperature is the dominant factor. The empirical formula for calculating sound speed in sea water is (Qihu Li, 2011)

$$c = 1410 + 4.21t - 0.037t^2 + 1.1S + 0.018d \ (\mathrm{m/s}) \tag{2.2}$$

Where *t* is the temperature of sea water (°C); *d* is the depth (m); *S* is the salinity (0/00). c≈ 1500 m/s when *t*=14°C, *S*=34.5, *d*=15m.

Because the propagation characteristics of sound in the sea strongly depend on the sound speed, it is very important to understand the distribution of sound speed for any specific area. The relation between sound speed and depth is called the sound speed profile (SSP).

2.2.2 Hydrophones

The elements that are used to acquire the underwater acoustic signals are called hydrophones, and those possess the capacity to receive an elastic pressure wave and transduce it to a variation of electrical voltage due to the piezoelectric ceramic properties with which are build.

Those hydrophones are acoustics transducers with the potential of surveying the underwater environment in a wide frequency spectrum span. A hydrophone can also detect signals in air, but due its construction design, that is intended to match its acoustic impedance to water and not to air, its performance in air is poorer due to lower sensitivity.

In the Figure 2.1, it is shown as an example a miniature hydrophone from the Danish company Brüel & Kjær, that hydrophone is usually used as a calibration reference standard, or for noises and cavitation measurements.

Figure 2.1: Hydrophone Type 8103 from Brüel & Kjær.

2.2.3 Low Noise Preamplifiers

Usually, signal levels at the output of the hydrophone is of very low level, sometimes in the range of microvolts of magnitude, so it is necessary to add a stage to amplify that low signal in order to be able to acquire it with an analog to digital converter for the processing of the acoustic signal in the digital domain. A hydrophone is made with a piezoelectric ceramic and its impedance may vary broadly, so it is mandatory to use a preamplifier who can match its impedance with the hydrophone and also be for low noise applications due the nature of the low voltage signals involved.

An ideal preamplifier will be linear (have a constant gain through its operating range), have high input impedance (requiring only a minimal amount of current to sense the input signal) and a low output impedance (when current is drawn from the output there is minimal change in the output voltage). It is used to boost the signal strength to drive the cable to the main instrument without significantly degrading the signal-to-noise ratio (SNR). In the figure 2.2, it is shown an example of hydrophone preamplifier circuit diagram, it was taken from the application note of OPA27/37. The transducer used is piezo type and has a relatively high output impedance, which requires a preamplifier with a high input impedance. The input impedance of this circuit is mainly determined by the value of the resistor R3, 1 MOhms here. The value of other components determines the frequency range used.

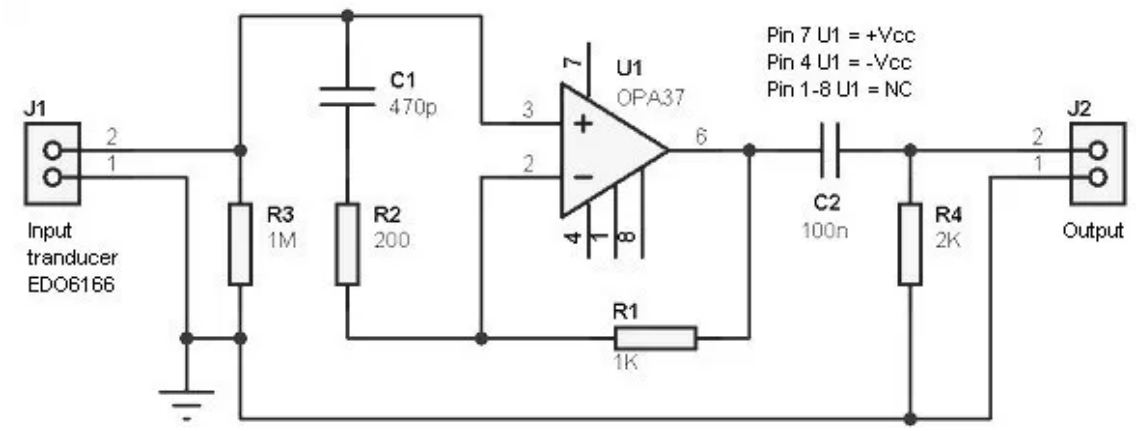


Figure 2.2: Hydrophone Preamplifier with OPA37.

The noise performance of a preamplifier is critical. According to Friis's formula (J.D. Kraus, 1966), when the gain of the preamplifier is high, the SNR of the final signal is determined by the SNR of the input signal and the noise figure of the preamplifier.

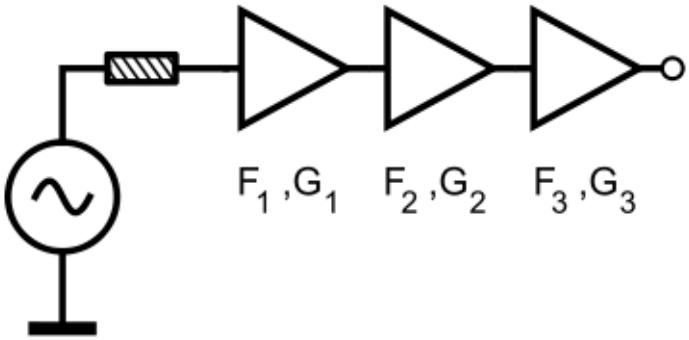


Figure 2.3: Schema of cascades of amplifying stages.

Friis's formula is used to calculate the total noise factor of a cascade of stages (Figure 2.3), each with its own noise factor F_i and gain G_i (assuming that the impedances are matched at each stage). The total noise factor can then be used to calculate the total noise figure. The total noise factor is given as

$$F_{total} = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 G_2} + \frac{F_4 - 1}{G_1 G_2 G_3} + \ldots + \frac{F_n - 1}{G_1 G_2 \ldots G_{n-1}} \quad (2.3)$$

Where F_i and G_i are the noise factor and available power gain, respectively, of the *i*-th stage, and *n* is the number of stages. Note that both magnitudes are expressed as ratios, not in decibels.

An important consequence of this formula is that the overall noise figure of a radio receiver is primarily established by the noise figure of its first amplifying stage.

Subsequent stages have a diminishing effect on signal-to-noise ratio. For this reason, the first stage amplifier in a receiver is often called the low-noise amplifier (LNA).

2.2.4 Hydrophones Array

An omni-directional hydrophone in the ocean environment can determine only if the target is present, but it is impossible to determine the incidental angle of the target, in order to know the direction of arrival of the acoustic signal, better known as bearing of a contact. So, a solution to know the bearing of a contact, is to configure a group of hydrophones together, consisting of a hydrophone array, then it will be possible to determine the incidental angle of the contact. In order to process the hydrophones array it is necessary a signal processor that can handle all the inputs and with some special algorithms can calculate the bearing of the contacts. Target bearing, multitarget resolution or even target ranging is the main capability of a hydrophone array. The key technique in completing these tasks is the beamforming, which steers each hydrophone of the array to the target usually using time delay, and uses time accumulation to obtain a system gain. Even though there are many benefits for using a hydrophone array as it was described previously, the complexity of the system greatly increase, and for this work was proposed a simpler approach, as described in the section 3.1.1.

Figure 2.4: Cylindrical Hydrophone Array [17].

2.2.5 Acoustic Beamforming

Beamforming is the core part of the signal processor of a modem sonar system, regardless of whether it is passive or active sonar. It can be considered as a spatial filter: it can allow the incidental signal from some direction to pass through, but it rejects all interferences and noise from other directions. Beamforming theory has been studied thoroughly since the 1940s, particularly in the field of radar antennas. These studies included the analysis of directivity function, system gain calculation of arrays, optimum weighting, and array baffling (Qihu Li, 2011). Many research results and technical principles in radar theory can be applied to sonar systems with some kind of modification.

In the figure 2.5 shows a diagram with a conventional Beamformer (implemented with the delay and sum method) who has N hydrophones, and for each hydrophone there is time delay line. For the calculation of the delay time for every hydrophone, it is necessary to take in account the physical geometry of the array, the distance between the hydrophones, the speed of the propagation of sound in the water, and the desired incidence angle θ_0. Subsequently, the results for every delay line are added for a specific interval of time, called integration time. And finally, a mathematical operation called "Mean square root" is applied to the previous added signal; the result of the operation is a numerical value that represents the acoustic energy for the beam θ_0. It is necessary to calculate the acoustic energy for several distinct angles, in order to know in which angle was gotten the maximum energy, that means there is a possible contact in that direction of the hydrophones array.

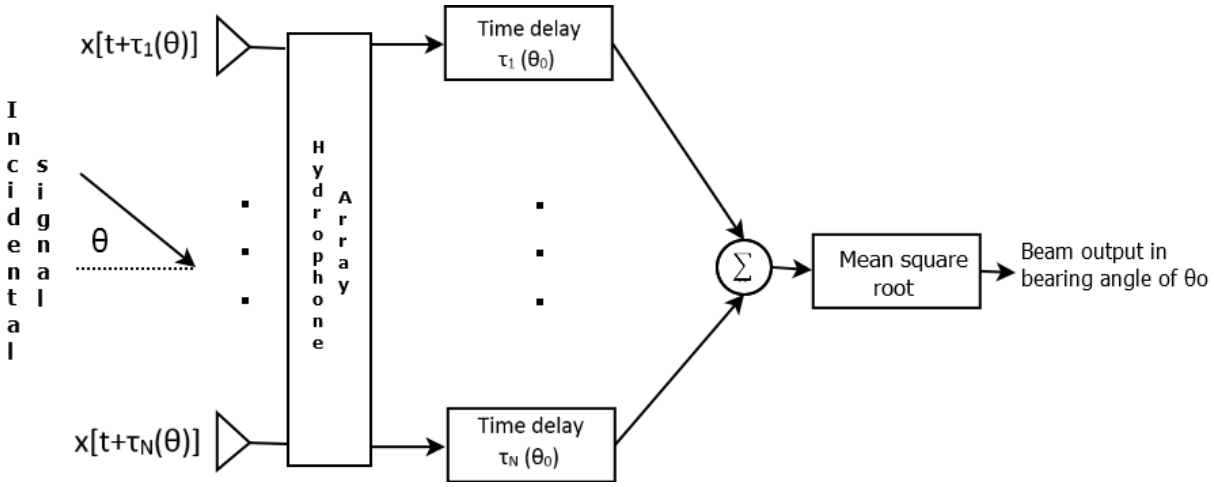


Figure 2.5: Conventional delay and sum Beamformer.

2.3 Basic Theory of Digital Signal Processing.

Digital processing of signals includes data storage, transformation, transmission, etc. Some basic theory and techniques of digital signal processing are introduced in the following points. Including: analog to digital conversion, digital filtering, signals on time and frequency domain.

2.3.1 Digital conversion of Analog signals

All modern sonars and almost all modern technologies for signal processing are performed in the digital domain due to the increased power of the signal processors and the small size, low energy, relative cheap cost and the great implicit advantages that carries working in the digital domain like the implementation of new algorithms in software without the need of new hardware, among others advantages. But all physical signals in nature are in the analog domain, so it is always essential a stage to convert those analog signals to the digital domain.

The figure 2.6 shows a block diagram of the digitalization process of an analog signal. In the first block, there is an "Anti-aliasing filter", its main function is to avoid spectrum distortion due to sampling operations.

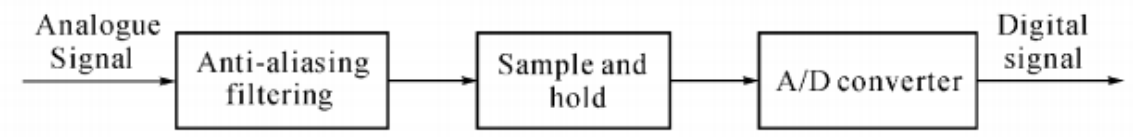


Figure 2.6: Digitization of an analog signal.

The sampling and hold block are related with the samples taken of the analog signal, for example, in the figure 2.7 is shown a sinusoid signal. The signal frequency is f and the sampling duration is T_s. Before sampling, the signal is:

$$x(t) = A\cos(2\pi ft+\theta) \tag{2.4}$$

Where A is the amplitude of the signal and θ represents the phase. After sampling, the signal becomes a discrete time series.

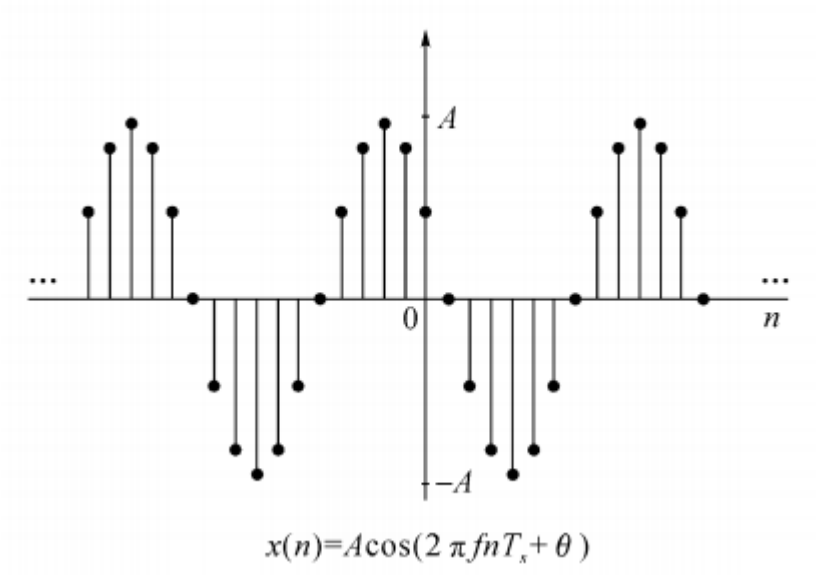


Figure 2.7: Amplitude quantization of an analog signal [1].

The third block is related to amplitude quantization, and it is on charge to choose the digital amplitude value closest to the original analog amplitude. Figure 2.8, shows an example of a 3-bit layer. Normally, the most significant bit of a digital expression is the sign bit (0 indicates positive and 1 indicates negative).

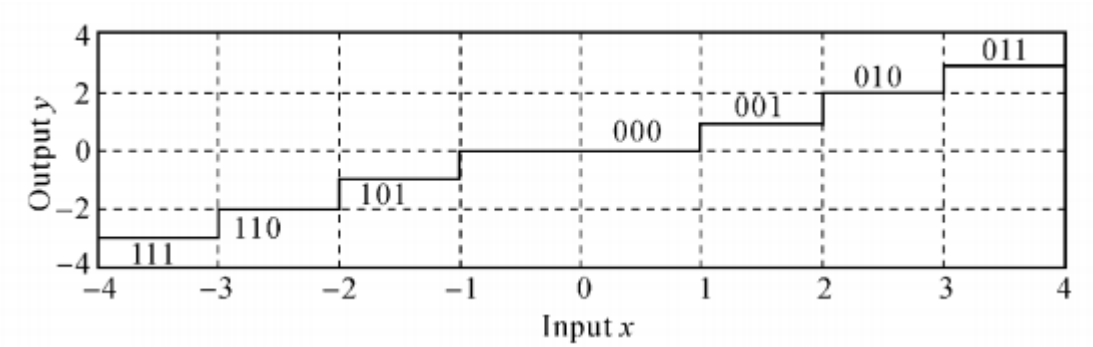


Figure 2.8: Amplitude quantization with signed binary numbers [1].

2.3.2 Digital Filtering of signals

Digital filtering is one of the most important operations for signal processing, and linear filtering is the most popular digital filtering technique. It can be considered as a special kind of digital transformation. In few words, a digital filter is a system that performs mathematical operations on a discrete-time signal to reduce or enhance certain aspects of it. There are some basic filter terms that helps to understand the behavior and are shown in the table 2.1

Filter Type	Behavior
Low Pass	Let low frequency signals pass through, and suppresses high frequency.
High Pass	Let high frequency signals pass through, and suppresses low frequency.
Passband	Range of frequencies passed by a filter.
Stopband	Range of frequencies blocked.
Transition band	In between these.

Table 2.1 Signal filter terms.

A digital filter uses a digital processor to perform numerical calculations on sampled values of the signal. The processor may be a general-purpose computer such as a PC, or a specialized DSP (Digital Signal Processor) chip, or an electronic device as FPGA (Field Program Gate Array), in the figure 2.9 shows one example of the implementation of a low pass digital filter in the processor block, and shows how this algorithm affects the input signal.

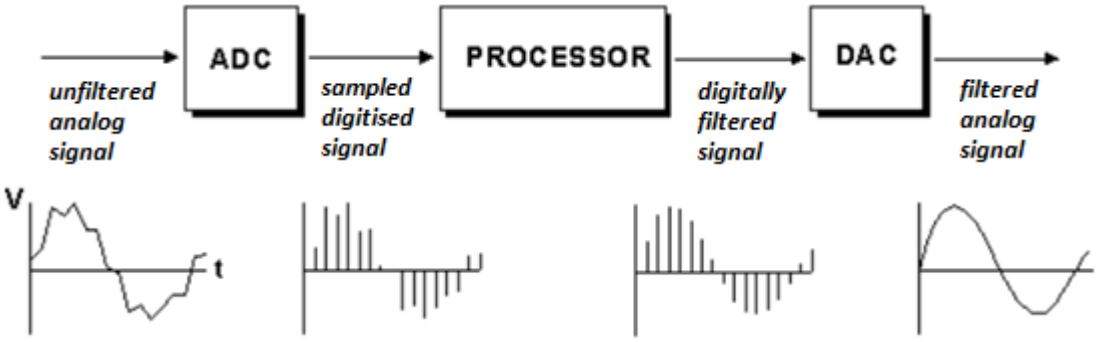


Figure 2.9: Low Pass digital filter example.

2.3.3 Digital signals in time and frequency domain.

The operation and processing of a digital signal involves many characteristics of the signal, including time domain (such as mean value, signal power, correlation function, etc.) and frequency domain (such as amplitude spectrum, power density spectrum, cross power spectrum, etc.).

The time-domain representation gives the amplitudes of the signal at the instants of time during it was sampled. However, in many cases it is needed to know the frequency content of a signal rather than the amplitudes of the individual samples.

The Fourier transform is a mathematical operation used generally to transfer a signal in the time to the frequency domain, so this operator is very useful for analyzing signals in both domains. The mathematical definition of the Fourier transform for a signal *x(t)* in the time domain can be expressed

$$X(f) = \int_{-\infty}^{\infty} x(t) \exp(-2\pi \mathrm{j} f t) \mathrm{d}t \tag{2.5}$$

In the frequency domain, you can separate conceptually the sine waves that add to form the complex time-domain signal. The figure 2.10 shows single frequency components (f_0, $2f_0$, $3f_0$), which spread out in the time domain, as distinct impulses in the frequency domain. The amplitude of each frequency line is the amplitude of the time waveform for that frequency component. The representation of a signal in terms of its individual frequency components is the frequency-domain representation of the signal. The frequency-domain representation might provide more comprehension about the signal and the system from which it was generated.

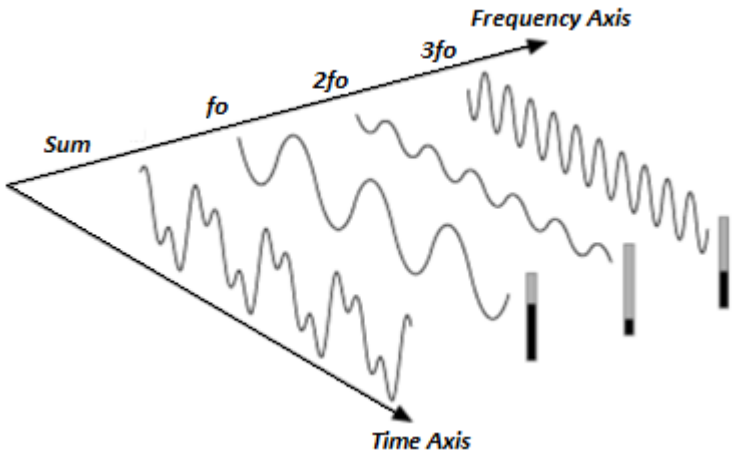


Figure 2.10: Complex time-domain signal vs frequency components.

In modern days, where most of the processing are performed in the digital domain, it is necessary an algorithm who can do the same than equation 2.5 but in the digital domain, so far, the most efficient algorithm to perform that task is named FFT (Fast Fourier Transform) with the Cooley-Tukey or twiddle angle factor implementations.

In the figure 2.11, it can be seen a code example of an implementation of the FFT using Matlab/Octave programming language.

```
clear
N = 16; %FFT size
M = log2(N); %Number of stages
Sstart = 1; %First stage to compute
Sstop = M; %Last stage to compute
Bstart = 1; %First butterfly to compute
Bstop = N/2; %Last butterfly to compute
Pointer = 0; %Init Results pointer
for S = Sstart:Sstop
  for B = Bstart:Bstop
    Z = floor((2^S*(B-1))/N); %Compute integer z
    % Compute bit reversal of Z
    Zbr = 0;
    for I = M-2:-1:0
      if Z>=2^I
        Zbr = Zbr + 2^(M-I-2);
        Z = Z -2^I;
      end
    end %End bit reversal
    A1 = Zbr; %Angle factor A1
    A2 = Zbr + N/2; %Angle factor A2
    Pointer = Pointer +1;
    Results(Pointer,:) = [S,B,A1,A2];
  end
end
Results, disp('    Stage B-fly, A1, A2'), disp(' ')
```

Figure 2.11: Matlab twiddle angle factor implementation [7].

2.4 Typical Sonar Plots.

When using a sonar there are some minimum tasks that need to be implemented in that system in order to be considered a sonar, for example the ability to detect direction of arrival of an acoustic source, to detect the acoustic signature, and the range(distance). In the following points of this section 2.4 is going to be described the typical ways of represent that retrieved information.

2.4.1 BTR (Bearing Time Record)

This plot is used to know the direction of arrival of the acoustic signal in function of time, it allows to track the change of the acoustic contact using a waterfall plot. Every single line of the waterfall corresponds to one sample on a specific time of the resulting energy for every single beam implemented in the beamformer module (bearing), as shown in figure 2.5. An example of the Bearing Time Record plot can be seen in Figure 2.12,

where the white pattern represents the tracked bearing for 100 seconds, and can be observed of that figure, that the detected bearing started in 110° approx. and later the contact it moved to 80° and so on, the pattern was kept in all 100 seconds.

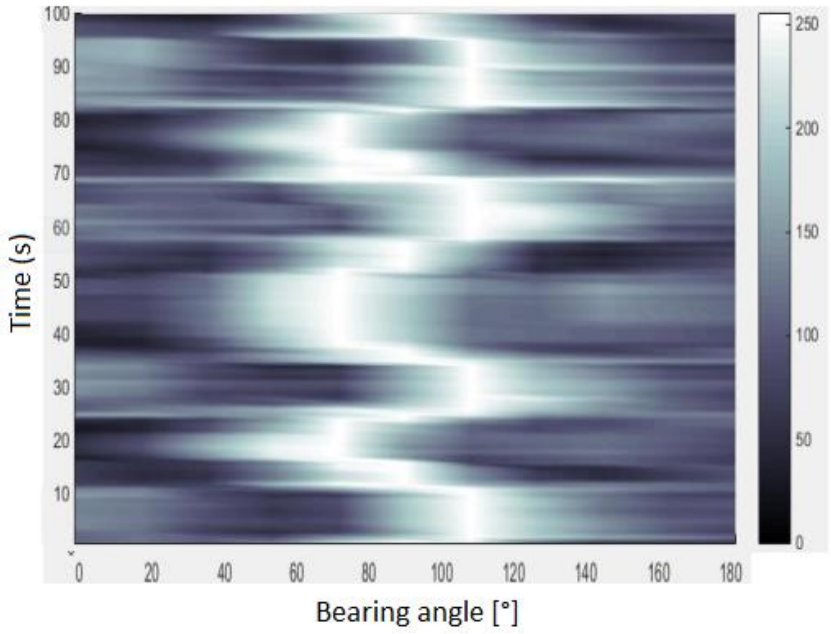


Figure 2.12: Bearing Time Record plot [8].

2.4.2 LOFAR (Low Frequency Analysis and Recording).

LOFAR, is a broadband analysis, generally estimates the noise vibration of the target machinery. The input signal of LOFAR comes from the beamforming system, therefore it usually has a SNR (Signal to Noise Ratio) higher than a single hydrophone, at least 10 lg N dB, where N is the channel number, and it is called to this process a conformed signal, in the figure 2.13 we can see the whole process to retrieve a LOFAR data. In the first block, the acoustic signal pass through an analogue to digital conversion as described in section 2.3.1, and the next block is related to a low pass digital filter as mentioned in section 2.3.2, and the last block is the Fast Fourier Transform algorithm in order to analyze the signals in the frequency domain.

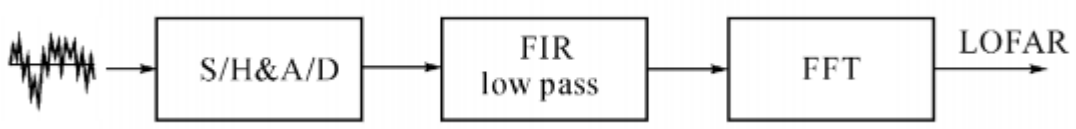


Figure 2.13: LOFAR algorithm.

The LOFAR plot basically shows the frequency components of the audio signal over time (acoustic signature), so every sample of the LOFAR is recorded and displayed in a

waterfall plot. In the figure 2.14 can be seen a LOFAR example, where it is possible to observe two main frequency components at 1800 and 2200 Hz over the 200 seconds than lasts the waterfall.

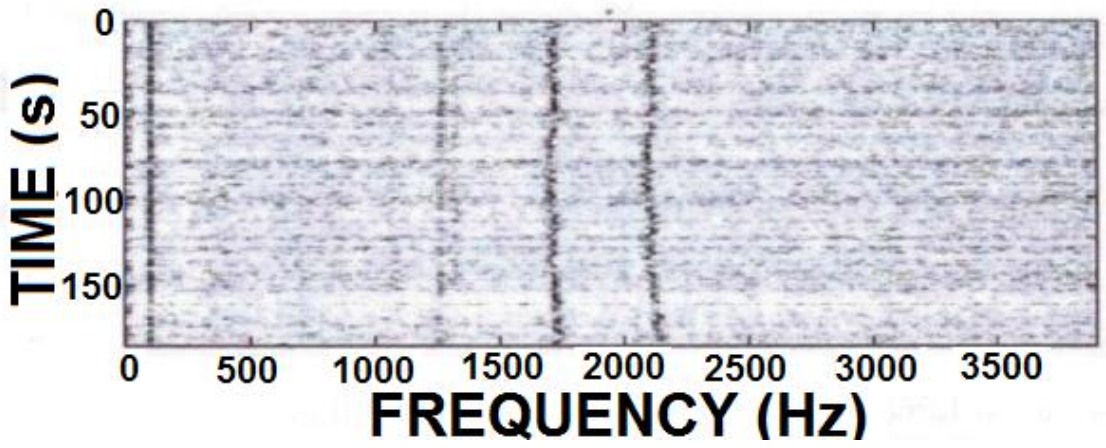


Figure 2.14: LOFAR display [18].

2.4.3 DEMON (Detection Envelope Modulation On Noise)

DEMON is a narrowband analysis that provides the characteristics of the propeller: number of shafts, shafts rotation frequency, and blade rate of the target. The input signal of DEMON comes from the beamforming system, in the figure 2.15 we can see the whole process to retrieve a DEMON data. In the first block the acoustic signal pass through an analogue to digital conversion as described in the 2.2.1 section, and the next block is related to a band pass digital filter implemented to limit the cavitation frequency range, and later an envelope detector is needed in order to perform the demodulation, and the last block is the Fast Fourier Transform algorithm in order to analyze the signals in the frequency domain.

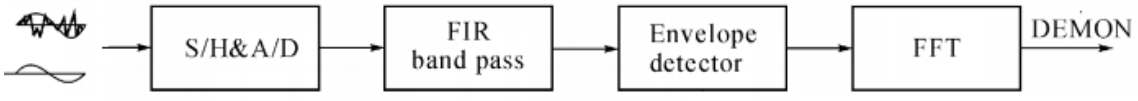


Figure 2.15: DEMON algorithm.

The DEMON plot basically shows the narrowband frequencies components of the audio signal over time, so every sample of the DEMON is recorded and displayed in a waterfall plot. In figure 2.16, it can be seen an example, where it is possible to observe five main frequency components at 5,10,18,23 and 28 Hz over the time than lasts the waterfall.

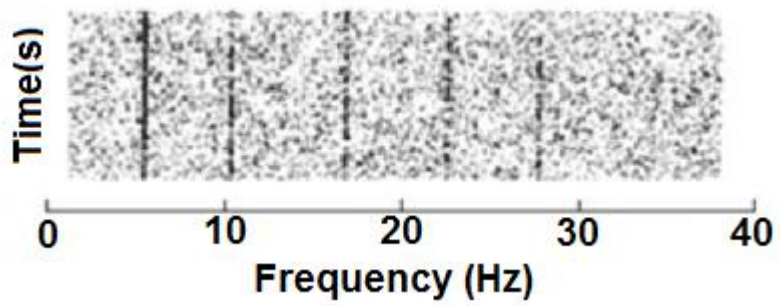


Figure 2.16: DEMON display [18].

2.4.4 PPI (Plan Position Indicator)

There are many different kind of sonars, one way to categorize them, is the mode of operation. The PPI plot is often used when the sonar is operated with the monostatic mode (active sonar), that mode of operation employs the emission and reception of acoustic signals in the same array of transducers. This configuration is designed to get information that are used to know the bearing (direction) and range (distance) of the contact in one single plot.

In order to get the data for build the PPI plot, it is necessary to send a burst of acoustic signal commonly known as “ping” to the acoustic transducers to the desired directions, later the beamformer processor start to wait for the echo of the transmitted signal, that echo is normally generated due the reflexion of the acoustic wave when hits the contact vessel.

In figure 2.17 can be seen a typical PPI plot, usually there are several concentric circles. Those circles are equidistant separated, and with the help of those circles it can be measured the distance range of the contact vessel. The equidistant separation represents a distance normally measured in meters, yards, etc. In image 2.17, it can be seen different kinds of spots or lines who are lighter than the background. Those lighter lines mean a higher level of reflected acoustic energy, and that higher levels means a higher probability of a detected contact vessel, and with the help of the PPI traces can be measured the bearing (direction of arrival of the acoustic echo) and the range (distance of the contact in function of the time that took the sound the round trip).

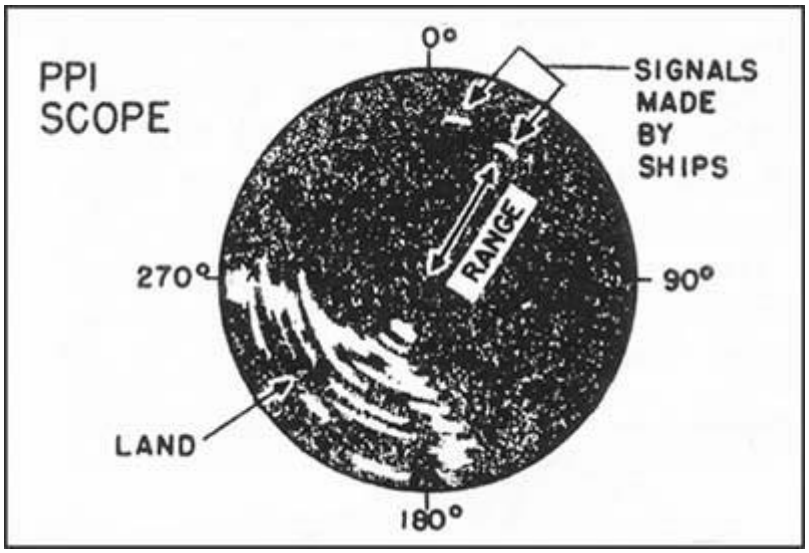


Figure 2.17: Typical PPI plot [19].

2.5 Typical Classification Algorithms.

Classification can be understood as a technique that predicts a discrete response. For example, whether an email is genuine or spam, or whether a tumour is small, medium, or large. Classification models are trained to classify data into categories. Typical applications include medical imaging, speech recognition, and credit scoring (Mathworks (2018), "Applying Supervised Learning").

Underwater contact vessel classification and recognition (similar to speech recognition) is a very attractive topic in sonar design due the different kind of applications that can be given. This task, so far is usually performed by a trained sonar operator and employ his auditory sense in order to hear and classify the contacts manually (on the base of his expertise). But with the boom of the machine learning techniques and the dawn of high-performance computers nowadays is a hot topic for research in order to automatize the whole process of classification by an electronic processor. At the present time, there are many kinds of theoretical results and technical achievements in speech recognition, fingerprint identification and human face recognition, and all those topics can be referenced to sonar signal processing applications. But an underwater acoustic signal has

their own specific characteristics in order to perform the classification, for example the noise of the propeller plays a very important role in radiated noise recognition.

The Naval Office of the USA began artificial neural network research earlier and some application software has already been introduced to the market [9-10].

There is a basic requirement for target classification in digital sonar, and that requirement is to retrieve the signal with a high confidence level. Therefore, it usually needs a high SNR (Signal to Noise Ratio) in the signals which will be classified. And the classifier algorithm has to use the original signal rather than the processed signal, this is because the original signal contains necessary information about the target, and the processed signal may lose some information which is important in the target classification (Qihu Li, 2011).

Choosing the right algorithm for classification purposes, can seem a no easy task, there are lots of supervised and unsupervised machine learning algorithms, and each takes a different approach to learning. But algorithm selection also depends on the size and type of data employed, the insights needed from the data, and how those insights will be used. In the following subthemes it's going to briefly describe some popular classification algorithms.

2.5.1 Logistic Regression

This algorithm fits a model that can predict the probability of a binary response belonging to one class or the other. Because of its simplicity, logistic regression is commonly used as a starting point for binary classification problems.

Commonly used:

- When data can be clearly separated by a single, linear boundary.
- As a baseline for evaluating more complex classification models.

In the below figure, shows an example of a logistics regression example and can be seen a threshold defined by a linear regression process, that threshold is known as a hypothesis.

Figure 2.18: Threshold to classify samples [20].

The hypothesis can be represented as an equation, and that equation is based on the sigmoid function also known as logistic function, and is shown in the equation 2.6.

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} \tag{2.6}$$

The $h_\theta(x)$ is the estimate probability for each 'x' input to get a y=1, where the 'y' value is binary, due the classification is binary. So, the hypothesis predicts y=1, when $\theta^T x \geq 0$ and for $\theta^T x \leq 0$ y=0. (Holehouse, 2018).

2.5.2 Neural Network

This algorithm was inspired by the function of the human brain. A neural network consists of highly connected networks of neurons that relate the inputs to the desired outputs. The network is trained by iteratively modifying the strengths or weights of the connections so that given inputs map to the correct response.

Commonly used:

- For modelling highly, nonlinear systems.
- When data is available incrementally and you wish to constantly update the model.
- When there could be unexpected changes in your input data.
- When model interpretability is not a key concern.

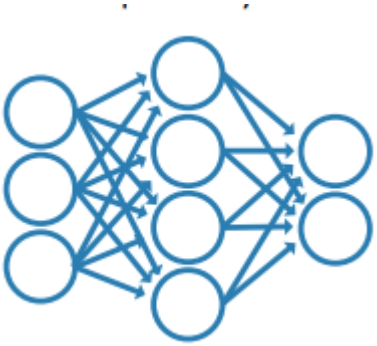


Figure 2.19: Illustration of a neural network [20].

The perceptron is the simplest neural network possible (a computational model of single neuron), and has one or more inputs, a processor and one single output. Every input is connected to the neuron, but before to reach the input, the input must be multiplied by a weight (W) that value is normally between -1 to 1. Typically, when a perceptron is created its weights are initialized to random values, and its inputs later are multiplied by the weights and finally all the resulted lines are added. The output of the perceptron is generated when the result of the added lines is passed through an activation function. That function either turn off or on the perceptron. In order to solve a problem with a possible ambiguity caused when all the inputs are zero, it is added a new input named "bias", this input usually have a value of one and also need a weight W to multiply the bias.

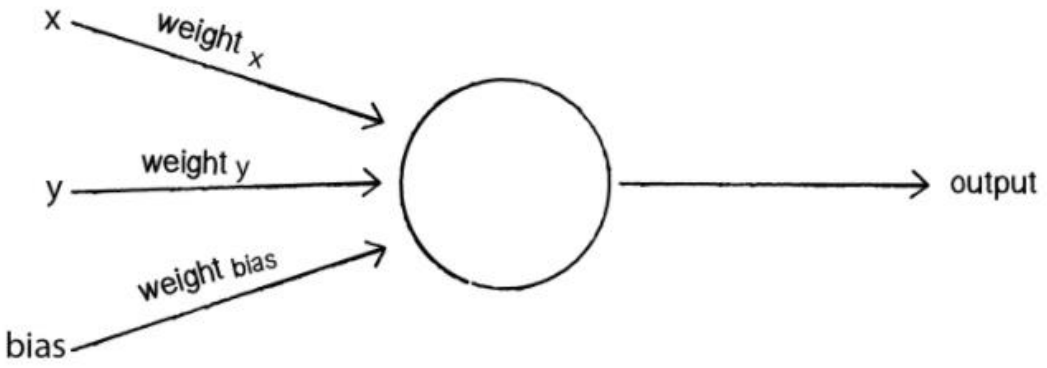


Figure 2.20: Representation of a perceptron.

Training a neural network is necessary in order to predict something accurately, one way to train the network is employ techniques of supervised learning. With this method, the inputs of the network have entries that corresponds to a known output. In this way the network can know if it has made a correct prediction. And if it is correct, the network can

learn of its error and adjust the weights W [12]. The training process can be summarized as follow:

- Provide the perceptron with entries for which there is a known output.
- Ask the perceptron to provide an answer with their actual values.
- Calculate the error. (The answer is typically a False or True).
- Adjust the weights in order to reduce the error.
- Start again the whole process.

2.5.3 Naïve Bayes

A naive bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. It classifies new data based on the highest probability of its belonging to a particular class.

Commonly used:

- For a small dataset containing many parameters.
- When you need a classifier that's easy to interpret.
- When the model will encounter scenarios that weren't in the training data, as is the case with many financial and medical applications.

Figure 2.21: Naïve Bayes Classifier [20].

2.5.4 Decision Tree

A decision tree predicts responses to data by following the decisions in the tree from the root (beginning) down to a leaf node. A tree consists of branching conditions

where the value of a predictor is compared to a trained weight. The number of branches and the values of weights are determined in the training process. Additional modification, or pruning, may be used to simplify the model.

Commonly used:

- When it is needed an algorithm that is easy to interpret and fast to fit.
- To minimize memory usage.
- When high predictive accuracy is not a requirement.

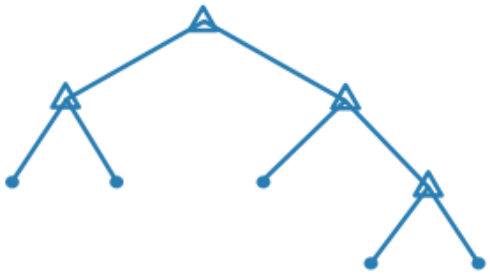


Figure 2.22: Decision Tree Classifier [20].

2.5.5 K Nearest Neighbors (kNN)

kNN categorizes objects based on the classes of their nearest neighbors in the dataset. kNN predictions assume that objects near each other are similar. Distance metrics, such as Euclidean, city block, cosine and Chebychev, are used to find the nearest neighbor.

Commonly used:

- When you need a simple algorithm to establish benchmark learning rules.
- When memory usage of the trained model is a lesser concern.
- When prediction speed of the trained model is a lesser concern.

Figure 2.23: kNN Nearest Neighbors Classifier [20].

2.5.6 Support Vector Machine (SVM)

Classifies data by finding the linear decision boundary (hyperplane) that separates all data points of one class from those of the other class. The best hyperplane for an SVM is the one with the largest margin between the two classes, when the data is linearly separable. If the data is not linearly separable, a loss function is used to penalize points on the wrong side of the hyperplane. SVMs sometimes use a kernel transform to transform nonlinearly separable data into higher dimensions where a linear decision boundary can be found.

Commonly used:

- For data that has exactly two classes (it can be used also for multiclass classification with a technique called error correcting output codes).
- For high-dimensional, nonlinearly separable data.
- When you need a classifier that's simple, easy to interpret, and accurate.

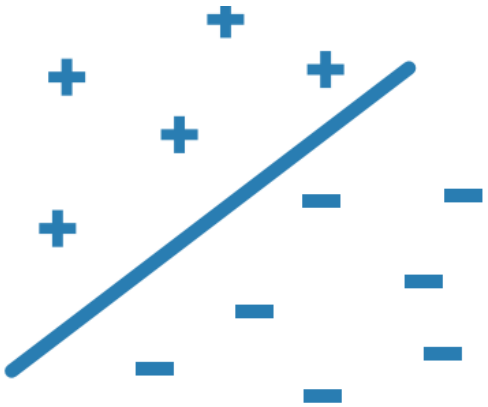


Figure 2.24: Support Vector Machine Classifier [20].

3 EXPERIMENTAL PROTOCOL.

This thesis investigates the problem of using a neural network classifier method for performing multi-class estimation. This approach was chosen on the base of the great results the artificial networks are gaining for problems related to pattern recognition and classification at the present time, and the global trend in machine learning algorithms implementations using ANNs. The methods developed are applied to the problem of acoustic signal processing for speech recognition. So, in order to perform that research, and verify functionalities, it is necessary an acquisition system for the acoustic signals of the marine vessels that will be under the classification process. An overview of the acquisition system is shown in the figure 3.1.

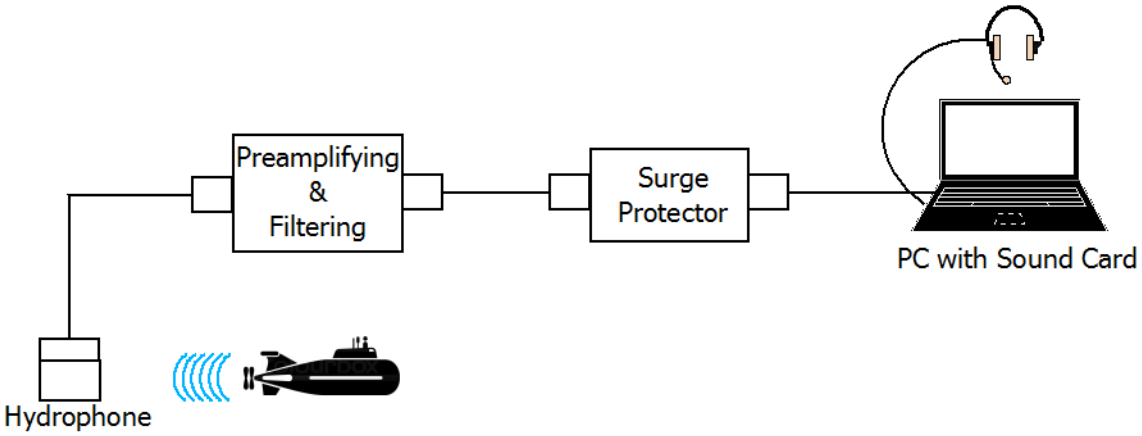


Figure 3.1: Acquisition system for the acoustic signals.

Another essential tool will be the software employed to retrieve the sound, to generate the data base, to train the neural network, and to run the final classification implementation. In the first part of the figure 3.2 is shown a UML (Unified Modeling Language) schema describing the communication between an actor (software operator) and the software components (divided by four classes), it was also added a block diagram in order to represent those classes and their interaction in a simpler way.

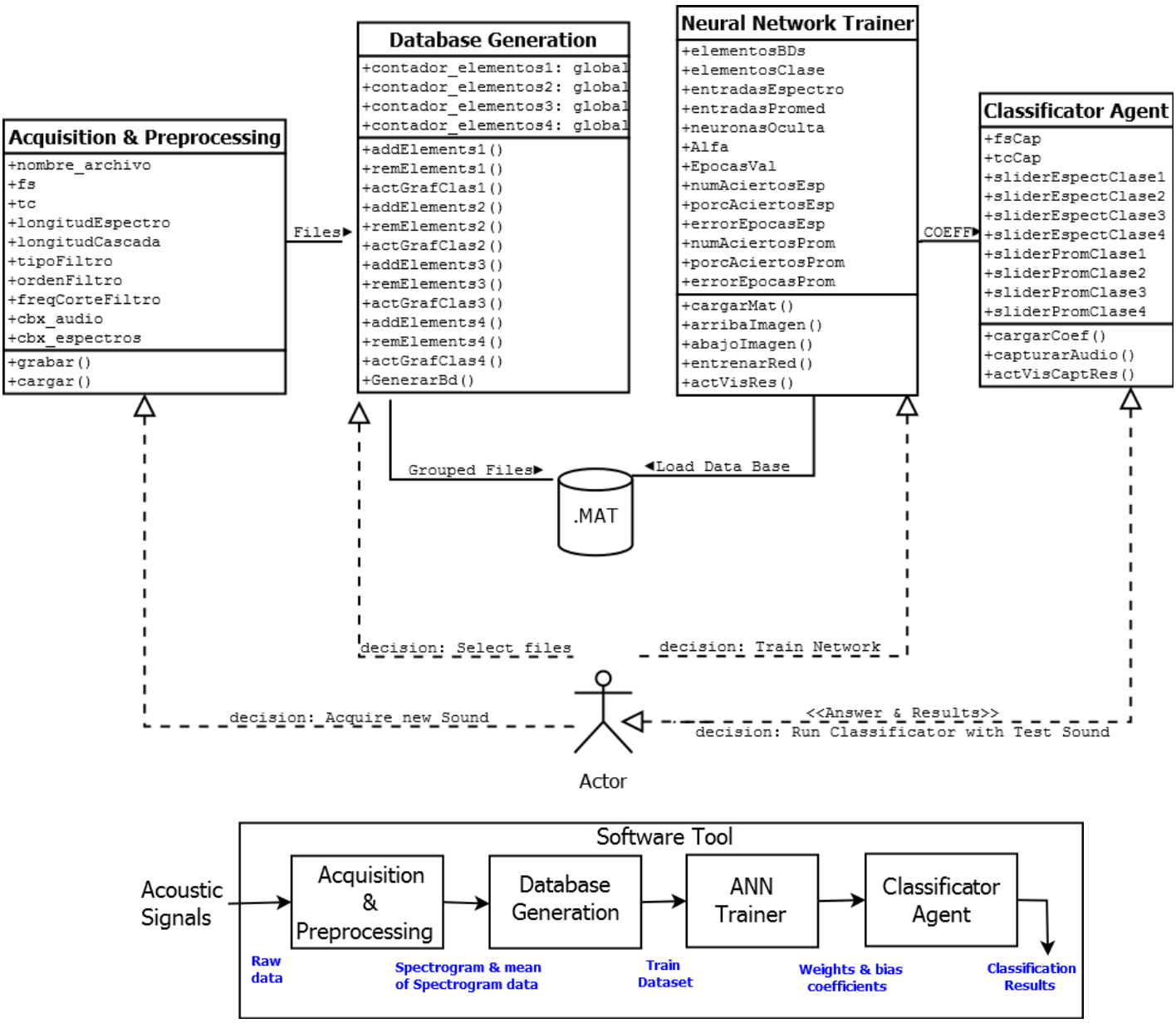


Figure 3.2: Schemas describing the Software Tool.

3.1 Required Hardware

3.1.1 Hydrophone

In order to detect the acoustic signals, it is necessary a sensor. For this thesis project, it is proposed a cylindrical hydrophone, with the following features that are explained in the below lines.

In the Figure 3.3 it is shown a hydrophone designed and fabricated at the INIDETAM (Instituto de Investigación y Desarrollo Tecnológico de la Armada de México), in the laboratory of underwater acoustics, using piezoelectric ceramics as sensing elements. Its construction is such that it is absolutely waterproof. This hydrophone due its cylindrical design has a 360° sensitivity response. For this project purpose, this is a desired feature because it is possible to acquire any acoustic signal in all the surrounding environment without the need to give any special direction (either mechanically or with a beamforming technique) to the sensor in order to find the acoustic source or target.

Figure 3.3:Proposed cylindrical hydrophone.

This cylindrical hydrophone was individually calibrated until to get the final frequency response showed in the figure 3.4, where can be perceived the flat response in a wide frequency range, which includes the human audible range that goes approximately from 20 to 20kHz, and those are the experimental proposed frequencies for this work. The measurements were done in a hydroacoustic water tank in free-field conditions, using a calibration system set-up.

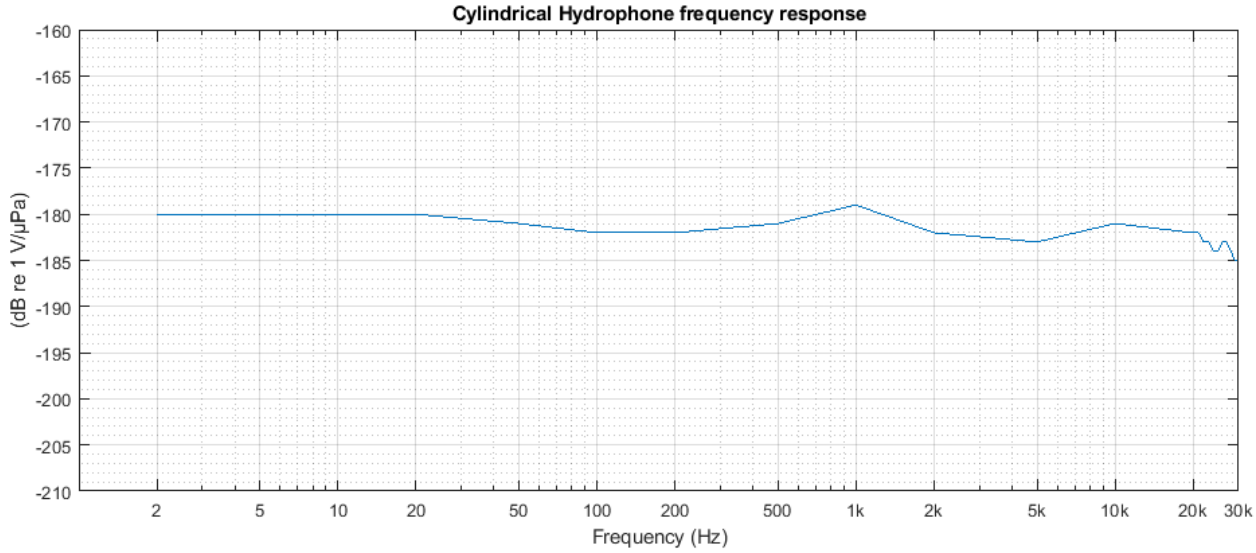


Figure 3.4: Hydrophone frequency response.

A flat frequency response for a wide range of frequency is a desired feature for the hydrophone, since this guarantees that the acoustic measurements will not contains some bias signals in some frequencies.

3.1.2 Preamplifier

For this work, it has been used the Stanford Research Low Noise Preamplifier, Model SR560, which has a 4nV/√Hz input noise, a variable linear gain from 1 to 50,000, two configurable signal filters, 1 MHz bandwidth, input impedance of 100 MΩ, two insulated output BNC connectors that provide 600 Ω and 50 Ω outputs.

This preamplifier is very convenient due to the flexibility that grants to modify the amplification gain and the filters cutoffs to eliminate any noise that is not part of the desired signal. The SR560 contains two first-order RC filters whose cutoff frequency and type High Pass Filter (HPF) or Low Pass Filter (LPF) can be configured from the front panel. Together, the filters can be configured as a 6 or 12 dB/oct rolloff low-pass or high-pass filter, or as a 6 dB/oct rolloff band-pass filter.

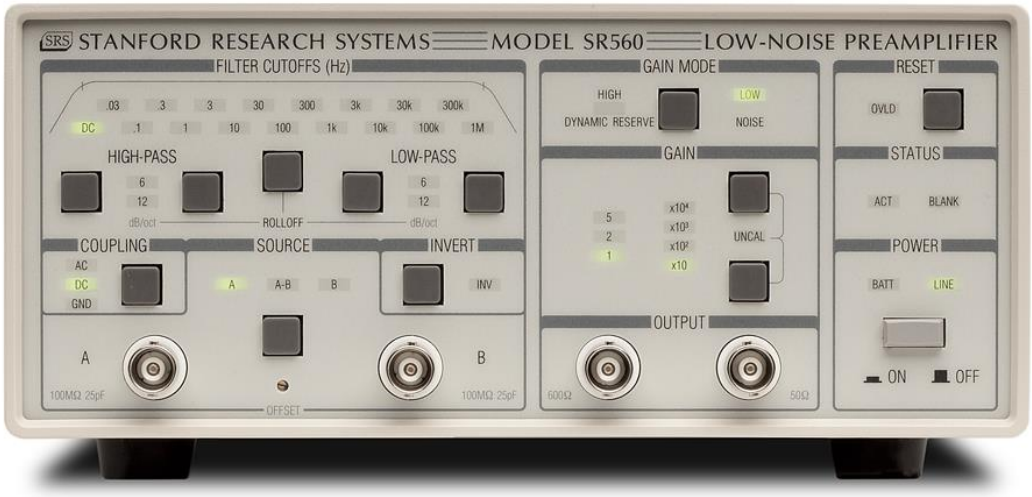


Figure 3.5: Low Noise Preamplifier Stanford Research.

In order to use the preamplifier with the hydrophone, it is necessary to do some configuration to the equipment, the proposed basic configuration is shown in the table 3.1.

FUNCTION	SELECTION
COUPLING	DC
LOW-PASS	10K
SOURCE	Channel A
GAIN MODE	Low Noise
GAIN	$5x10^2$
POWER	BATT
OUTPUT	50Ω
HIGH-PASS	100

Table 3.1 Recommended configuration parameters.

For the coupling function, it is suggested DC instead of AC because it is desired to process the whole signal, and AC coupling just shows the AC component. For the filters, for HIGH-PASS it is recommended to set it to 100 Hz in order to attenuate the 60 Hz electrical component typically have, and reduce the electrical noise. For LOW-PASS it is recommended to set it to 10kHz due it is the desired bandwidth for all the measurements and tests. For the source, either A or B channels can be used indistinctly, just for matter of order the channel A is selected. As we saw in the section 2.1.3 the first amplifier stage must be a Low Noise preamplifier in order to do not corrupt the signal with electronic noise or other type of noise, so in the GAIN MODE is selected Low Noise. For the GAIN function it is recommended $5x10^2$ as can be seen in the table 3.1, but this value depends

on the environment where the acoustics samples are recorded, for example the recommended value was used in a hydroacoustic tank with a small remote operated vehicle. For the POWER, it is recommended operate the preamplifier on batteries in order to reduce the electric noise that can generate the electric lines that are necessary to operate plugged to an AC outlet. For the OUTPUT 50Ω is suggested, in order to have a low impedance output, and thus have a more robust output signal.

3.1.3 Surge Protector

The output signal from the preamplifier Stanford research, can reach ±26 Volts in saturation, that scenario usually happens in cases where the hydrophone is physically touched or beaten, that voltage is too high for any kind of audio card in any computer, so if that voltage is applied to a computer audio card, it will damage the card. So, in order to protect the computer of overvoltage, it is necessary to create an electronic circuit that can be able to cut any signal above any desired voltage.

There are electronic circuits that are designed in order to limit the voltage, for example the diode clipper circuits they limit or clip-off the positive and/or the negative part of an input AC signal [13]. As zener clipper circuits limit or cut-off part of the waveform across them, they are mainly used for circuit protection. In the figure 3.6 we can see a clipping circuit using a two zener diodes, the clipping voltage depends on the voltage of the zener diode, if the output waveform exceeds the voltage of the zener diode, the zener diode will “clip-off” the excess voltage from the input producing a waveform with a flat top still keeping the output constant at the voltage of the Zener.

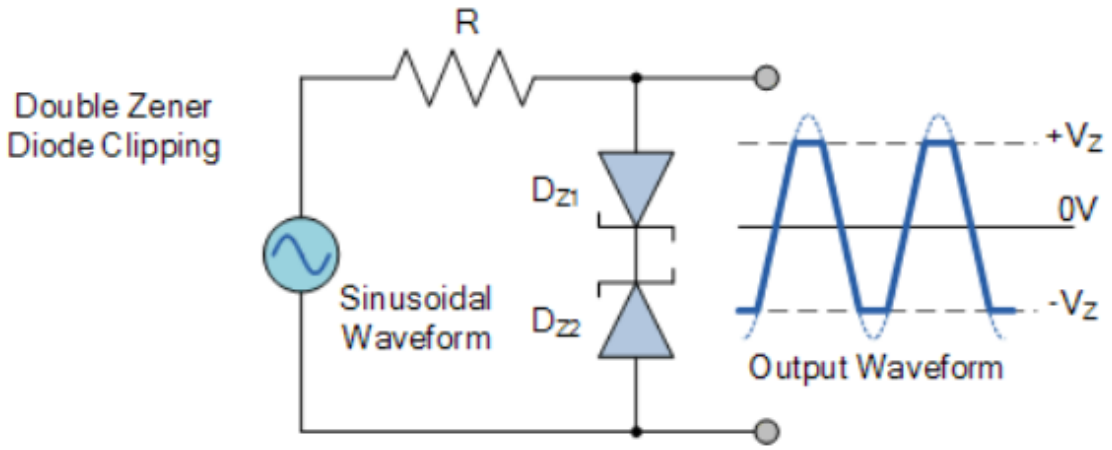


Figure 3.6: Zener diode Clipping Circuit.

For this work, the sound card that have been used is "Creative Malcolm chip+Recon3Di", but there is no clear information about the maximum voltage levels that can receive the card, so it was necessary to do a measurement of the common voltages that receives the sound card, for that purpose it was used a headphone jack connected to a smartphone, and it was generated a continuous sine wave signal, with frequency of 500Hz with maximum amplitude, all this was set with the help of a function generator app by Keuwlsoft. The concluding measurements were ±1.65v (3.3 volts peak to peak), so the dynamic range of and smartphone in its output headphone jack is ±1.65v, and that voltage by standard can be safely be applied to any audio card in a computer.

The resistor connected in series with the zener diodes, limits the current flow through the diodes, in order to avoid that the zeners get damaged for over current. For this application the selected diodes maximum power are ½ watts and the voltages of the diodes are 1.8v (a little bit higher than 1.65v measured previously), then the maximum current (MaxCurr) than a zener diode can receive is 280mA as can be seen in equation 3.1, and for this circuit implementation it is proposed a 470Ω resistor, and its value is five times higher than the minimum necessary 86.4Ω calculated in the 3.2 equation, taking in account the maximum voltage input value is Vs = 26v, for the case of overvoltage in the preamplifier equipment.

$$\text{MaxCurr} = \frac{Watts}{\text{Voltage}} = \frac{\frac{1}{2}W}{1.8\text{V}} = 280\ mA \qquad (3.1)$$

$$\text{ResSerie} = \frac{Vs - VoltageDiode}{\text{MacCurr}} = \frac{26-1.8}{280\text{mA}} = 86.42\Omega \qquad (3.2)$$

In the figure 3.7 it's shown the clipping electronic circuit with the proposed values of the diodes and the resistor according to the previous calculations.

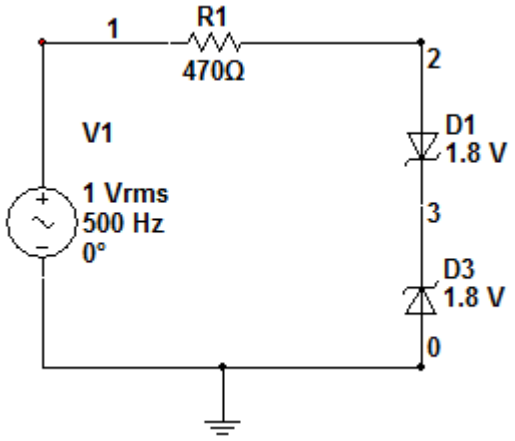


Figure 3.7: Surge protector with values.

The circuit of the figure 3.7, can be simulated it on software and validate its behavior, with the help of Multisim that is a SPICE (Simulation Program with Integrated Circuits Emphasis) simulation and circuit design software for analog and digital electronics in education and research. In the figure 3.8 in the first image can be seen two sinusoidal signals, the blue one is the input signal (1 Vrms or 2.83 Vpp) and the red one is the output signal, both signals have the same scale 2v/Div in the oscilloscope, and both signals have the same amplitude, as can be observed from the figure, there is not a cut of the output signal, both have the same shape and the same amplitude, that is due the peak to peak voltage doesn't surpass the 3.6 Vpp (1.8v x 2) of the design, so there is not cut of the output signal. But for the second image of figure 3.7 the input signal represented as the blue sinusoid now is 10 Vrms or 28.3 Vpp with a scale of 20v/Div in the oscilloscope, the red sinusoid is the output and have a 5v/Div scale, and can be observed that both signals don't have the same amplitude, the output have been clipped to almost the desired 3.6Vpp. So, with this simple electronic design can be achieved the desired behavior of the voltage, in order to protect the computer sound card due over voltages.

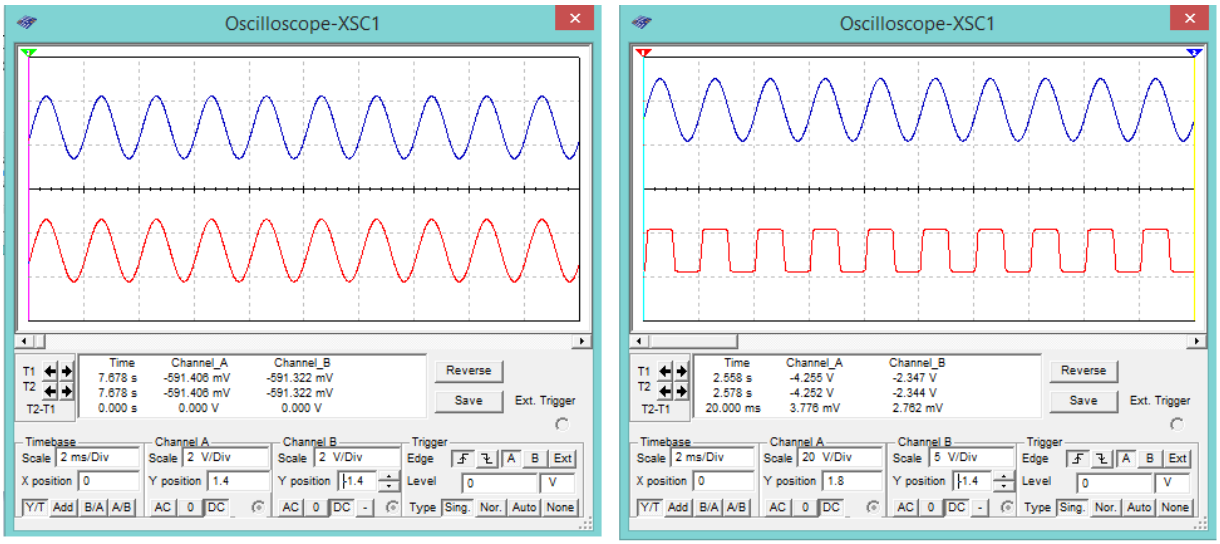


Figure 3.8: Simulation of Surge Protector Circuit.

In the figure 3.9 in the first image it is shown the construction of the electronic circuit with the resistor and the two zener diodes with the cables, in one end can be found the headphone jack for plugging it in the computer sound card and in the other end the BNC connector for plugging it in the output of the preamplifier. The second image of the figure 3.9 shows the final product inside of an encapsulating material in order to improve the electronic board of possible electrical interferences and mechanical movements of the cables that may reduce the performance of the board and be more vulnerable to electromagnetic noises.

Figure 3.9: Surge Protector equipment.

3.1.4 Computer Specifications

For the implementation of this thesis work it was necessary to use a computer equipment for both to acquire the acoustic signals with the sound card that carries an embedded analog to digital converter and for run the algorithms for preprocessing the data and the classificator agent that was trained previously. The hardware specifications of the employed computer are shown in the table 3.2.

Alienware 17 R2	
Brand	Alienware
Processor	Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz
Cores	8
Graphic Card	NVIDIA GeForce GTX 970M
Clock Speed	2.5 GHz
Hard Disk	1 TB
Memory Ram	16 GB
Operating System	Windows 8.1,64 bits
Sound Card	Creative Malcolm chip+Recon3Di
Model	17 R2

Table 3.2 Technical Specifications of the employed computer.

3.2 Required Software

3.2.1 Software Tab for acquisition and processing of signals.

For the implementation of the software part, it was used MATLAB R2018a. MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran and Python. And for this thesis project are used MAT extension files as a way to store information in the local hard drive, those files are in a binary data container format that MATLAB program uses. The extension was developed by

Mathworks and MAT files are categorized as data files that may include variables, functions, arrays and other information.

In the figure 3.2 was presented the complete schema of the software tool implemented, and now in this chapter section it will be described the "Acquisition & Preprocessing" software section. The main purpose of this piece of code is to provide the means to acquire the needed audio signals by two ways. The first one is using the computer sound card when introducing an audio plug to the microphone entry port of the computer, and the other way is to provide a WAV (Waveform Audio File Format) file to the system. And finally apply if desired a digital filtering to the acquired audio signal and to set up the parameters for the creation of the spectrogram image. In the figure 3.10 can be seen one UML block for the Acquisition & Preprocessing class with their respective attributes and methods. All those attributes are related with parameters for configuration of the captured audio file and the visualization of the signals. And the two functions are on charge to activate the process for recording the audio(grabar) and for load a WAV audio file from the computer with the attributes previously selected.

Figure 3.10: UML block for the Acquisition & Preprocessing Tab.

In the figure 3.11, it is described an important process of this software section that is the conversion of an acoustic signal in the time domain to a spectrogram. In order to create the spectrogram, first the acquired acoustic signal must be divided in sections according to the length of the spectrogram waterfall, later to each time slot section it is applied the fast fourier transform (FFT) algorithm, in order to pass the signal to the frequency domain, to finally stack all the resulted spectrum in a single matrix, which is called spectrogram.

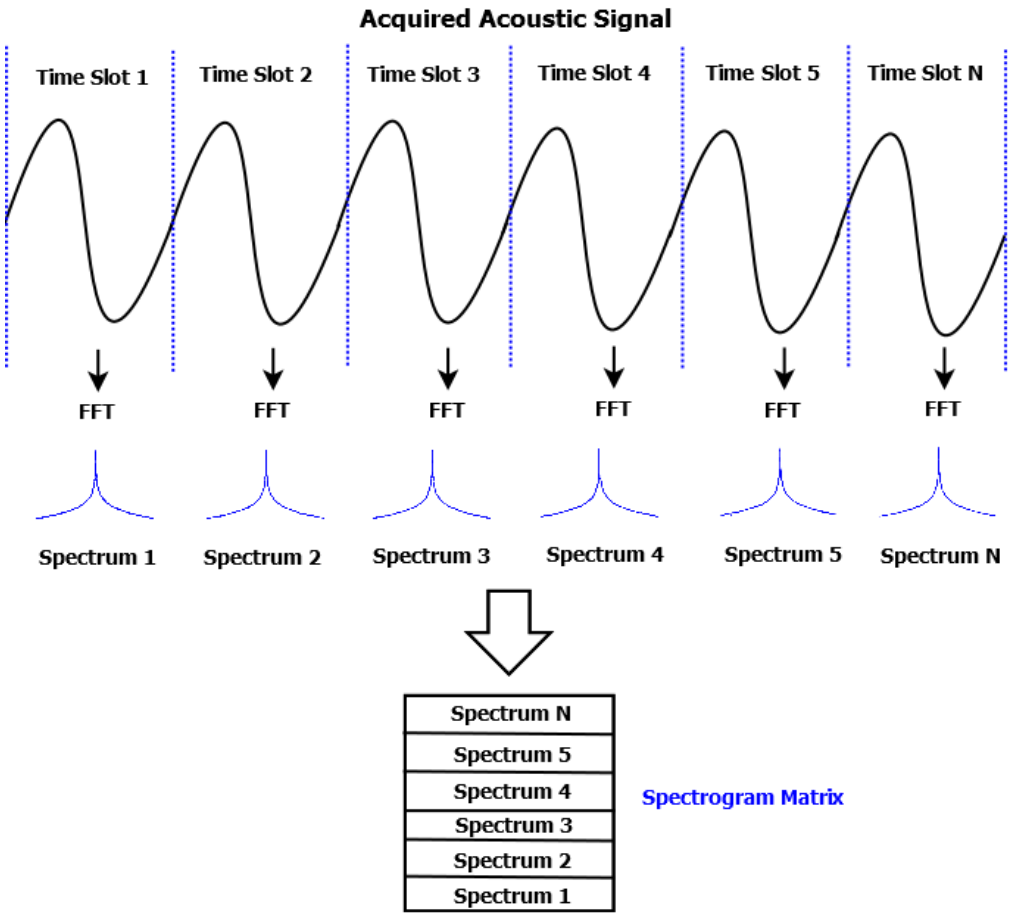


Figure 3.11: Spectrogram Matrix Creation.

Now in the figure 3.12, it is described the process of convert a spectrogram matrix to a spectrogram mean vector. Basically, for every column of the spectrogram, the numeric values were added and finally divided by the number of elements, and those final numbers are the elements of the spectrogram mean that is a unidimensional array.

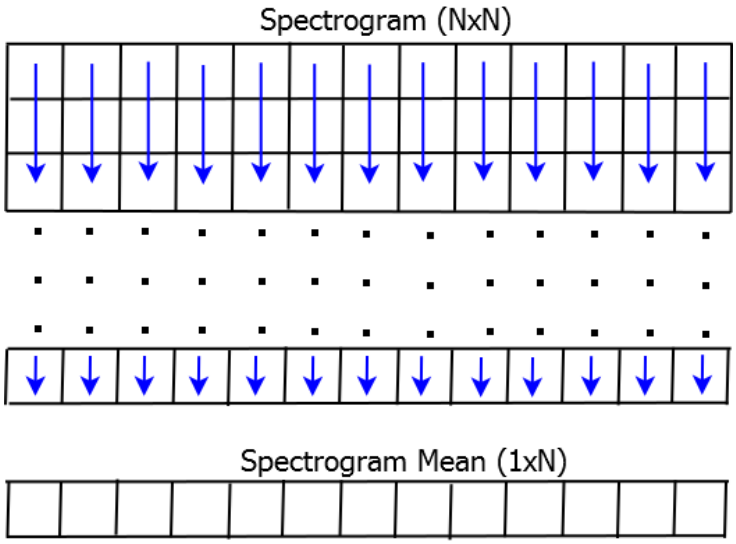


Figure 3.12: Spectrogram to spectrogram mean.

In the figure 3.13, can be seen a full view of the Acquisition & Preprocessing tool in the first Tab of the software GUI (Graphical User Interface). For demonstration purposes was loaded an audio in the tool, in order to see typical audio signals plots, the first plot (the upper one) represents the audio signal acquired in the time domain, the second plot (the middle one) displays the spectrogram of the previous acquired audio signal, and the last plot displays the spectrogram mean. The second and third plots are the ones that will be used to train the artificial neural network proposed for this thesis work.

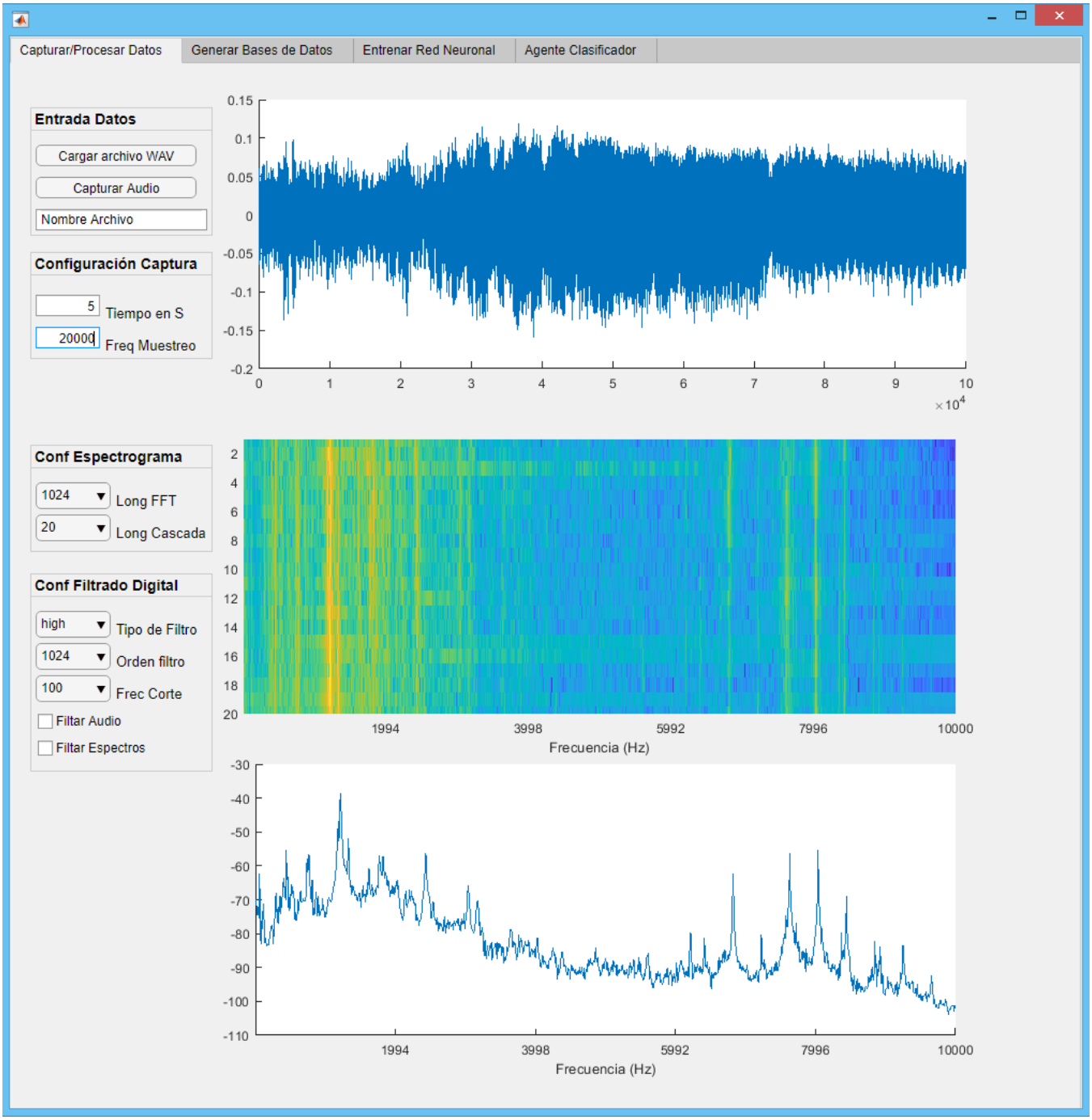


Figure 3.13: Acquisition & Preprocessing Tab.

This software section was designed having in mind to get the capability to acquire audio signals and preprocess them if it is necessary, for example, like filter any desired frequency band, and the other important job is to visualize the different kind of plots, in order to know better the kind of involved signals for this project. So, in order to achieve

the previous capabilities was necessary to implement the software components described in the table 3.3 for the Acquisition & Preprocessing class.

"Acquisition & Preprocessing" Components	
Cargar archivo WAV	This button executes the method "cargar()", and open a dialog box to load a WAV file, and when finish the capture time, immediately start to plotting the signals in the three graph areas and finally saves all the retrieved data in variables (inside a MAT file) for future use. The graphs plotted and the saved data depends on the parameters of configuration provided before.
Capturar Audio	This button executes the method "grabar()", and start to acquiring audio signals , and when finish the capture time, immediately start to plotting the signals in the three graph areas and finally saves all the retrieved data in variables (inside a MAT file) and a audio WAV file for future use. The graphs plotted and the saved data depends on the parameters of configuration provided before.
nombre_archivo	This edit text field is intended to use it in order to save the previously acquired audio (with "Capturar Audio" button) in a WAV file with a custom name.
Tiempo en S	This edit text field is intended to use it in order to set time of acquisition of the sampled signal in seconds.
Freq Muestreo	This edit text field is intended to use it in order to set the sample rate of the acquired signal in hertz.
First Plot area	This plot area (the upper one) is intended to use it in order to display the acquired audio signal in the time domain. In the X axis can be seen the total number of samples, and in the Y axis the magnitude of the acquired signal, the maximun and minimum range is 1 to -1 units respectively.
Long FFT	The use of this Drop-Down component is to select the length of the final resultant array after apply the Fast Fourier Transform algorithm to the audio signal.
Long Cascada	The use of this Drop-Down component is to select the number of lines the waterfall plot.
Tipo de Filtro	The use of this Drop-Down component is to select the type of filter (High or Low pass) applied to the acquired audio signal, for the case that feature is enabled.
Orden Filtro	The use of this Drop-Down component is to select the order of the filter previously selected.
Frec Corte	The use of this Drop-Down component is to select the cut off frequency of the digital filter.
Filtrar Audio	The use of this Check Box component is to enable the feature to save the acquired audio with the filtered values and not with the real values.
Filtrar Espectros	The use of this Check Box component is to enable the digital filter feature to the Power spectral & Averaged plots.
Second Plot area	This plot area (the middle one) is intended to use it in order to display the Power Spectral Density in a waterfall form or Spectrogram. In the X axis can be seen frequency values, and in the Y axis the number of samples of the waterfall (represents the time), and the Z axis represents with colors in the image the magnitude of the frequency spectrum of the acoustic signal acquired in dBs (applying $20*\log_{10}$). Yellow means higher magnitude value (warm color), blue means lower value (cold color).
Third Plot area	This plot area (the lower one) is intended to use it in order to display the Averaged Power Spectral Density of the spectrogram plotted in the second plot area (this plot contains the mean of each column of the spectrogram). In the X axis can be seen frequency values in hertz, and in the Y axis the magnitude in dBs of the frequency spectrum for the acoustic signal acquired.

Table 3.3 Acquisition & Preprocessing components.

3.2.2 Software Tab to create the training database.

In this chapter section, it will be described the tool to generate a data base file with all the needed elements in order to train the artificial neural network (ANN). Therefore, the first Tab is intended to retrieve the fundamental information and save that particular information in singles data files, but in order to train an ANN is necessary to create a tool, to group all those single files in a bigger file that contains all the needed information to train both neural networks (the neural network for the spectrogram image and for the meanof that spectrogram). And that final file which is called training database must have a specific format, in the figure 3.14 can be seen the format, where every column of the big final matrix represents one image of the spectrogram converted previously in a column unidimensional array, the final length of that column array is the product of the selected length of the FFT ("Long FFT" GUI component) with the length of the waterfall ("Long Cascada" component). Finally, every single image for the first class must be concatenate it in a horizontal way, the same process is for the remaining three. A similar process was implemented to create the training database for the mean of the spectrogram, but this case was straight forward due the input vector is unidimensional, so, it was not necessary to convert a matrix to a unidimensional vector like in the first case.

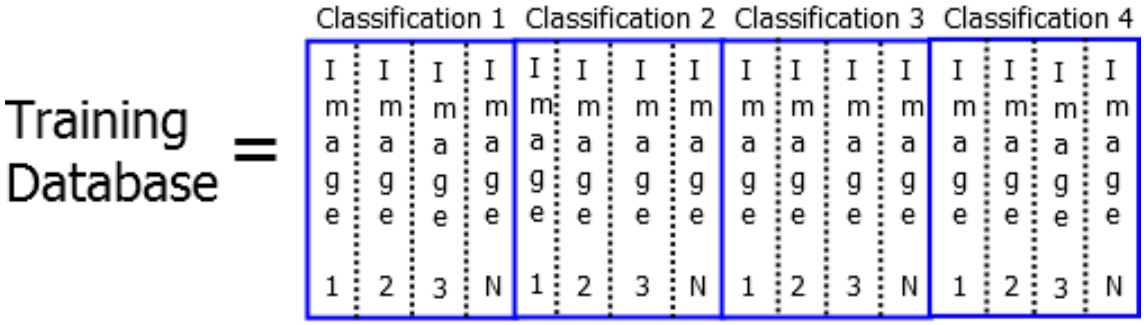


Figure 3.14: Training database matrix format.

So, this software tab at the end of the process, generates a MAT file named "imagenesEntrenamiento.MAT" with two training databases one named "Espectrograma" and the other one "Promediado" and one variable named "dimEspec" that contains the dimensions of one single spectrogram image, this last information is important for the moment when one spectrogram image needs to be rebuilt.

In the figure 3.15, it can be seen the UML block for the create database GUI tab with their respective attributes and methods. There are four attributes for this software section and all of them are related with counters of the number of training elements selected, and there are 13 methods, the first twelve are related with basic operations of add, remove and plot audio elements for each class, and the last method "GenerarBd" is on charge to wrap all the selected elements and create one single MAT file "imagenesEntrenamiento.mat" with all the necessary data to train the ANN.

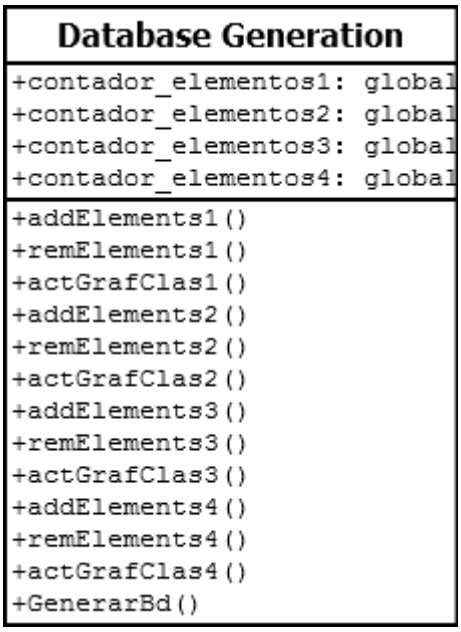


Figure 3.15: UML block for the database creation Tab.

In the figure 3.16, it can be seen a full view of the database creation GUI tab. For demonstration purposes was loaded two audio elements in the tool per class, in order to see typical audio signals plots. That GUI tab can be divided in four areas (Clase 1, Clase 2, Clase 3, Clase 4), for the Clase 1 (the upper one component), there are two buttons, one is for remove an element and the other one is for add a new element, there is also a label that have a counter with the number of elements added. There is also a plot area where is possible to visualize the image of the element selected in the ListBox component and it is possible also to switch the plot between the spectrogram and the spectrogram mean using a component switch in the GUI. The same components were designed for the remaining three areas. And finally, there is a button component that activates the GenerarBd method, and generates the MAT file.

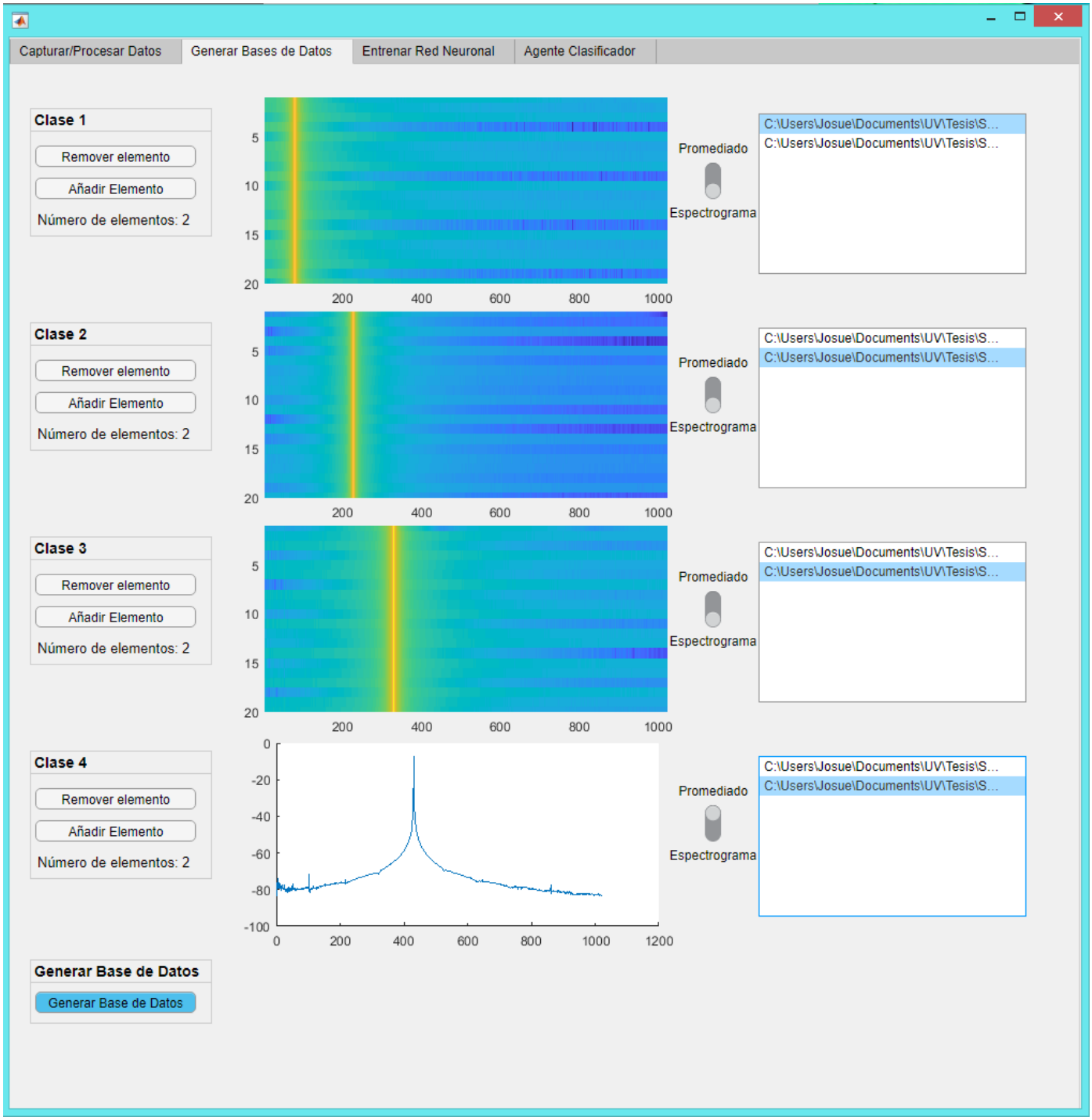


Figure 3.16: Database creation Tab.

This software section was designed having in mind to get the capability to easily create a single MAT file that contains all the information of the four different training classes with

the proper format to be use it for the ANN in the training process. In the table 3.4 are described all the GUI components used.

"Create database" Components		
C	Remover Elemento	This button executes the method "remElements1()", and remove the last element available in the ListBox component.
L	Añadir Elemento	This button executes the method "addElements1()", and open a dialog box to load a MAT file with the data of the training images previously generated to its respective ListBox.
A	Número de elementos: #	This label indicates the total number of added elements in the ListBox.
S	Plot Area	This plot area is intended to use it in order to display the Spectrogram or the Averaged Power Spectral Density of the spectrogram.
E	Promediado/Espectrograma	This switch component is used to select what kind of graphic is chosen to display in the plot area, can be the spectrogram or the average of the spectrogram.
1	ListBox	Every time a new training element is added, immediately this ListBox is updated with the location and name of the MAT file, and this listBox also executes the method actGrafClas1() allowing to the user select the item to be displayed in the plot area.
C	Remover Elemento	This button executes the method "remElements2()", and remove the last element available in the ListBox component.
L	Añadir Elemento	This button executes the method "addElements2()", and open a dialog box to load a MAT file with the data of the training images previously generated to its respective ListBox.
A	Número de elementos: #	This label indicates the total number of added elements in the ListBox.
S	Plot Area	This plot area is intended to use it in order to display the Spectrogram or the Averaged Power Spectral Density of the spectrogram.
E	Promediado/Espectrograma	This switch component is used to select what kind of graphic is chosen to display in the plot area, can be the spectrogram or the average of the spectrogram.
2	ListBox	Every time a new training element is added, immediately this ListBox is updated with the location and name of the MAT file, and this listBox also executes the method actGrafClas2() allowing to the user select the item to be displayed in the plot area.
C	Remover Elemento	This button executes the method "remElements3()", and remove the last element available in the ListBox component.
L	Añadir Elemento	This button executes the method "addElements3()", and open a dialog box to load a MAT file with the data of the training images previously generated to its respective ListBox.
A	Número de elementos: #	This label indicates the total number of added elements in the ListBox.
S	Plot Area	This plot area is intended to use it in order to display the Spectrogram or the Averaged Power Spectral Density of the spectrogram.
E	Promediado/Espectrograma	This switch component is used to select what kind of graphic is chosen to display in the plot area, can be the spectrogram or the average of the spectrogram.
3	ListBox	Every time a new training element is added, immediately this ListBox is updated with the location and name of the MAT file, and this listBox also executes the method actGrafClas3() allowing to the user select the item to be displayed in the plot area.
C	Remover Elemento	This button executes the method "remElements4()", and remove the last element available in the ListBox component.
L	Añadir Elemento	This button executes the method "addElements4()", and open a dialog box to load a MAT file with the data of the training images previously generated to its respective ListBox.
A	Número de elementos: #	This label indicates the total number of added elements in the ListBox.
S	Plot Area	This plot area is intended to use it in order to display the Spectrogram or the Averaged Power Spectral Density of the spectrogram.
E	Promediado/Espectrograma	This switch component is used to select what kind of graphic is chosen to display in the plot area, can be the spectrogram or the average of the spectrogram.
4	ListBox	Every time a new training element is added, immediately this ListBox is updated with the location and name of the MAT file, and this listBox also executes the method actGrafClas4() allowing to the user select the item to be displayed in the plot area.
Generar Base de Datos		This button executes the method "GenerarBd()", and start the process of loading all the MAT files previously selected wich contains the training images in order to finally create the training database as described in the figure 3.11.

Table 3.4 Specifications of the Create database GUI components.

3.2.3 Software Tab to train the Artificial Neural Network.

In this section it will be described the tool to train the artificial neural network (ANN). This GUI tab is the place where the two neural networks (spectrogram and spectrogram mean) topologies are defined, and the topology is in function of the number of inputs, the chosen neurons in the hidden layer and the final four neurons which belongs to the four final classes. In this section the main algorithm of this working thesis was implemented, and is the backpropagation algorithm using the method of gradient descent in order to train the multilayer networks. Also, in order to insert the input data for the training process was necessary to apply a unity-based normalization equation to the input data, to set all data in between the range of [0,1], this process is also called feature scalling and is expressed in the following equation.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.3}$$

Where X is the training unidimensional input array and X' is the normalized output array. In the figure 3.17 it's shown a UML block with the minimum necessary attributes and methods, the first five attributes (from top to bottom) are related with parameters of the topology of the ANN, the next two are parameters related with the training of the ANN, and the last six attributes are results of training both neural networks. There are also five methods, the first one is for load the training database that was created previously, the following two methods and the last one is for select the visualization of the images of the database, and the remaining method is on charge to execute the process of the training with all the previously configurated attributes and training data.

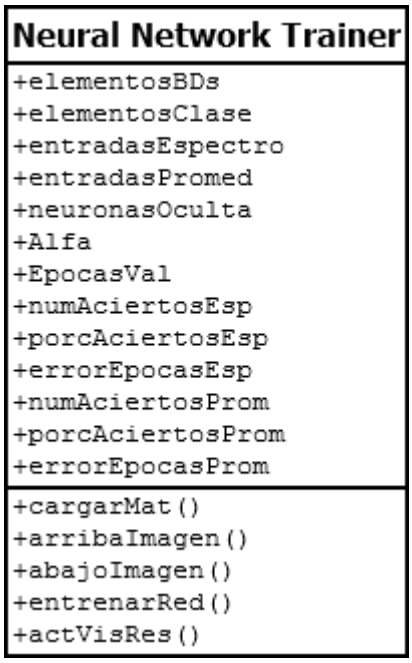


Figure 3.17: UML block for the training of the ANN.

In the figure 3.18, it can be seen a full view of the GUI tab to train the ANN. For demonstration purposes was loaded a MAT file that contains the total number of images for training the networks, in order to see audio signal plots and a typical training progress graph of a neural network. The GUI tab can be divided in three areas, the first one (from top to bottom) is the place to load and visualize the previous generated training data, the second area is where the parameters of the topology and the training of the neural network are set, and finally the third area is where the results of the training are presented in both ways, graphically and with numeric values.

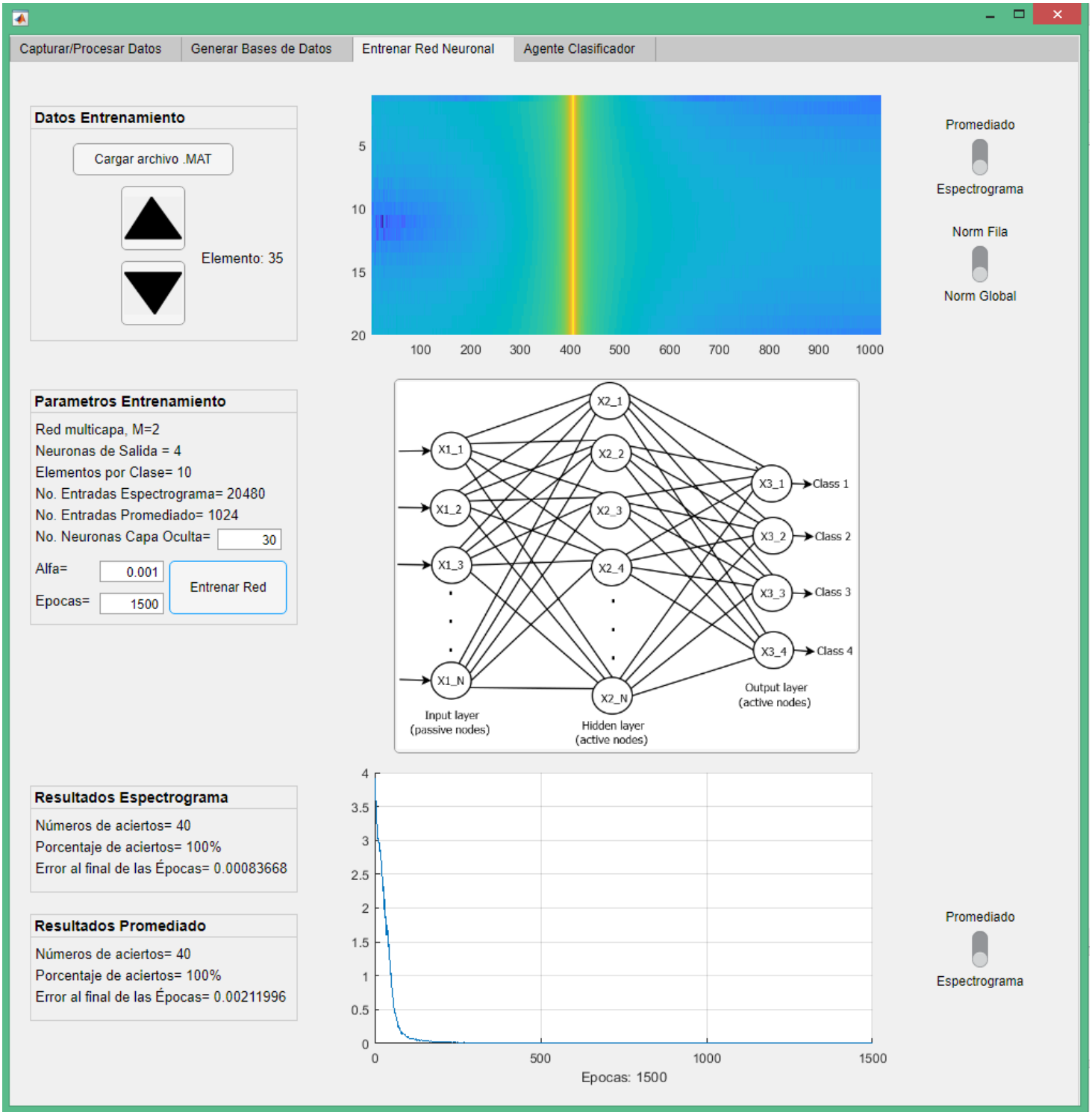


Figure 3.18: GUI Tab to train the ANN.

This software section was designed having in mind to get the capability to easily load a database with the training data of the four different classes and configure basics

parameters, run the training test, and finally see the results. In the table 3.5 are described all the GUI components used for this tab.

"Training Artificial Neural Network" Components		
DATOS ENTRENAMIENTO	Cargar archivo WAV	This button executes the method "cargarMat()", and open a dialog box to load a MAT file, that file contains the training images/signals needed in order to train the Artificial Neural Network.
	Up Arrow	This button executes the method "arribaImagen()", and this button is intended to go up over every image/signal that was saved inside the MAT file.
	Down Arrow	This button executes the method "abajoImagen()", and this button is intended to go down over every image/signal that was saved inside the MAT file.
	Elemento: #	This label indicates the element number displayed in the Plot area, the total number of elements depends on the MAT file.
	Upper Plot Area	This plot area is intended to use it in order to display the Spectrogram or the Averaged Power Spectral Density of the spectrogram of the selected element.
	Promediado/Espectrograma	This switch component is used to select what kind of graphic is chosen to display in the plot area, can be the spectrogram or the average of the spectrogram.
	Norm Fila/Norm Global	This switch component is used to select what kind of data normalization is chosen to display in the plot area and to train the neural network. One normalization (Norm Global) takes one global maximum for the whole image and operates the normalization; the other normalization (Norm Fila) is an applied to every single row of the spectrogram image, so there is a local maximum in every row.
PARAMETROS ENTRENAMIENTO	Red multicapa, M=2	This label indicates the number of layers that the neural network topology has. For this thesis project is a constant value and are two layers. One hidden layer and one output layer.
	Neuronas de Salida = 4	This label indicates the number of neurons that the neural network has in its output layer. For this thesis project is a constant value, and are four neurons. One for every class.
	Elementos por clase = #	This label indicates the number of training elements per class. This parameter is automatically set on the label when the MAT file is loaded.
	No. Entradas Espectrograma = #	This label indicates the number of inputs or input neurons the neural network for training the spectrogram will have. This parameter is automatically set on the label when the MAT file is loaded.
	No. Entradas Promediado = #	This label indicates the number of inputs or input neurons the neural network for training the average of spectrogram will have. This parameter is automatically set on the label when the MAT file is loaded.
	No. Neuronas Capa Oculta= #	This label indicates the number of neurons in the hidden layer of the neural network. This parameter is manually set in the textbox area, its default value is 30.
	Alfa = #	This label indicates the learning rate of the neural network. This parameter is manually set in the textbox area, its default value is 0.001.
	Epocas = #	This label indicates the number of training epochs for the neural network. This parameter is manually set in the textbox area, its default value is 1500.
	Entrenar Red	This button executes the method "entrenarRed()", and start the process of training the neural network with the previously loaded data and parameters .
	Neural Network Image	This image was posted just for informative purposes. It can be seen there the graphical representation of the artificial neural network topology.
RESULTADOS ESPECTROGRAMA	Números de aciertos = #	After the training has been done, some tests are performed in order to validate the results of the training. This test consists, in running a classification demo with the new gotten coefficients (weights and bias) for the neural network. So, this label indicates the number of correct classification elements for the spectrogram.
	Porcentaje de aciertos = #	This label indicates the percentage of correct classification elements for the spectrogram.
	Error al final de las Épocas = #	This label indicates the final error of the neural network when the training is done for the spectrogram.
RESULTADOS PROMEDIADO	Números de aciertos = #	After the training has been done, some tests are performed in order to validate the results of the training. This test consists, in running a classification demo with the new gotten coefficients (weights and bias) for the neural network. So, this label indicates correct number of the classification elements for the average of the spectrogram.
	Porcentaje de aciertos = #	This label indicates the percentage of correct classification elements for the average of the spectrogram.
	Error al final de las Épocas = #	This label indicates the final error of the neural network when the training is done for the average of the spectrogram.
Progress Visualization		This plot area is intended to use it in order to display the Spectrogram and the Averaged Power Spectral training progress. In X axis is displayed the number of training epochs and Y axis the error of training.
Promediado/Espectrograma		This switch component is used to select what kind of graphic is chosen to display in the plot area, can be the spectrogram or the average of the spectrogram training progress.

Table 3.5 Training Artificial Neural Network GUI components.

3.2.4 Software Tab to classify the acoustic signals.

In this section it will be described the software tool that implements the agent that is on charge to perform the classification task. Basically, the process to use this tab is to load the weights and bias coefficients previously generated, and finally acquire an audio sample in order to use it to test the behavior of the now trained neural network. For finally visualize the results of both neural networks (Spectrogram and mean of spectrogram).

In the figure 3.19, it's shown a UML block with ten attributes and three methods, the first two attributes (from top to bottom) are related with parameters of the acquisition of the audio signal (sample frequency and time of capture), the next eight are indicators of the final results for the spectrogram and the mean of the spectrogram neural networks. There are also three methods, the first one is for load the weights and bias coefficients, the next method is intended to acquire the sound that will be introduced to the trained neural network, and the last one is intended select the visualization of the images of the acquired audio.

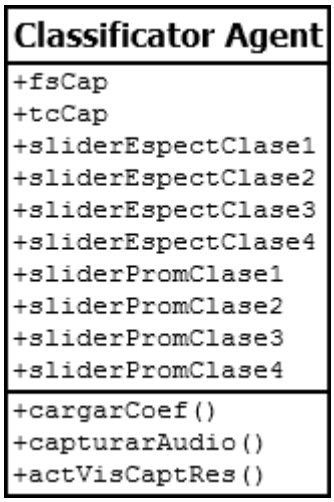


Figure 3.19: UML block for the Classificator Agent.

In the figure 3.20, it can be seen a full view of the last GUI tab intended to test the trained ANN. For demonstration purposes was loaded a MAT file that contains the weights and bias coefficients previously calculated, and also was captured an audio signal in the range frequency of the images with which were trained the ANN, in order to see the

classification results. The GUI tab can be divided in two areas, the first one (from top to bottom) is the place to load the training coefficients and to acquire and visualize the acoustic signal to be evaluate it by the classification neural network, the second area is where the classification results are presented for both inputs, the spectrogram and the mean of the spectrogram.

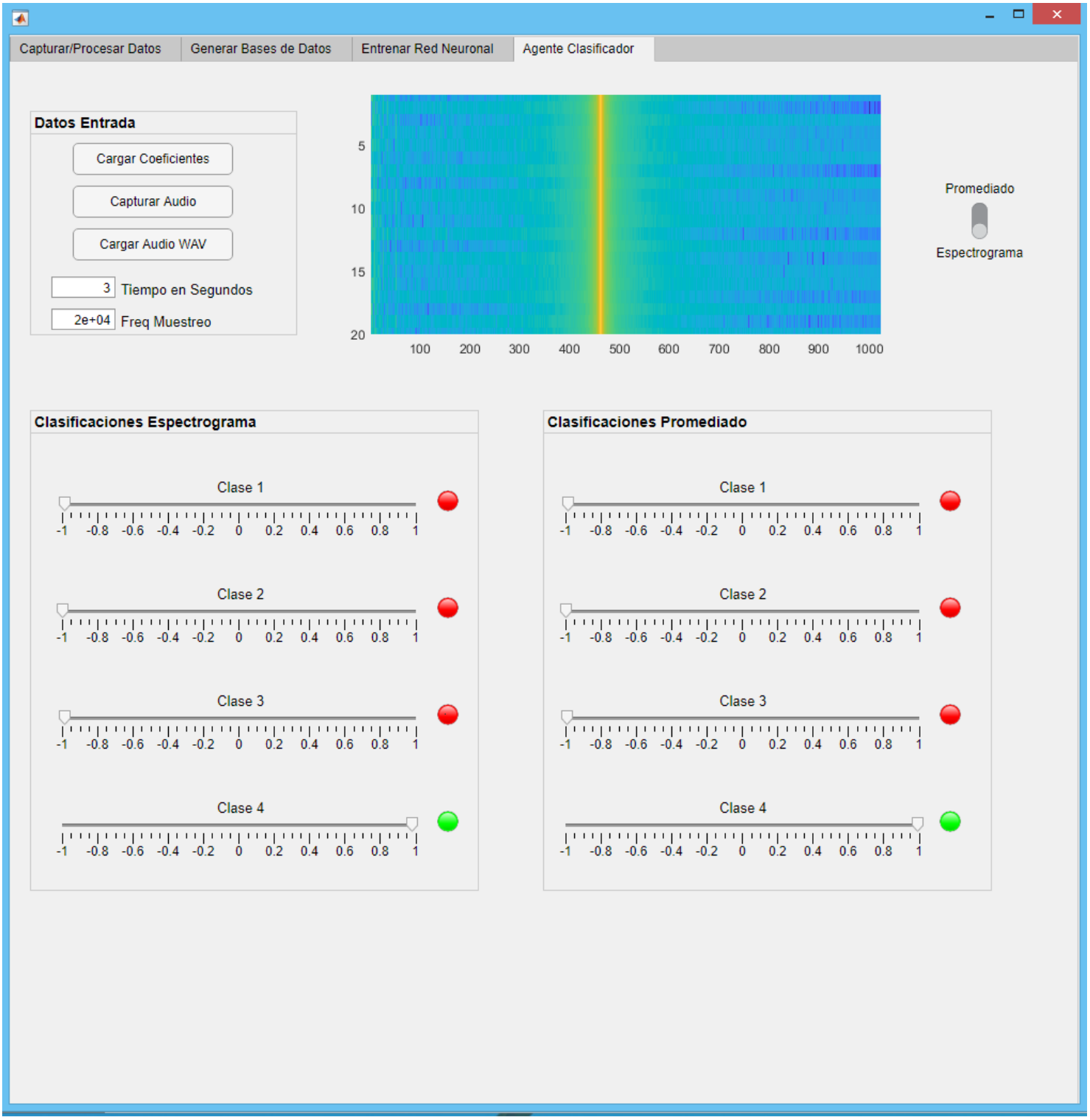


Figure 3.20: GUI Tab to classify the acoustic signals.

" Classificator Artificial Neural Network" Components		
DATOS ENTRADA	Cargar Coeficientes	This button executes the method "cargarCoef()", and open a dialog box to load a MAT file, that file contains the weights and bias that were calculated when the neural network was trained. Those coefficients are used in order to create the final neural network and test its behavior.
	Capturar Audio	This button executes the method "capturarAudio()", and this button is intended to acquire an audio signal by the audio card of the computer. This audio later is converted to the spectrum domain to finally be introduced in the respective inputs of the neural networks in order to see its classifications responses.
	Cargar Audio WAV	This button executes the method "cargarAudio()", and this button is intended to load an audio WAV file. This audio later is converted to the spectrum domain to finally be introduced in the respective inputs of the neural networks in order to see its classification responses.
	Tiempo en segundos: #	This label indicates the duration time in seconds that the audio will be acquired. Its default value is three seconds.
	Freq Muestreo	This label indicates the sample rate in samples per second applied to the audio that will be acquired. Its default value is 20Ksps.
	Plot Area	This plot area is intended to use it in order to display the Spectrogram or the Averaged of the Spectrogram for the previously acquired audio signal.
	Promediado/Espectrograma	This switch component is used to select what kind of graphic is chosen to display in the plot area, can be the spectrogram or the average of the spectrogram of the previous acquired audio signal.
ESPECTROGRAMA	Clase 1	This slider component is used to display the output value of the first neuron (class 1) for the Spectrogram neural network for the output layer. And the led associated to this slider indicates when the value is maximum (between the four classes) turns on green, and red when the value is not the maximum. With this numeric value can be measured how good is the classification, for a value closer to one can be infered that there is a high probability of a good classification, and the further away from one, the probability of a good classification decays.
	Clase 2	This slider component is used to display the output value of the second neuron (class 2) for the Spectrogram neural network for the output layer. And the led associated to this slider indicates when the value is maximum (between the four classes) turns on green, and red when the value is not the maximum. With this numeric value can be measured how good is the classification, for a value closer to one can be infered that there is a high probability of a good classification, and the further away from one, the probability of a good classification decays.
	Clase 3	This slider component is used to display the output value of the third neuron (class 3) for the Spectrogram neural network for the output layer. And the led associated to this slider indicates when the value is maximum (between the four classes) turns on green, and red when the value is not the maximum. With this numeric value can be measured how good is the classification, for a value closer to one can be infered that there is a high probability of a good classification, and the further away from one, the probability of a good classification decays.
	Clase 4	This slider component is used to display the output value of the fourth neuron (class 4) for the Spectrogram neural network for the output layer. And the led associated to this slider indicates when the value is maximum (between the four classes) turns on green, and red when the value is not the maximum. With this numeric value can be measured how good is the classification, for a value closer to one can be infered that there is a high probability of a good classification, and the further away from one, the probability of a good classification decays.
PROMEDIADO	Clase1	This slider component is used to display the output value of the first neuron (class 1) for the Average of Spectrogram neural network for the output layer. And the led associated to this slider indicates when the value is maximum (between the four classes) turns on green, and red when the value is not the maximum. With this numeric value can be measured how good is the classification, for a value closer to one can be infered that there is a high probability of a good classification, and the further away from one, the probability of a good classification decays.
	Clase2	This slider component is used to display the output value of the second neuron (class 2) for the Average of Spectrogram neural network for the output layer. And the led associated to this slider indicates when the value is maximum (between the four classes) turns on green, and red when the value is not the maximum. With this numeric value can be measured how good is the classification, for a value closer to one can be infered that there is a high probability of a good classification, and the further away from one, the probability of a good classification decays.
	Clase 3	This slider component is used to display the output value of the third neuron (class 3) for the Average of Spectrogram neural network for the output layer. And the led associated to this slider indicates when the value is maximum (between the four classes) turns on green, and red when the value is not the maximum. With this numeric value can be measured how good is the classification, for a value closer to one can be infered that there is a high probability of a good classification, and the further away from one, the probability of a good classification decays.
	Clase 4	This slider component is used to display the output value of the fourth neuron (class 4) for the Average of Spectrogram neural network for the output layer. And the led associated to this slider indicates when the value is maximum (between the four classes) turns on green, and red when the value is not the maximum. With this numeric value can be measured how good is the classification, for a value closer to one can be infered that there is a high probability of a good classification, and the further away from one, the probability of a good classification decays.

Table 3.6 GUI components for the Classificator Agent.

3.2.5 Dual channel function generator Android App.

In order to do some rapid training and testing for a neural network implemented, it can be used a signal generator app for a mobile phone, that tool can be used to generate and introduce some synthetic audio signals to the ANN through the microphone input of the computer. This app by Keuwlsoft puts a waveform on the audio output at 44.1 kHz and with a resolution of 16-bit. Two separate waveforms can be output to the left and right audio output channels respectively [14]. In the table 3.7 are listed the main features of the app.

Function Generator APP from KEUWLSOFT	
Signal waveforms	Sine, square & triangular
Frequency range	From 1 mHz up to 22 kHz
Amplitude	As percentage 0-100%
Duty Cicle	For square or skew triangular waveforms
Shift	The start phase of waveforms
Sweep frequency or amplitude	Single, Repeat & Bounce modes
Modulation	Amplitude (AM) & Frequency (FM)
Burst mode	For a specific number of waveforms (1-10000)
Noise	White & Pink generator

Table 3.7 Main features of the function generator App.

In the figure 3.21 can be seen a screenshot of the app, for more information about how to use it, visit the official site of the app.

Figure 3.21: Function generator Android App.

3.3 Experimentation Development

In this second part of the chapter 3, it will be described how to perform the experiment in order to test the hypothesis stated in the section 1.3. Therefore, the first thing to do is to verify that the implementation of the algorithm of neural network is working properly for four different kind of topologies using synthetic acoustic signals as inputs, the next step is to start to acquire the real audio signals for the four different kind of maritime vessels, to finally test the final Artificial Neural Networks for four different kind of topologies, with the real acoustic signature audios previously acquired.

3.3.1 Test of performance for the ANN with synthetic signals

A neural network can have many diverse topologies, and every single topology has its own behavior, so, a very important part of the experimentation is to precise the topology. In order to precise the topology, it is proposed to use synthetic acoustic signals to train the neural networks and to run tests for the ANN, with the end to observe behaviors. In the figure 3.22, it can be seen the proposed schema for synthetic audio generation and acquisition.

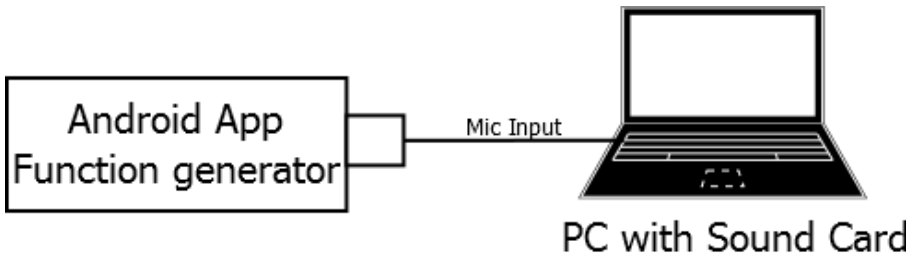


Figure 3.22: Schema for synthetic audio generation.

For the whole ANN performance test, it will be shown all the fixed and the variable configuration parameters used for the function generator, and for the signal acquisition tool, and for the neural network trainer tool in the table 3.8.

The audio signals were generated with the function generator App with the configuration parameters of the table 3.8 and their respective frequencies in order to send the audio to the computer with the help of a male to male audio cable, as it can be seen in the figure

3.23. Later, it is necessary to set the parameters for the acquisition tool software, to finally start acquiring the audio signals for every proposed frequency (40 in total, ten per class).

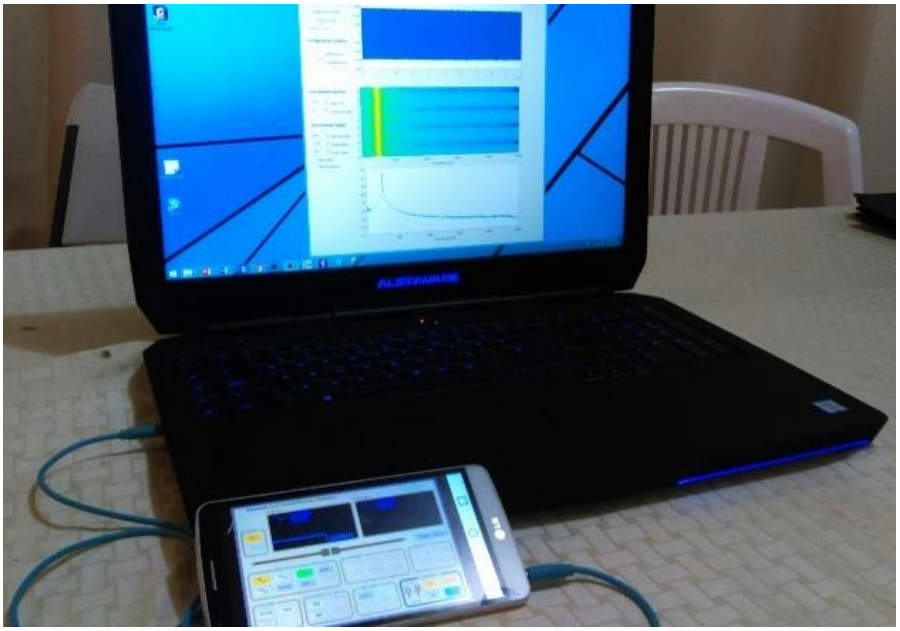


Figure 3.23: Audio synthetic signals acquisition process.

Test Parameters value setting		
FUNCTION GENERATOR	Waveform Type (fixed parameter)	Sine
	Amplitude (fixed parameter)	50% at 5% of Output Volume
	Bias (fixed parameter)	0.00%
	Phase (fixed parameter)	0.0°
	Frequency, Hz (fixed parameter)	Class 1= 750,800,850,900,950,1000,1050,1100,1150,1200 Class 2= 1750,1800,1850,1900,1950,2000,2050,2100,2150,2200 Class 3= 2750,2800,2850,2900,2950,3000,3050,3100,3150,3200 Class 4= 3750,3800,3850,3900,3950,4000,4050,4100,4150,4200
SIGNAL ACQUISITION	Acquisition Time (fixed parameter)	5 seconds
	Sample Rate (fixed parameter)	20 KSPS
	FFT Length(fixed parameter)	1024 samples
	Waterfall Length (fixed parameter)	20 samples
ANN TRAINER	Multilayer Network (Fixed parameter)	2 layers
	Output Layer (Fixed parameter)	4 Neurons
	Training elements per Class (Fixed parameter)	10 elements
	Input Layer number of elements for Spectrogram ANN (Fixed parameter)	20480 Neurons
	Input Layer number of elements for Average of Spectrogram ANN (Fixed parameter)	1024 Neurons
	Hidden Layer number of elements (Variable parameter)	30, 100, 200, 500 Neurons
	Learning Rate (Alpha) (Fixed parameter)	0.001
	Number of Training Epochs (Fixed parameter)	1500

Table 3.8 Test parameters value settings.

In the figure 3.24, it can be seen the ten training images for the class 1, for both the spectrogram and mean of spectrogram Artificial Neural Networks. The frequency range is from 750 to 1200 Hz.

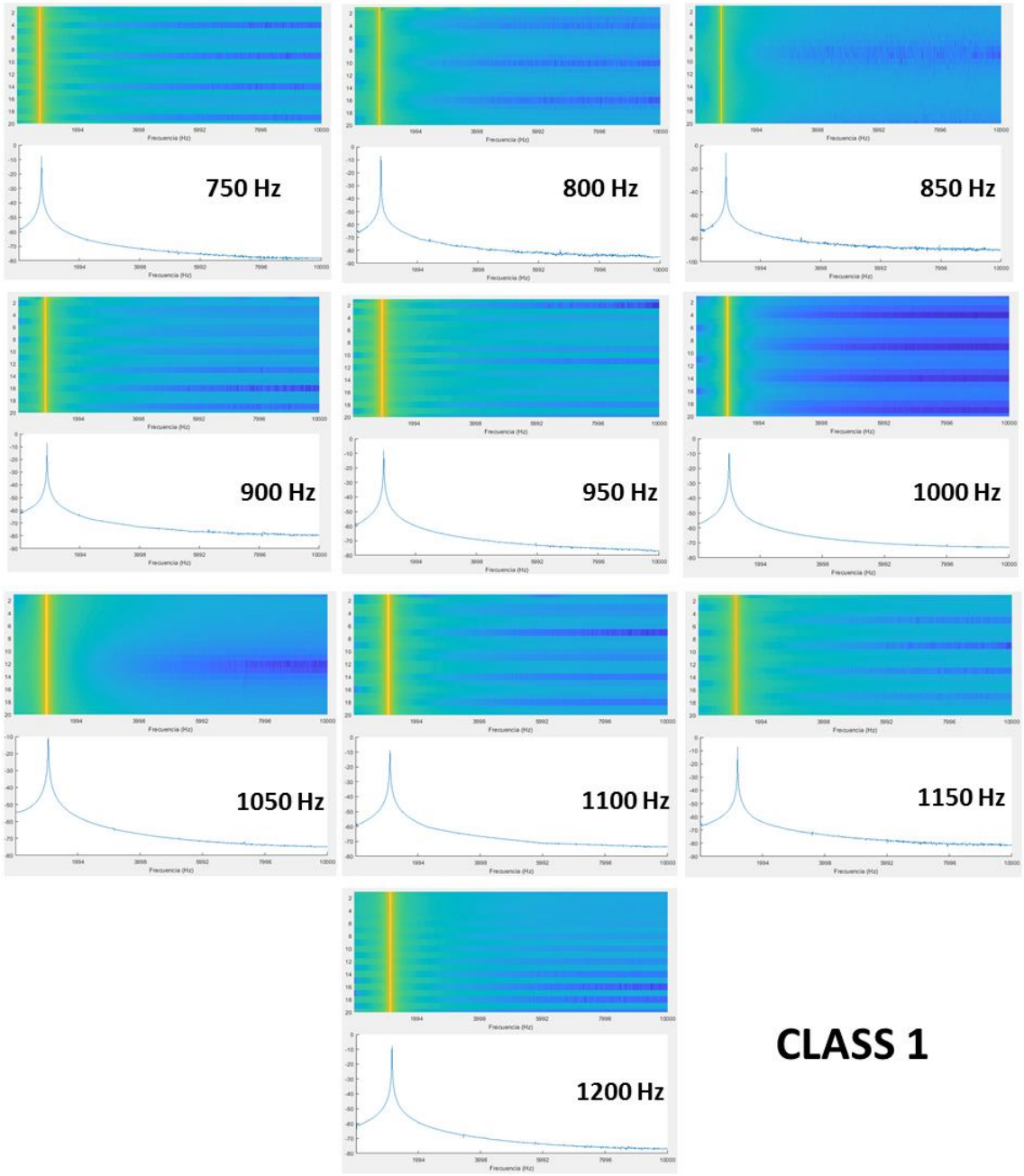


Figure 3.24: Training images for Class 1.

In figure 3.25, it can be seen the ten training images for the class 2, for both the spectrogram and mean of spectrogram Artificial Neural Networks. The frequency range is from 1750 to 2200 Hz.

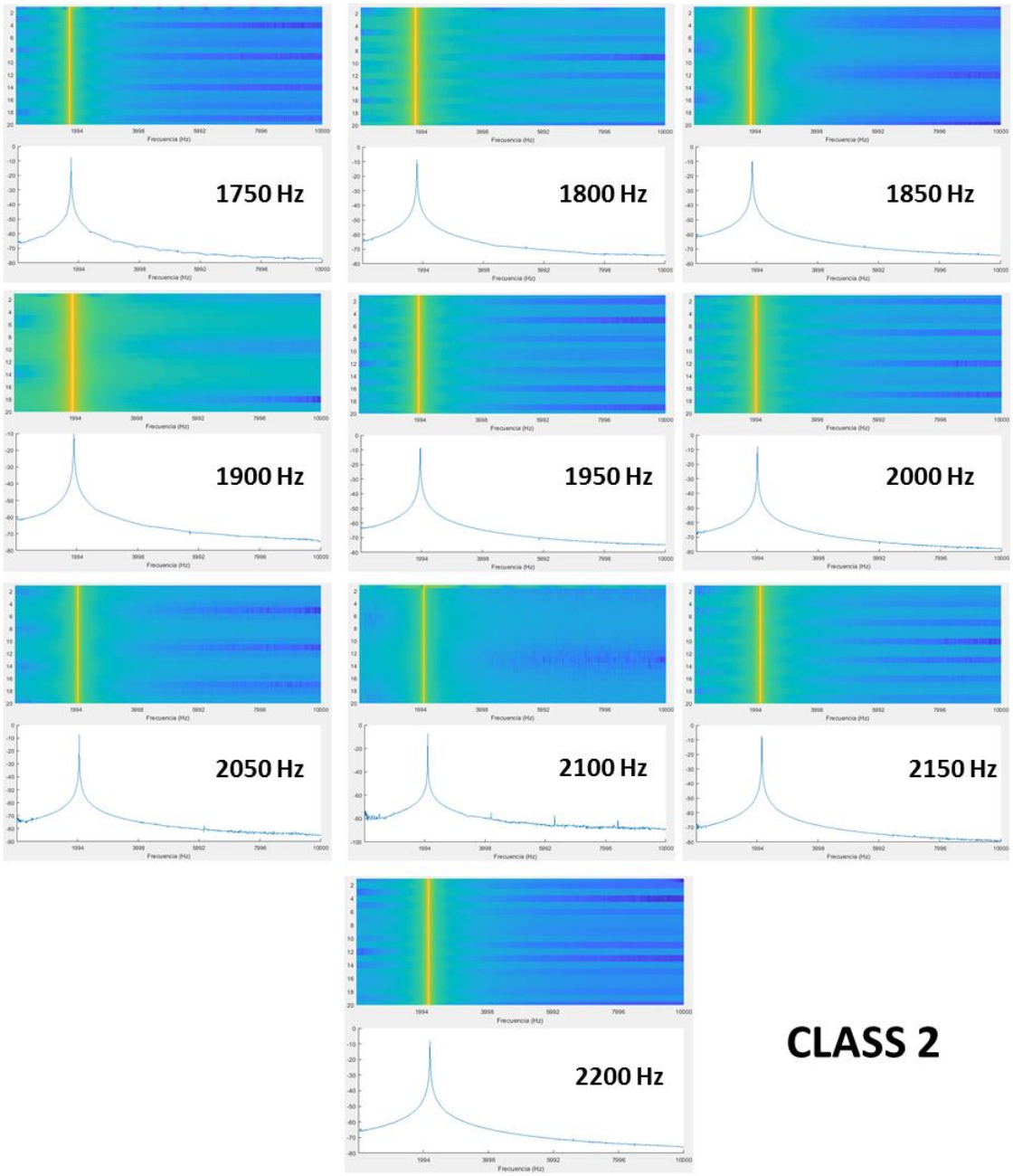


Figure 3.25: Training images for Class 2.

In the figure 3.26, it can be seen the ten training images for the class 3, for both the spectrogram and mean of spectrogram Artificial Neural Networks. The frequency range is from 2750 to 3200 Hz.

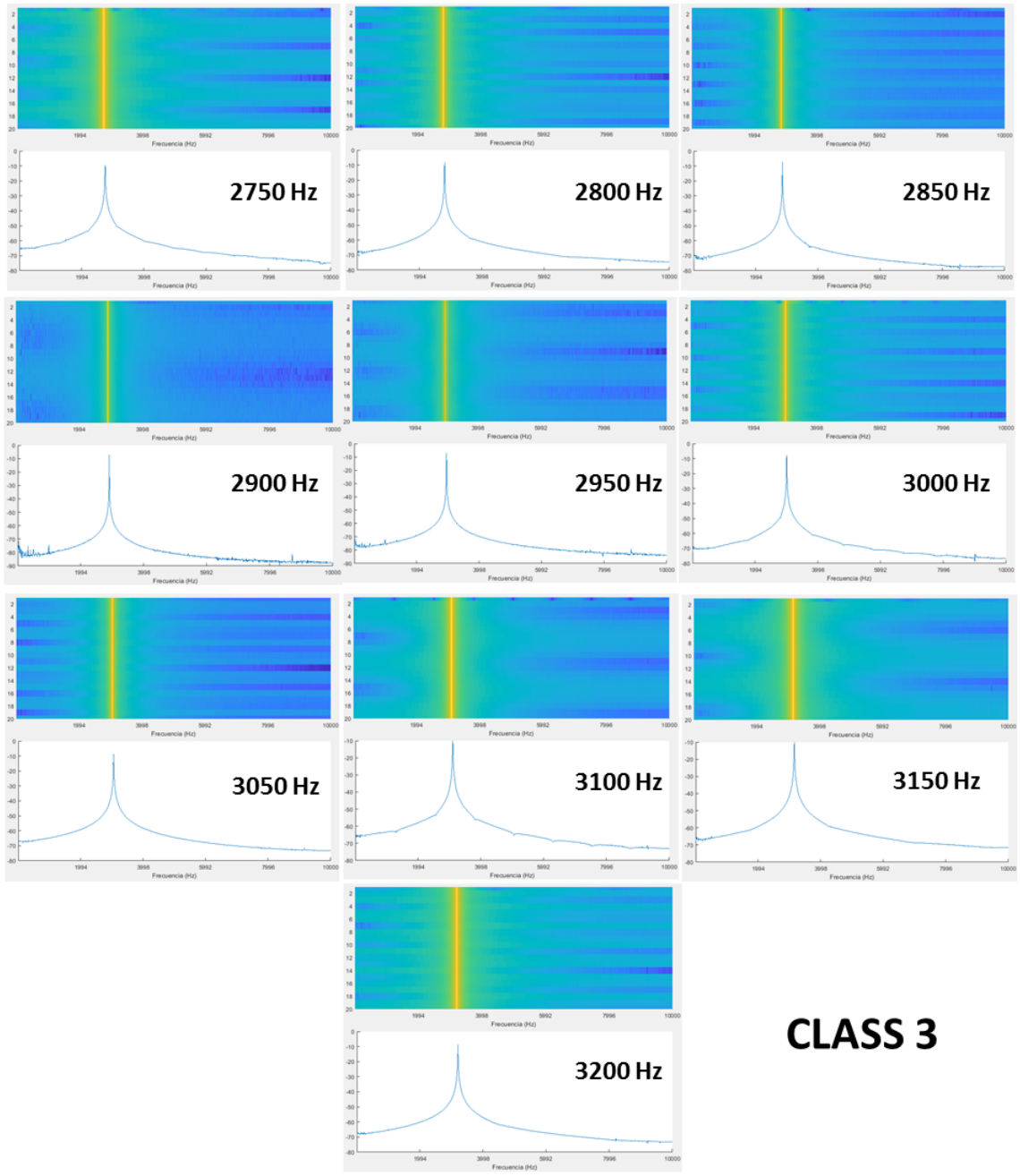


Figure 3.26: Training images for Class 3.

In figure 3.27, it can be seen the ten training images for the class 4, for both the spectrogram and mean of spectrogram Artificial Neural Networks. The frequency range is from 3750 to 4200 Hz.

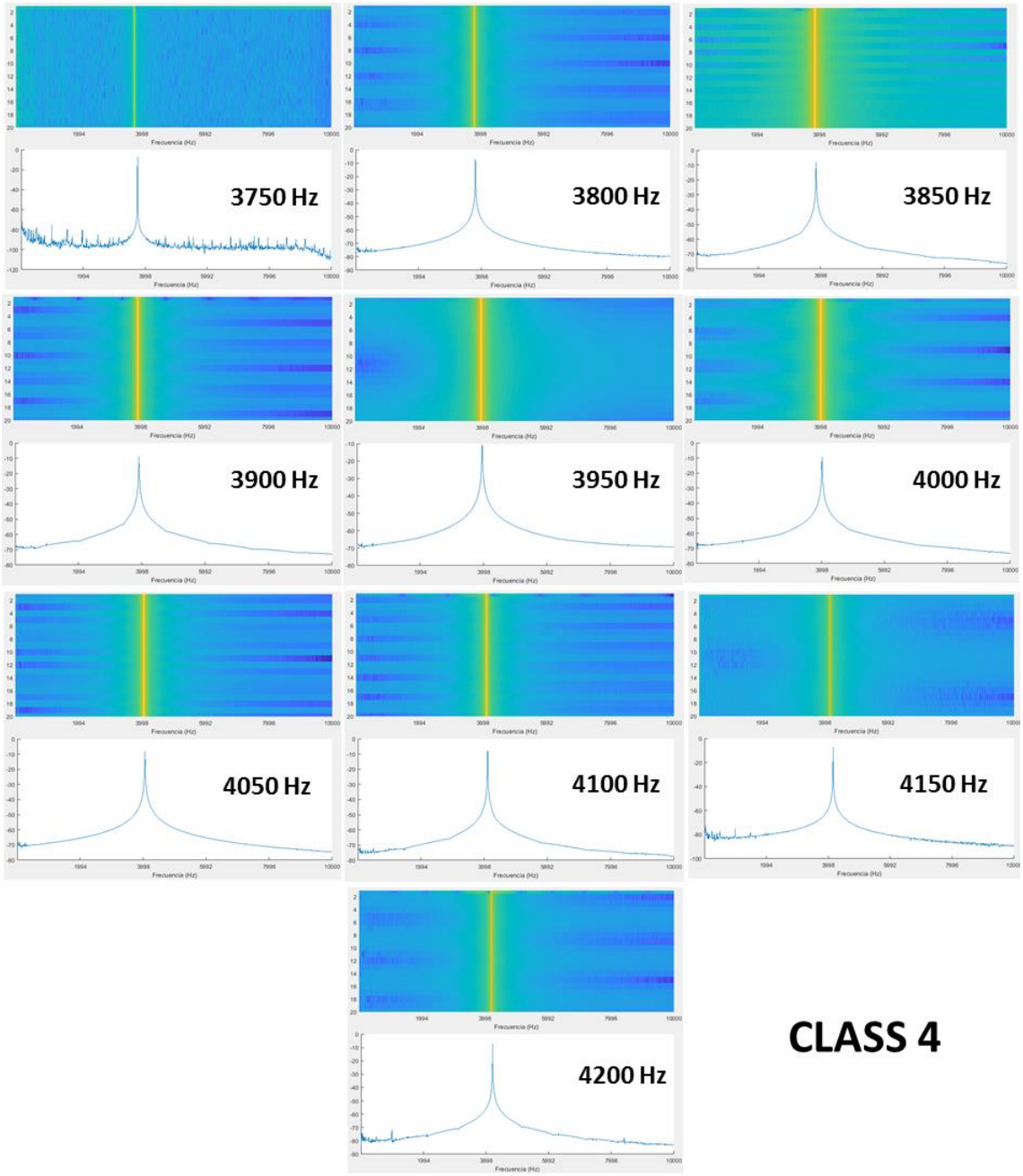


Figure 3.27: Training images for Class 4.

When all the training audios for the four classes are saved in the computer, now is time to start loading and sorting those files per class to finally generate the single training file that it is called database. In figure 3.28, it can be seen the result of loading ten audio files per class.

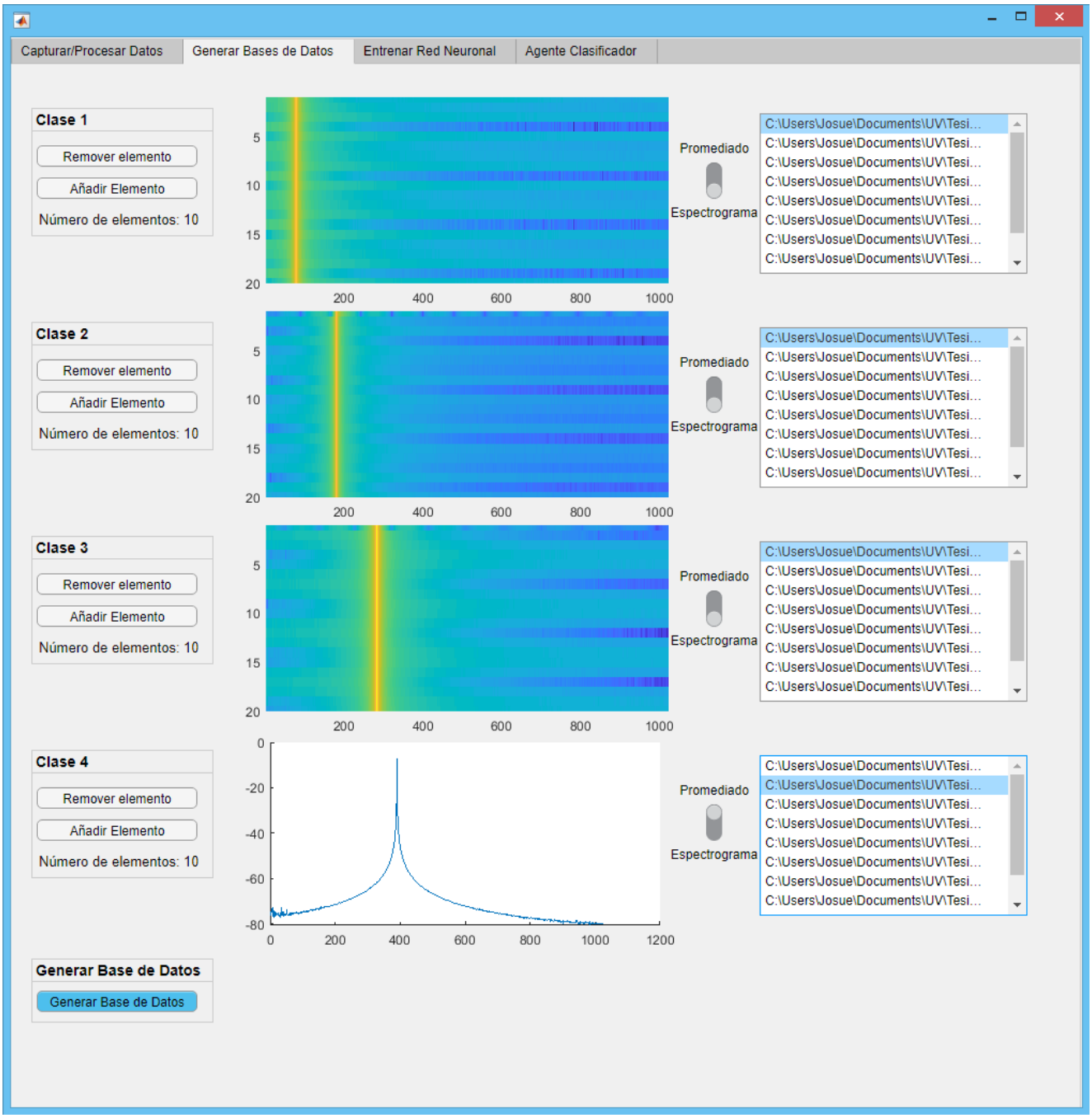


Figure 3.28: Loading and sorting audio files.

When all the ten audio files for the four different classes have been sorted with the format of the figure 3.14, they are good to be loaded in the trainer GUI and start the training process. For the below figure, it was trained the neural network with 30 neurons in the hidden layer and can be seen the training results in the bottom left side part of the GUI. In the next chapter can be found the summarized results for every kind of tested topology.

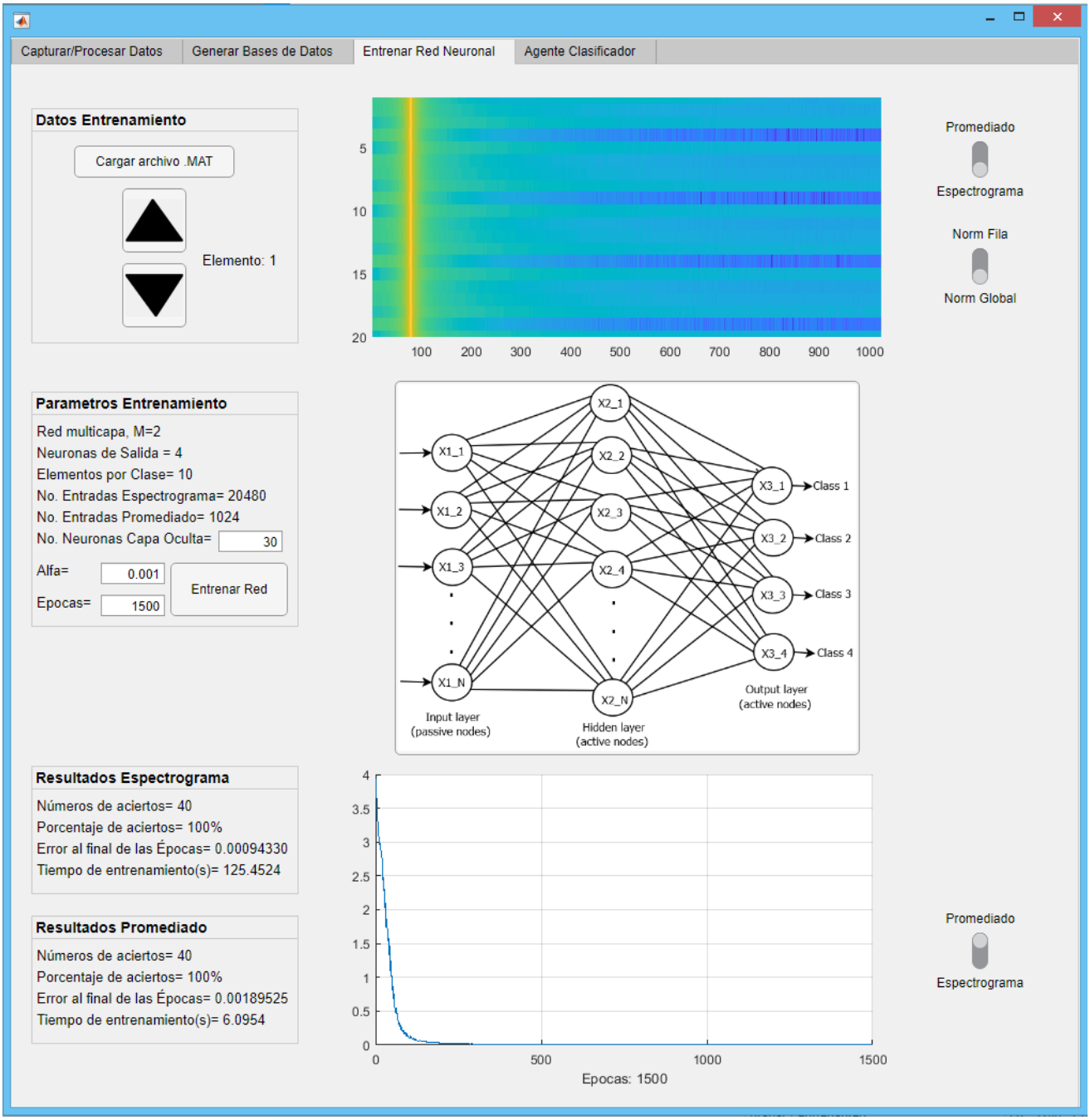


Figure 3.29: Training results for the ANN with synthetic audios.

When the training process is done, then the coefficients found by the gradient descend algorithm are ready to be loaded in the Classificator agent in order to verify its behavior. In the figure 3.30 can be seen the result of put under test a synthetic signal of 2kHz and the classification result of the agent was to point the Class 2 (the higher value of the last four neurons, marked with the green led) for both inputs, the spectrogram and the mean

of the spectrogram. In the next chapter can be found the summarized results for every kind of tested topology with their respective input audio.

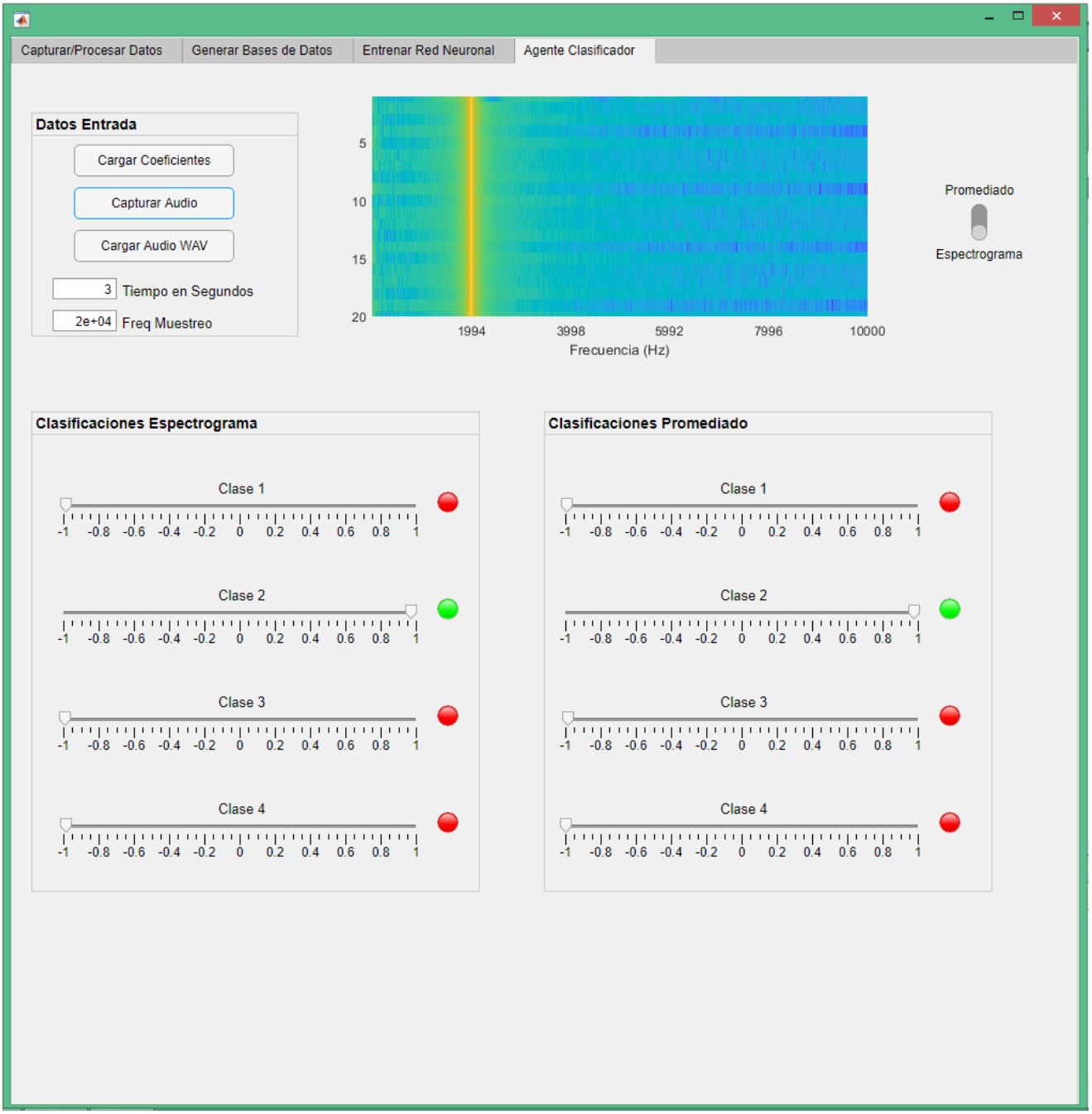


Figure 3.30: Testing the behavior of the Classifier Agent.

3.3.2 Test of performance for the ANN with real-world signals

In this section, it will be presented the same tests than the previous section but with the difference of the type of input signals, for this case it will be used real-world signals. Also, in the previous section every test was performed with synthetic audio signals, in order to easily test the implementations of the ANNs and correct the design-

implementation failures in a quickly way, and also was intended to test rapidly different ANN topologies and get the first results and some implementation experiences, and finally have better references in order to select the better ANN topology for the testing of the classificator with real-world acoustic signatures.

In order to accomplish the tests for this section, it was necessary to use the whole acquisition system (described in figure 3.1) in order to collect the sounds of the four proposed classes for this thesis work. The following maritime vessels were used.

For the Class 1, the sounds of a submersible ROV (Remote Operated Vehicle) with an electric 320 KV brushless motor and 5-blade propeller [15] are going to be used to train the artificial neural network and to test the performance for the classification agent for this class.

Figure 3.31: Class 1, Seawolf ROV [15].

For the Class 2, the sounds of a merchant ship, with a typical diesel engine with usually five or four blade propellers and around 102 RPMs, are going to be used to train the artificial neural network and to test the performance for the classification agent for this class.

Figure 3.32: Class 2, Merchant ship.

For the Class 3, the sounds of a fishing boat with a typical diesel engine and around 2300 RPMs, are going to be used to train the artificial neural network and to test the performance for the classification agent for this class.

Figure 3.33: Class 3, fishing boat

Finally, for the Class 4, the sounds of a motorboat with an outboard gasoline engine and around 4000 RPMs and three or four blade propellers, are going to be used to train the

artificial neural network and to test the performance for the classification agent for this class.

Figure 3.34: Class 4, motorboat with outboard engine.

In order to collect the audio signals for the submersible ROV, it was used the facilities and the equipment of the laboratory of underwater acoustics at INIDETAM.

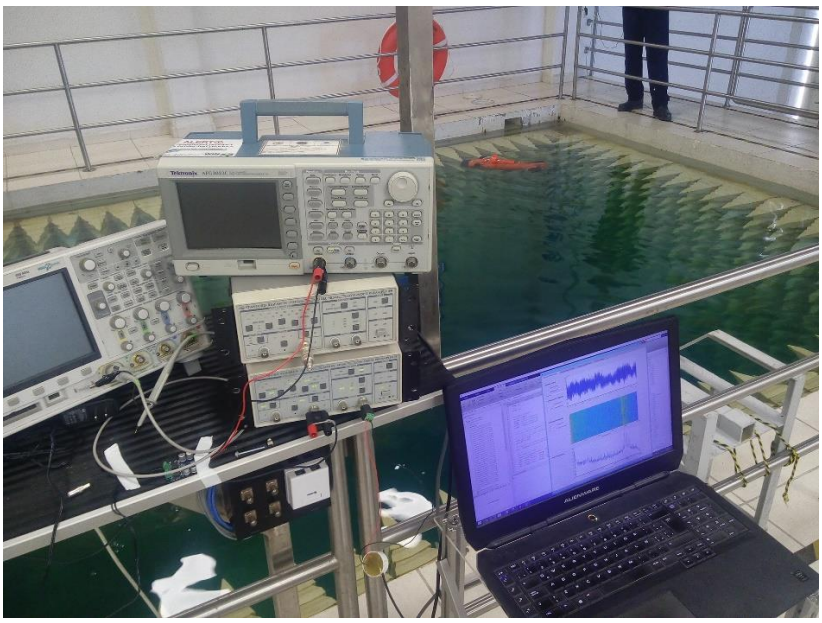


Figure 3.35: Collecting audio signals for the Class 1 ROV.

And for collecting the audio signals of the class 2, 3 and 4, was necessary to use a motorboat in order to chase the merchant ships, the fishing boats, and other motorboats with outboard engine.

Figure 3.36: Collecting audio signals for the Class 2,3 and 4.

In figure 3.37, it can be seen ten training images for the class 1 that belongs to the submersible ROV, for both the spectrogram and mean of spectrogram Artificial Neural Networks. The first five images were captured when the ROV was operating at low speed and the last five at medium speed, all those signals were captured in a hydroacoustic tank with a controlled environment, free of external noise and acoustic reverberations.

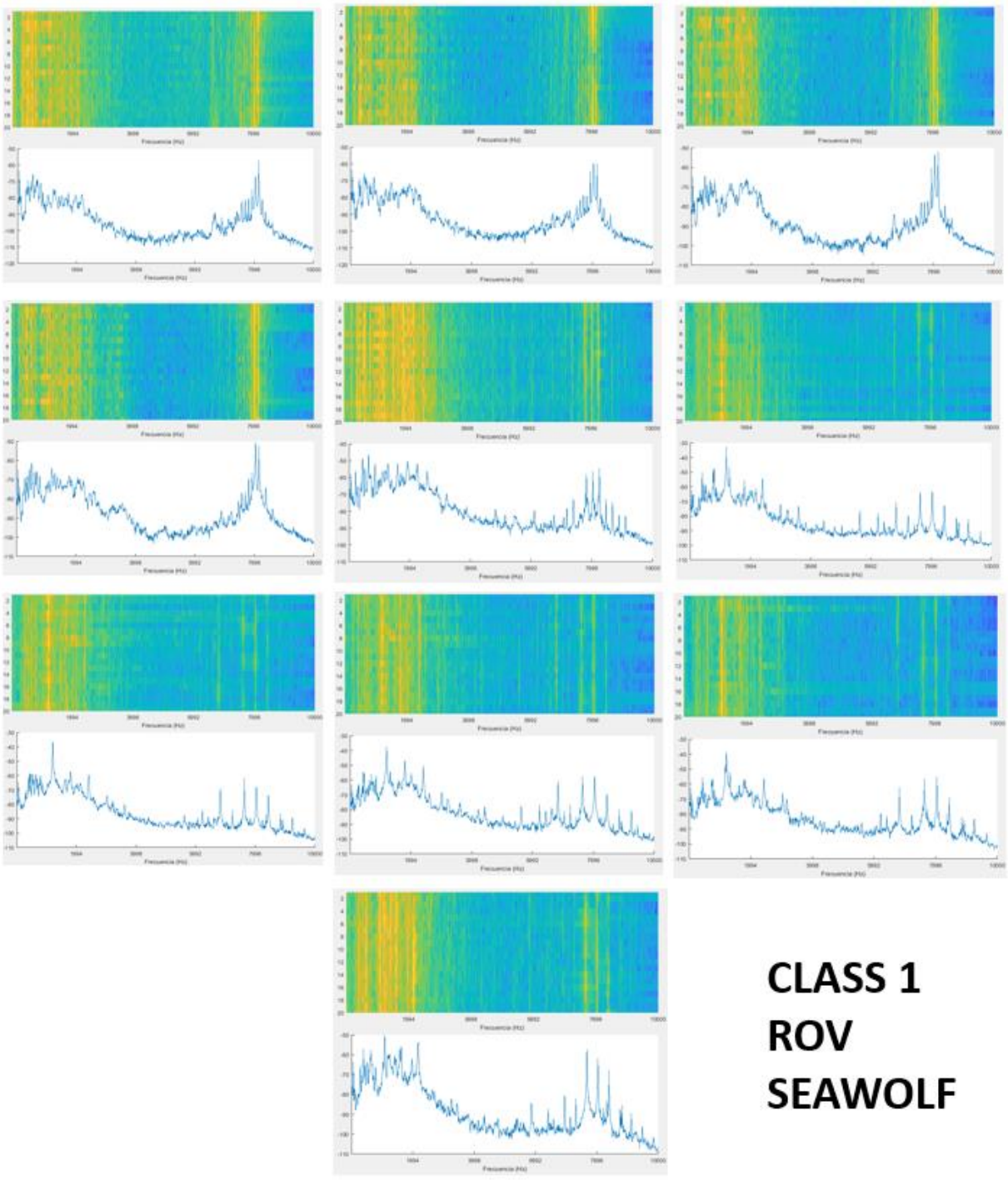


Figure 3.37: Training images for Class 1 (ROV Seawolf).

In figure 3.38, it can be seen ten training images for the class 2 that belongs to a merchant ship, for both the spectrogram and mean of spectrogram Artificial Neural Networks. The

ten training images were filtered using a low pass filter with a cut frequency of 5kHz in order to reduce the environmental noise.

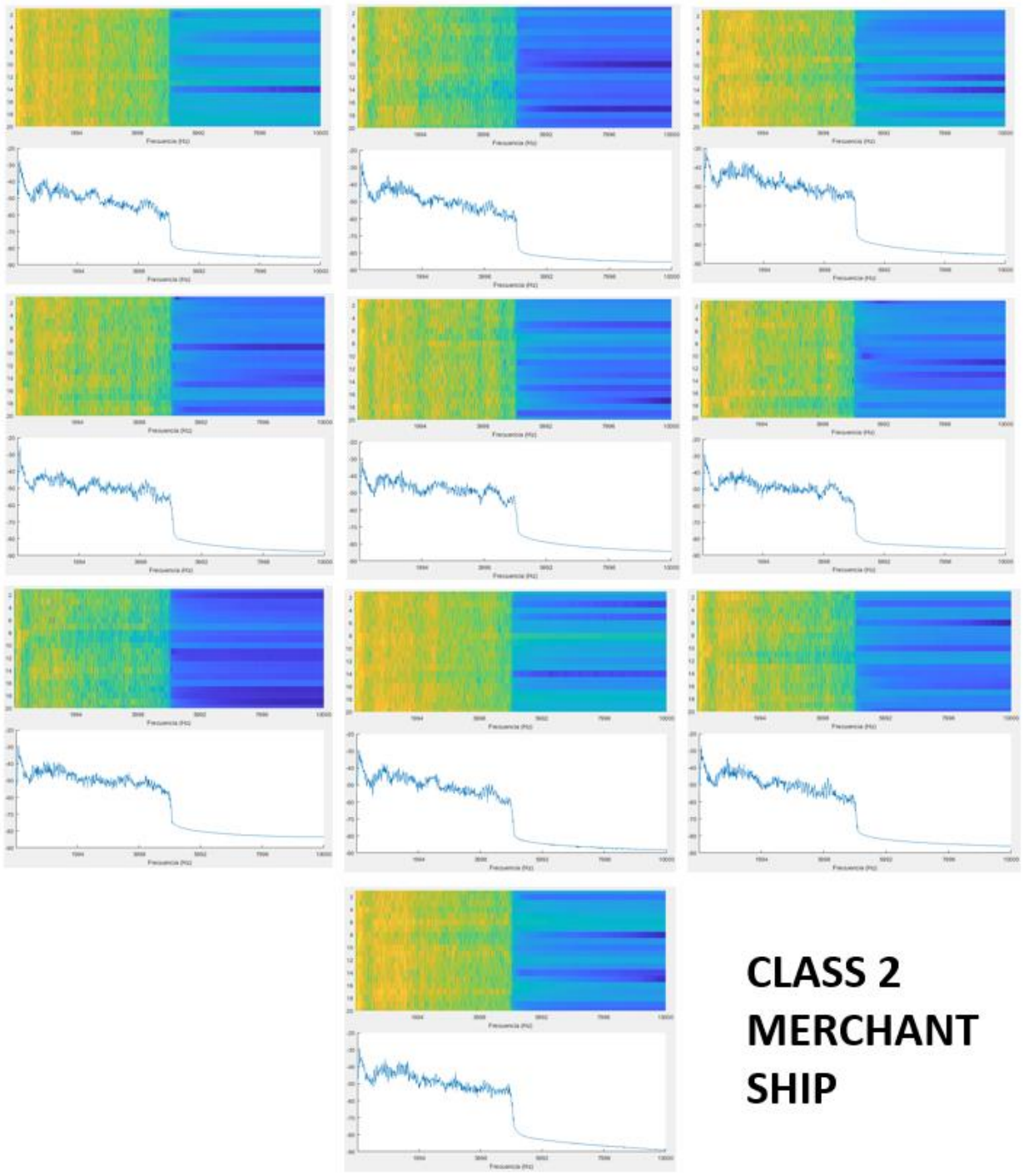


Figure 3.38: Training images for Class 2 (Merchant Ship).

In figure 3.39, it can be seen ten training images for the class 3 that belongs to a fisher boat, for both the spectrogram and mean of spectrogram Artificial Neural Networks. The ten training images were filtered using a low pass filter with a cut frequency of 5kHz in order to reduce the environmental noise.

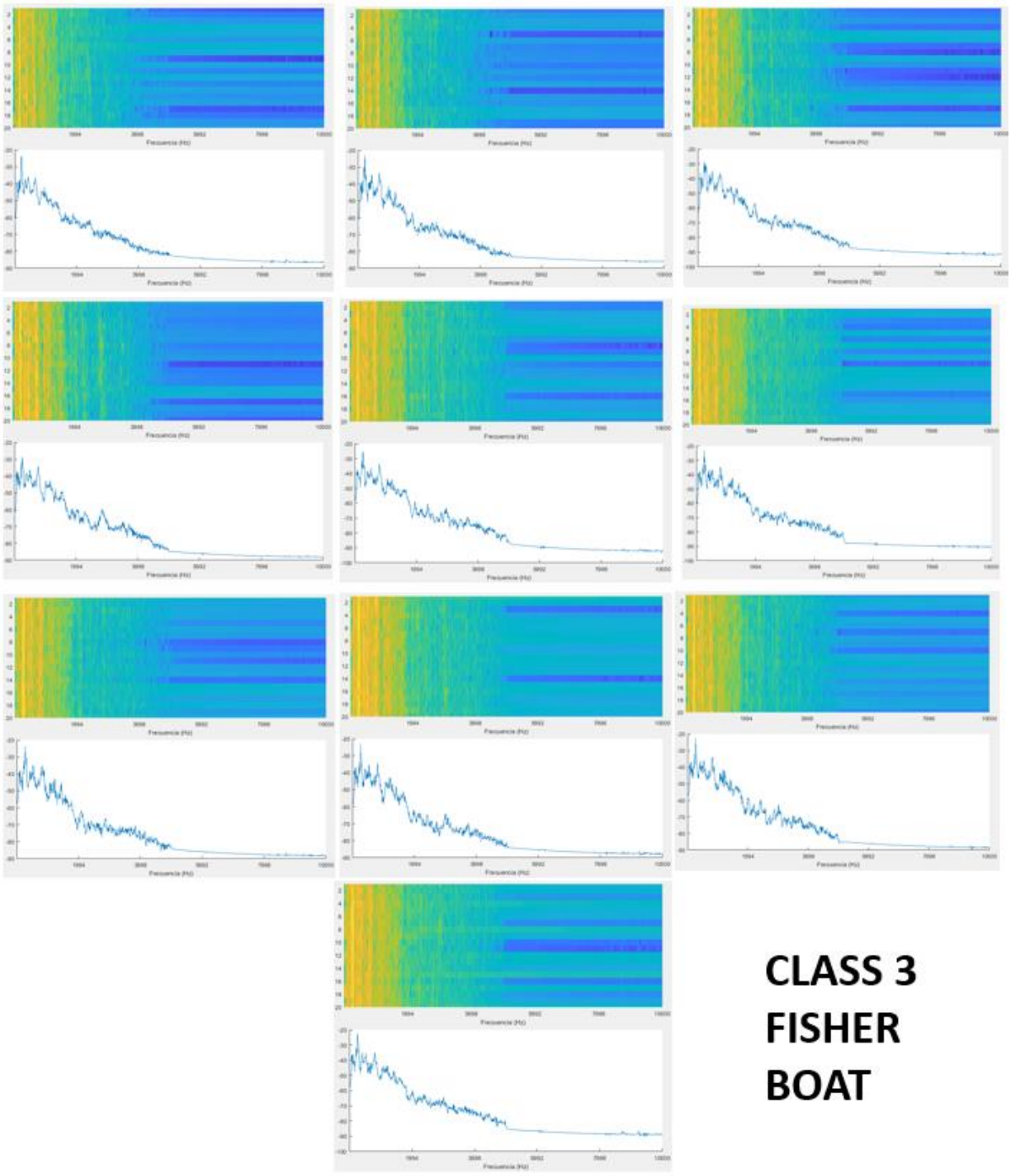


Figure 3.39: Training images for Class 3 (Fisher boat).

In figure 3.40, it can be seen ten training images for the class 4 that belongs to a motorboat with an outboard engine, for both the spectrogram and mean of spectrogram Artificial Neural Networks. The ten training images were filtered using a low pass filter with a cut frequency of 5kHz in order to reduce the environmental noise.

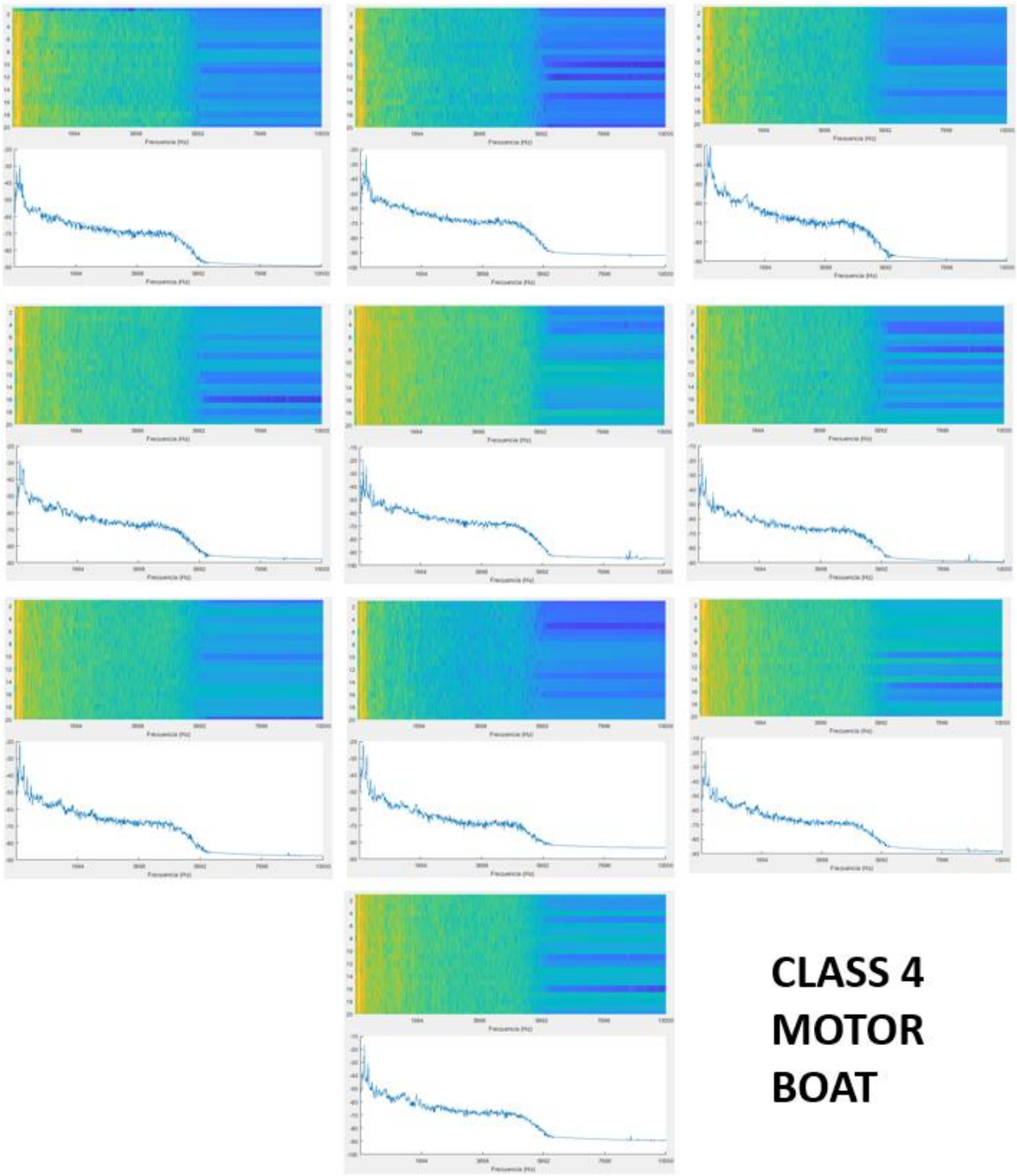


Figure 3.40: Training images for Class 4 (Motor boat).

When the acquisition process for the acoustic signatures for the four maritime vessel classes is done, it is time to start loading and sorting those files per class to finally generate the single training file that it is called database. In figure 3.41, it can be seen the result of loading ten audio files for each class.

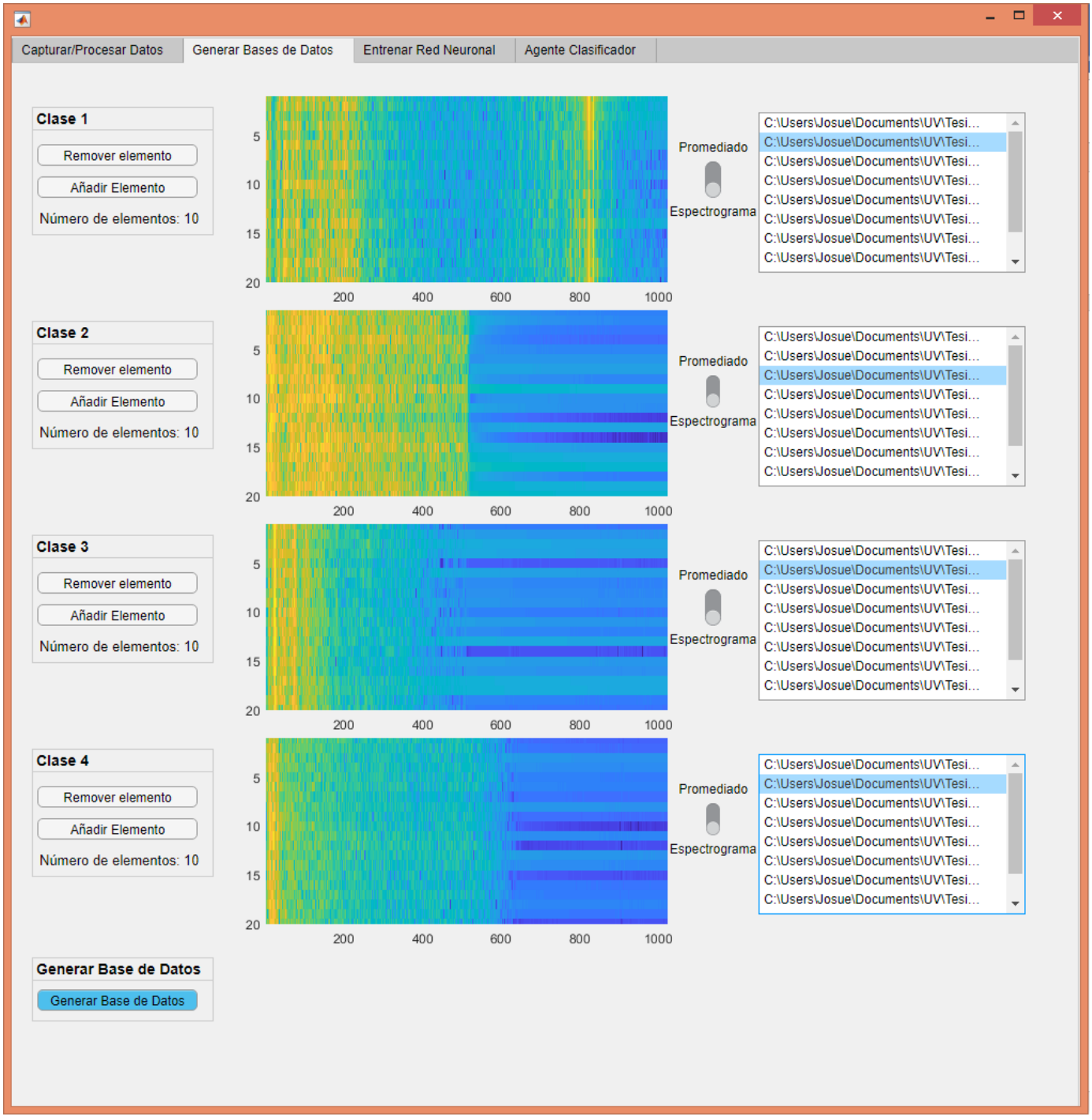


Figure 3.41: Loading and sorting audio files.

When all the ten audio files for the four different classes have been sorted with the format of the figure 3.14, they are good to be loaded in the trainer GUI and start the training process. For the below figure, it was trained the neural network with 30 neurons in the

hidden layer and can be seen the training results in the bottom left side part of the GUI. In the next chapter in the section 4.2 can be found the summarized performance results for the classifier powered with the ANN.

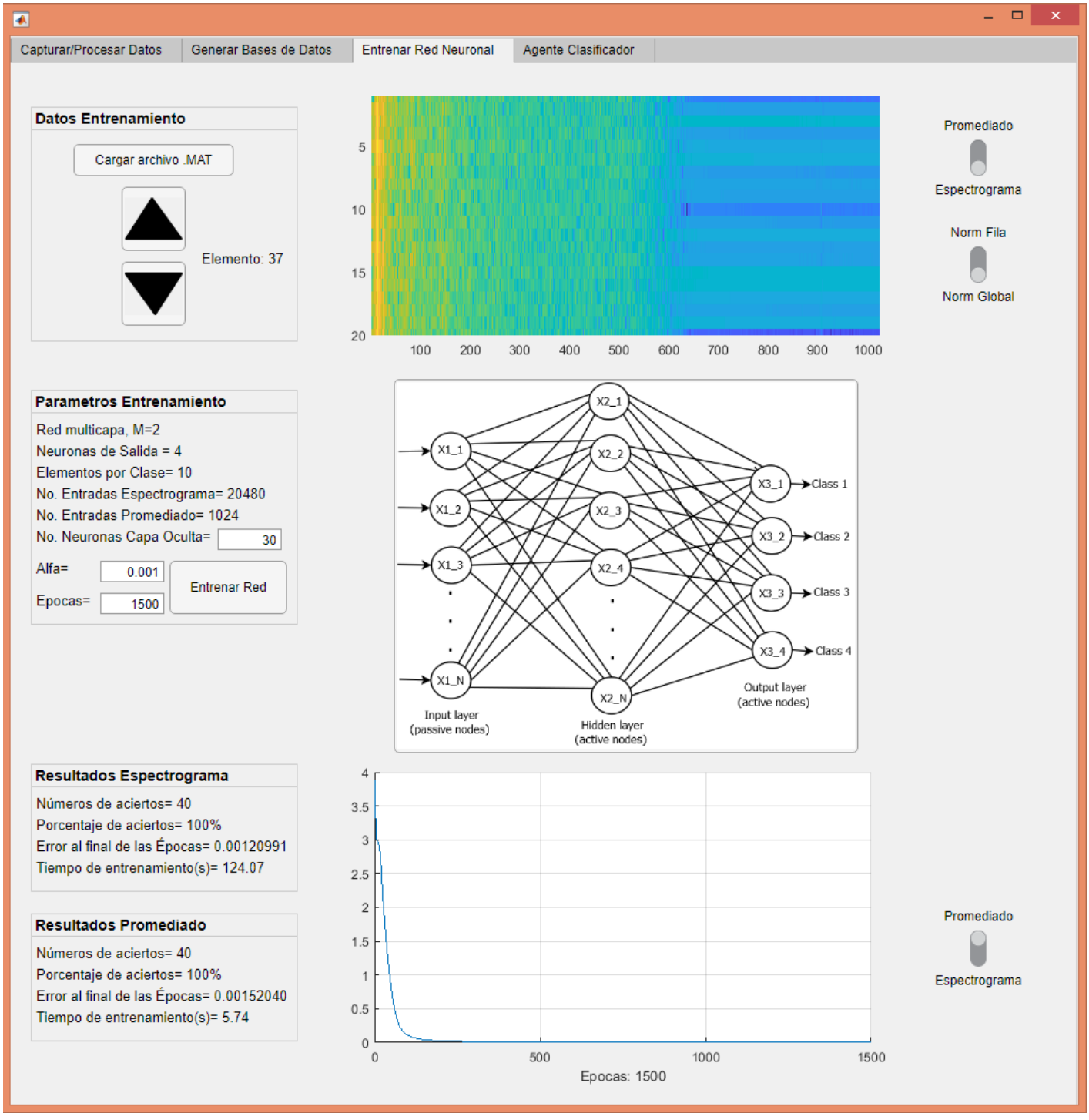


Figure 3.42: Training results for the ANN with real-world audios.

In figure 3.43, it can be seen the result of put under classification test a real-world acoustic signal of the submersible ROV, and the classification result of the agent was to point the Class 1 (the higher value of the last four neurons, marked with the green led) for both

inputs, the spectrogram and the mean of the spectrogram. In the next chapter in the section 4.2.2 can be found the summarized results for every kind of tested acoustic signature signal.

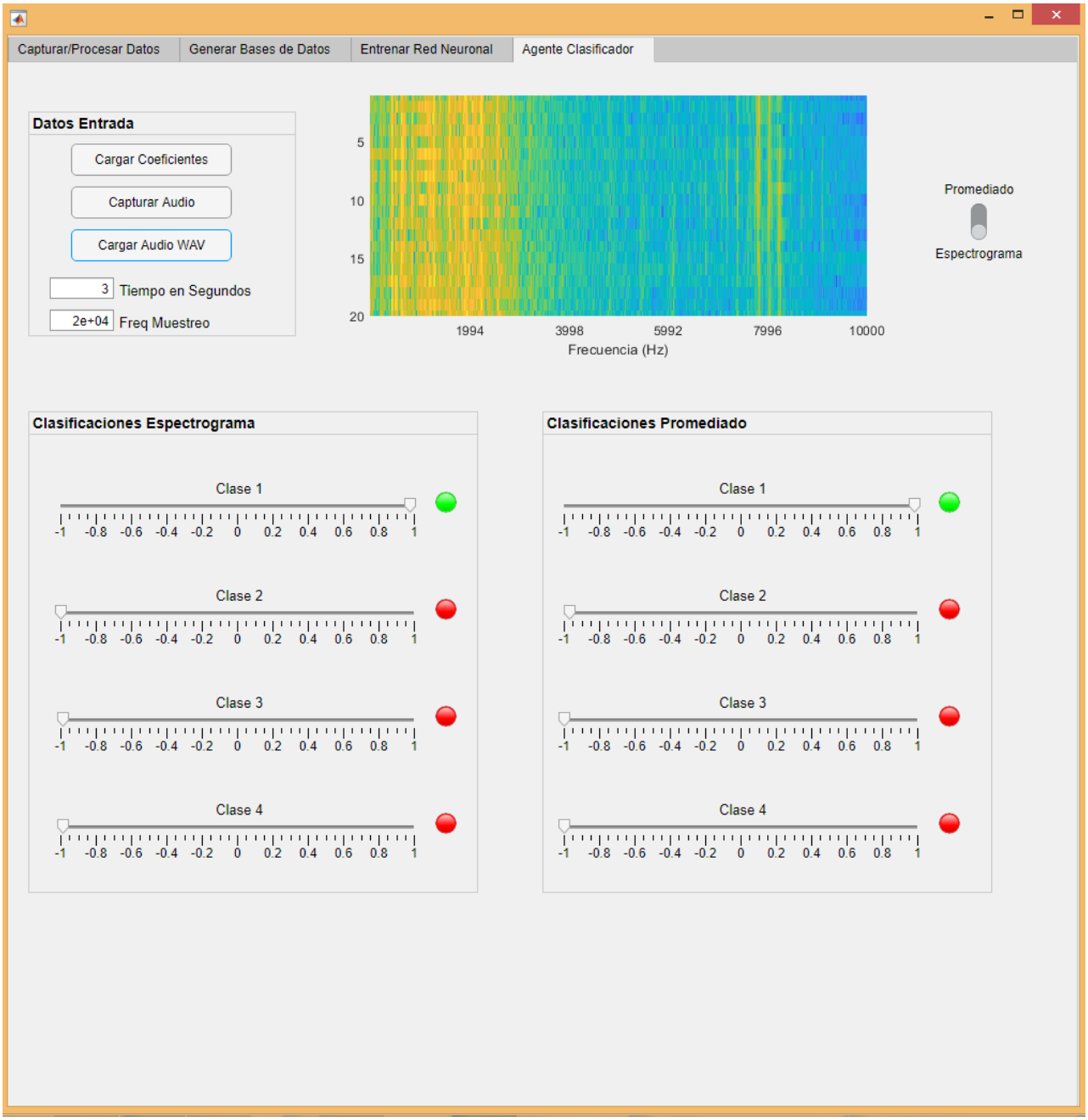


Figure 3.43: Testing the behavior of the Classifier Agent.

4 ANALYSIS AND RESULTS.

In this chapter, the results of the previous tests with the ANNs for both training and classifying (detailed in the sections 3.2.1 and 3.2.2) are presented and discussed, with the chief end to give the best possible information about how good or bad each of the implemented classificator is performing.

4.1 Performance results for the ANN with synthetic signals.

In the section 3.2.1 was described the proposed tests for the ANNs with synthetic signals. Those synthetic signals are a good way to verify the performance in a quick and controlled way (free of noise). In the following sections it will be summarized the results for both tests, the training and classification ones.

4.1.1 Training metrics

The training metrics used for this section will be the following: error curve and a summary table, for both spectrogram and mean of spectrogram for the four proposed topologies (30,100,200,500 neurons in the hidden layer) for the neural network. The error curve shows how the training is behaving while the epochs are passing. The gradient descend algorithm is on charge to make this error as minimum as possible.

As can be seen in the linear figures (for example in the figure 4.1), the error tends to drop suddenly in the first epochs and is hard to see the decline curve in the further epochs, for that reason was added a logarithmic ($\log_{10}$) plot in order to observe the decline of the curve in further epoch values.

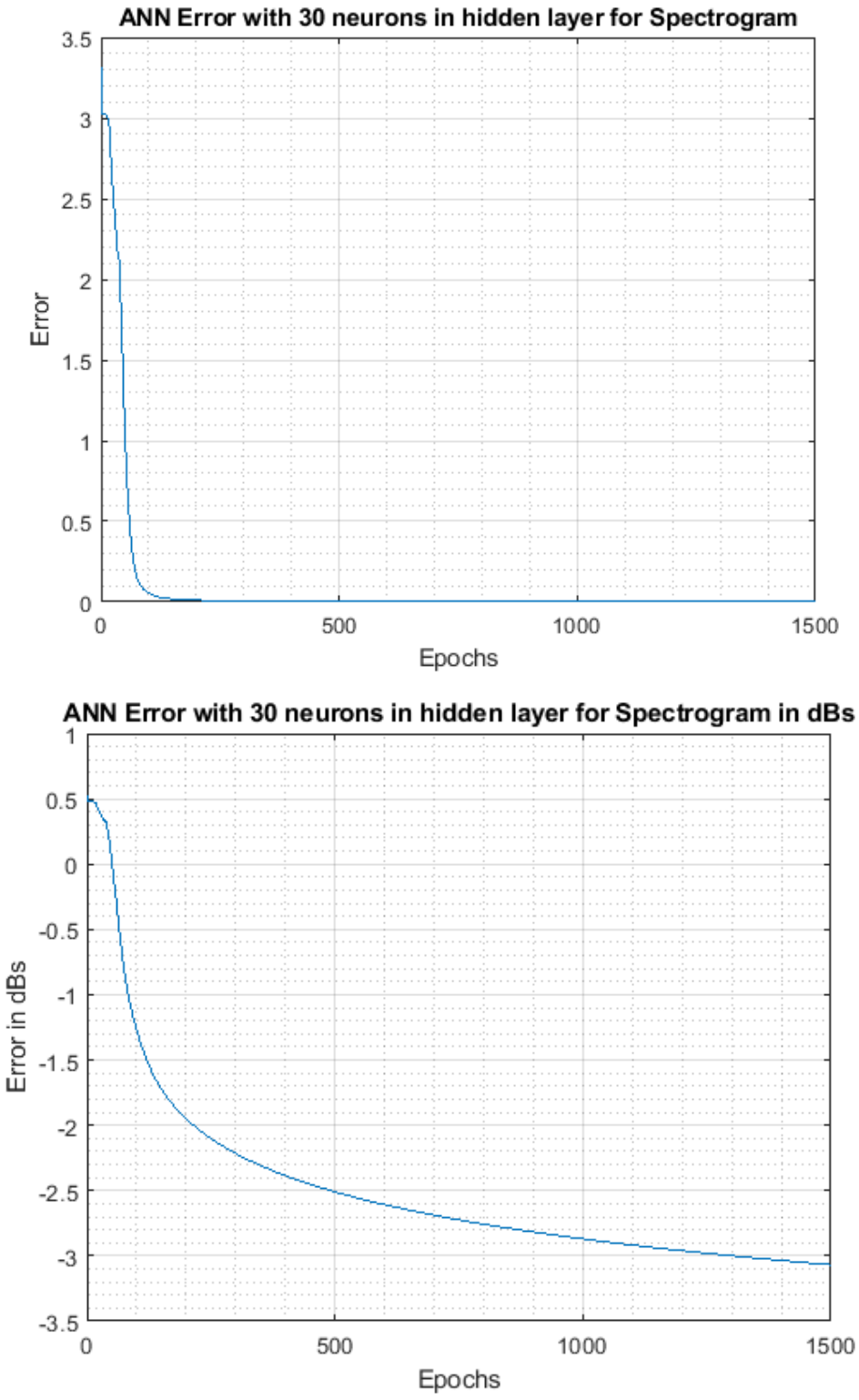


Figure 4.1: Error curves in lineal and logarithm scale for 30 neurons.

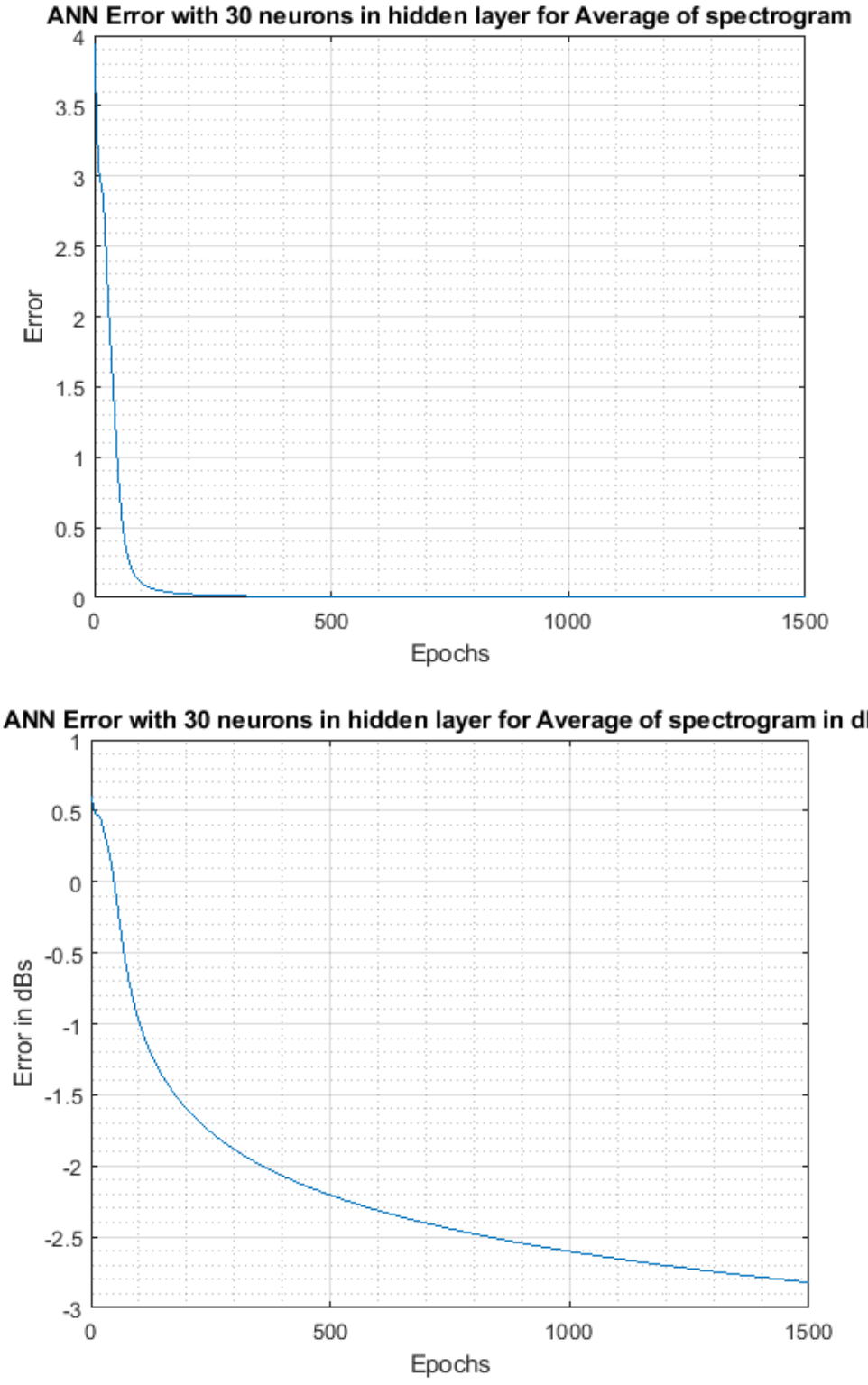


Figure 4.2: Error curves in lineal and logarithm scale for 30 neurons.

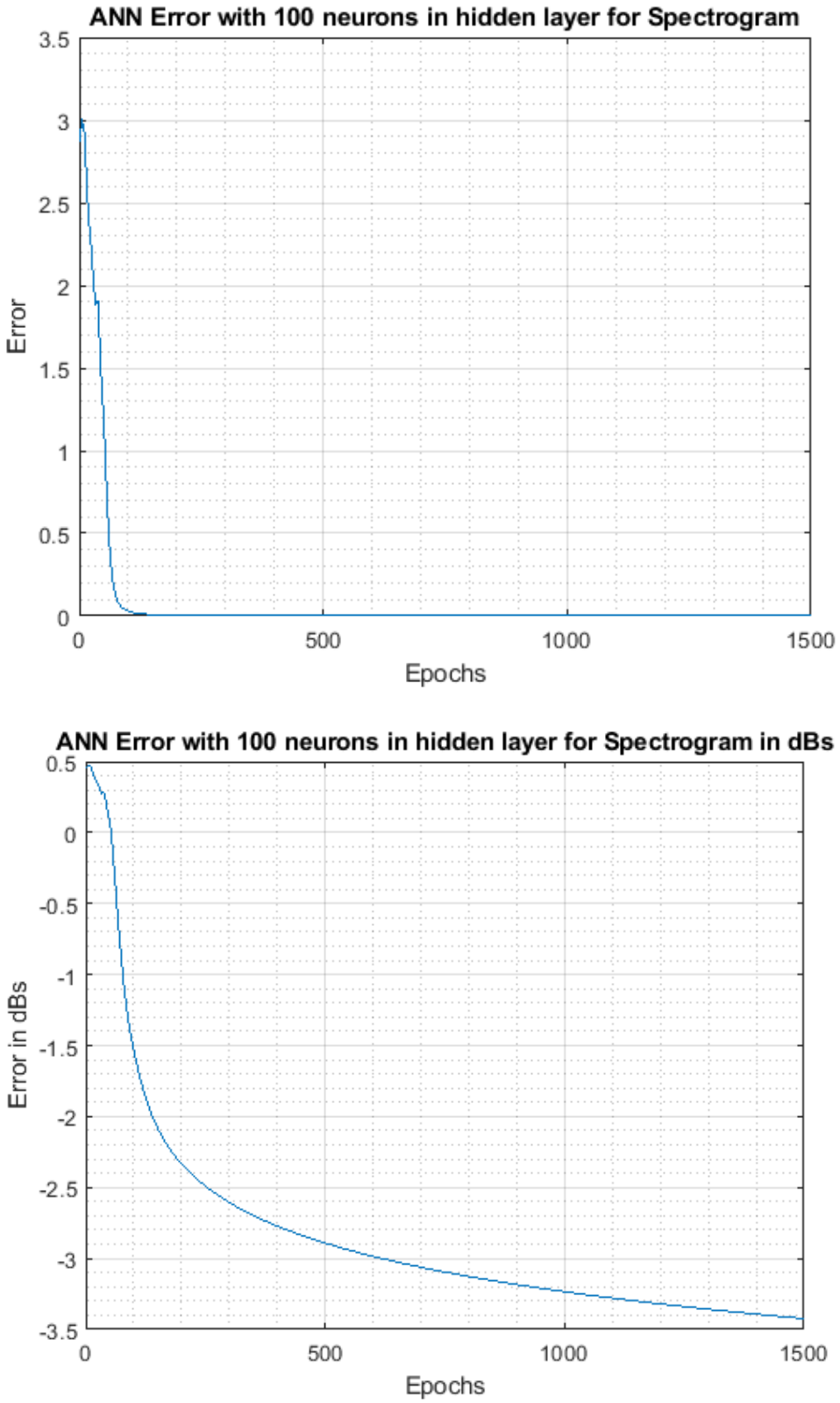
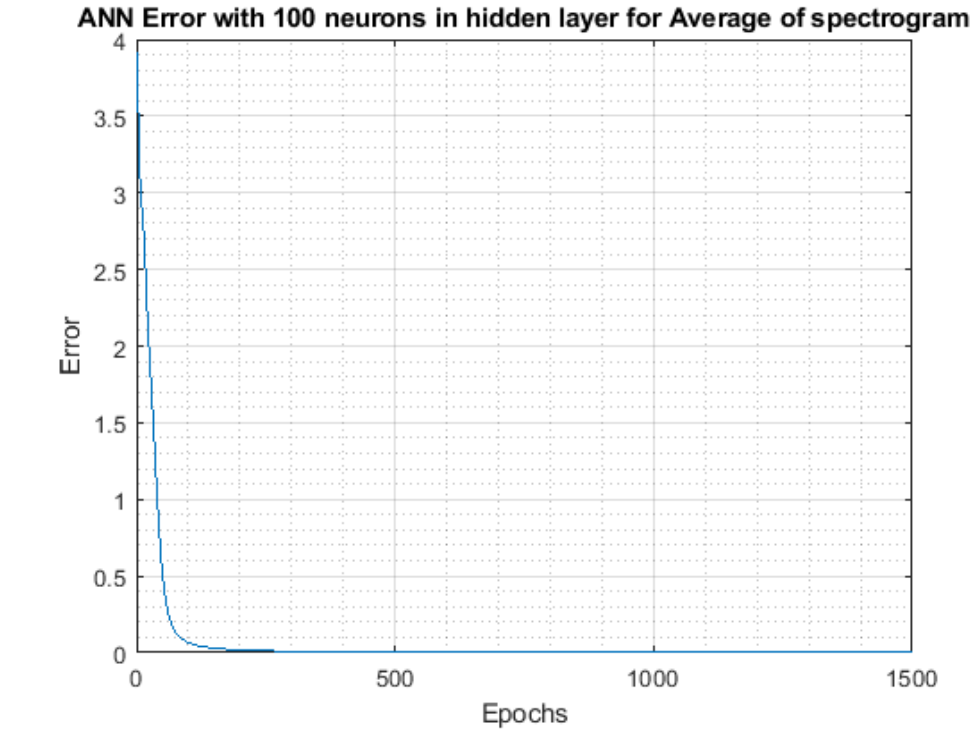


Figure 4.3: Error curves in lineal and logarithm scale for 100 neurons.



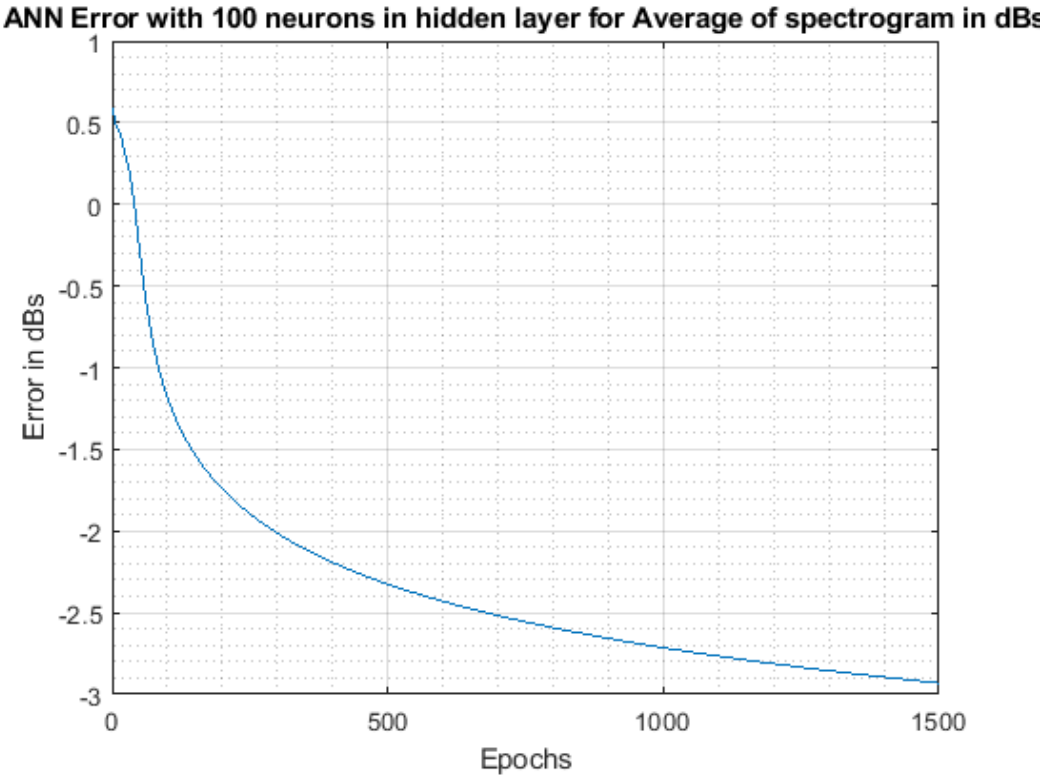


Figure 4.4: Error curves in lineal and logarithm scale for 100 neurons.

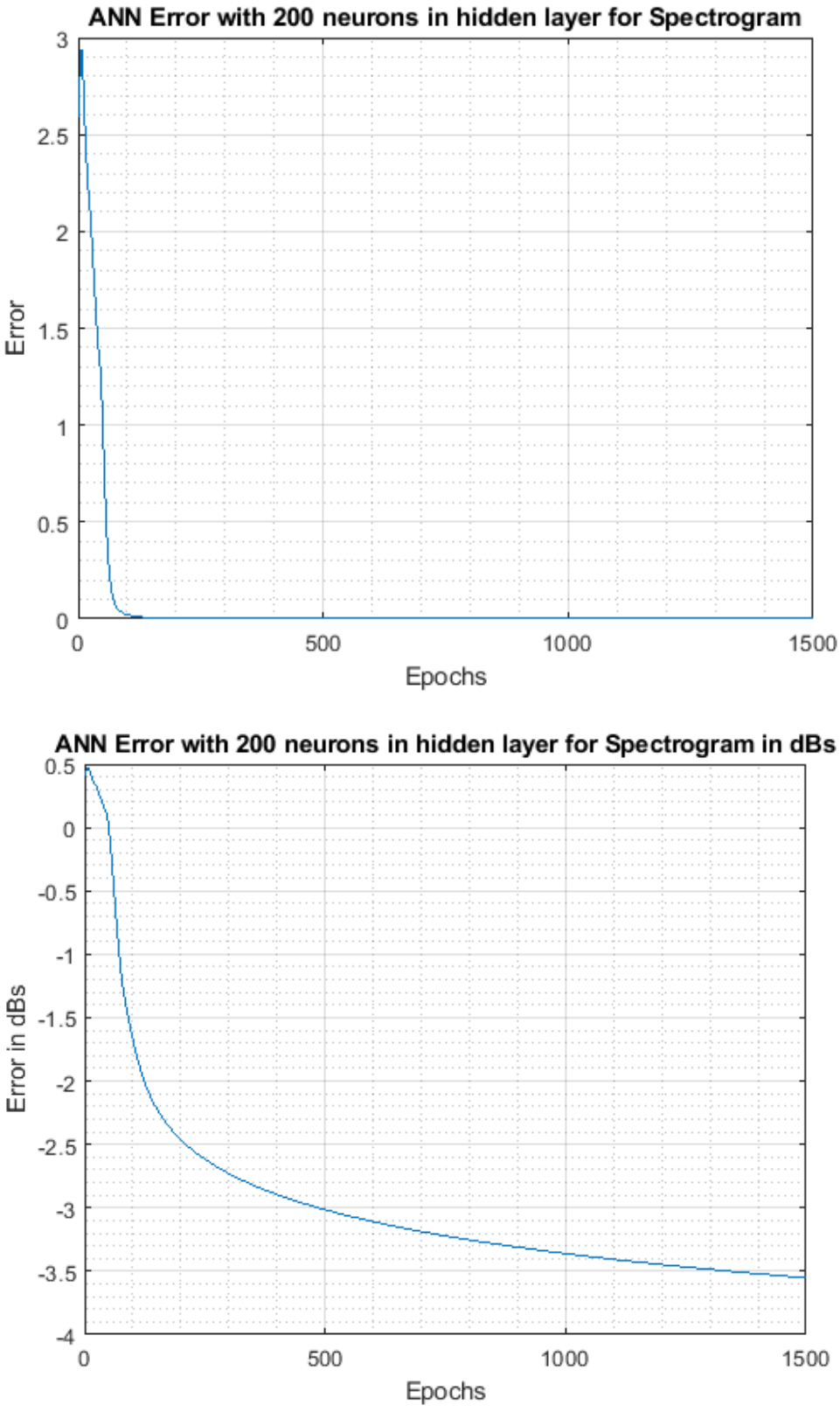


Figure 4.5: Error curves in lineal and logarithm scale for 200 neurons.

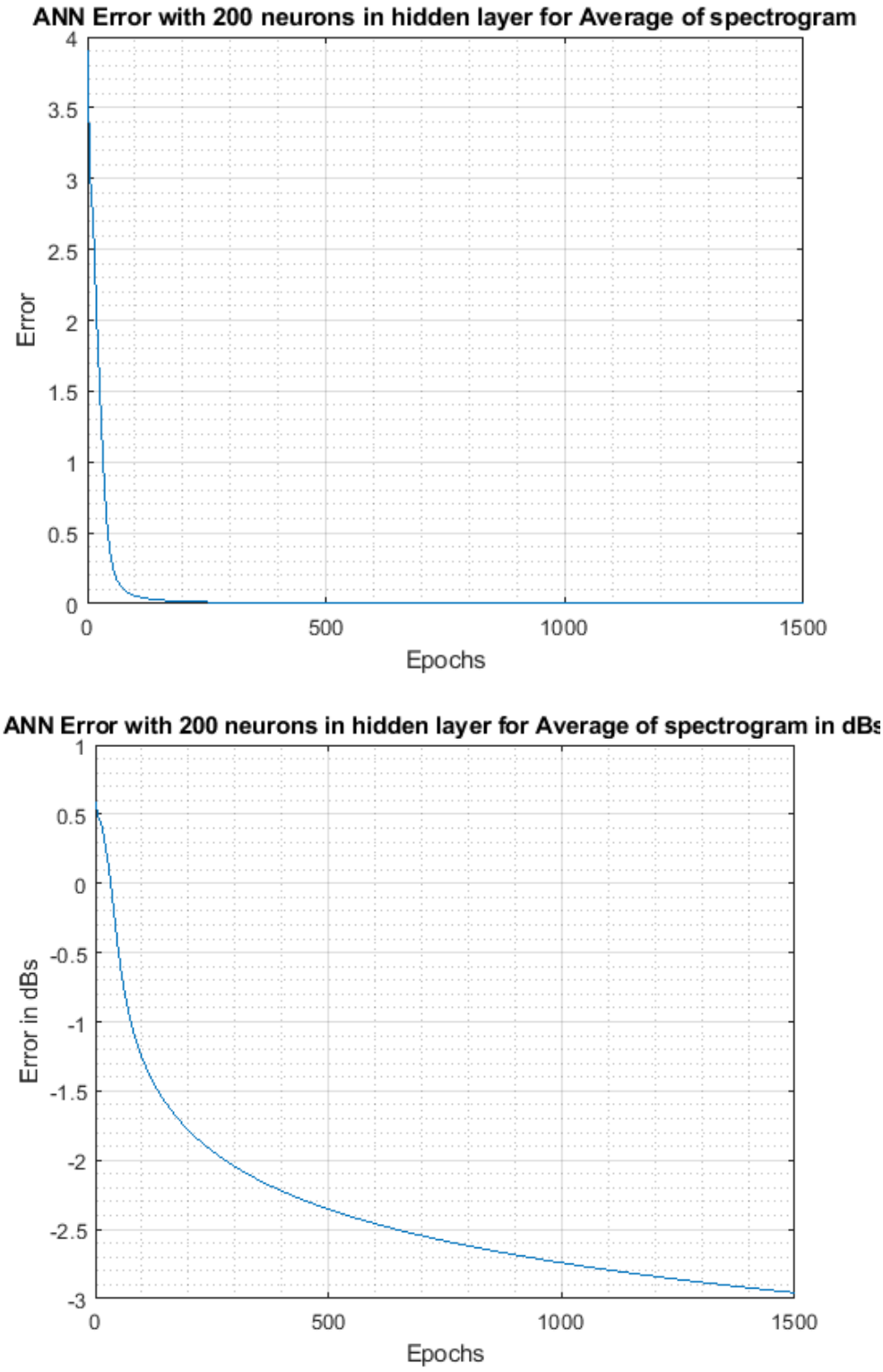


Figure 4.6: Error curves in lineal and logarithm scale for 200 neurons.

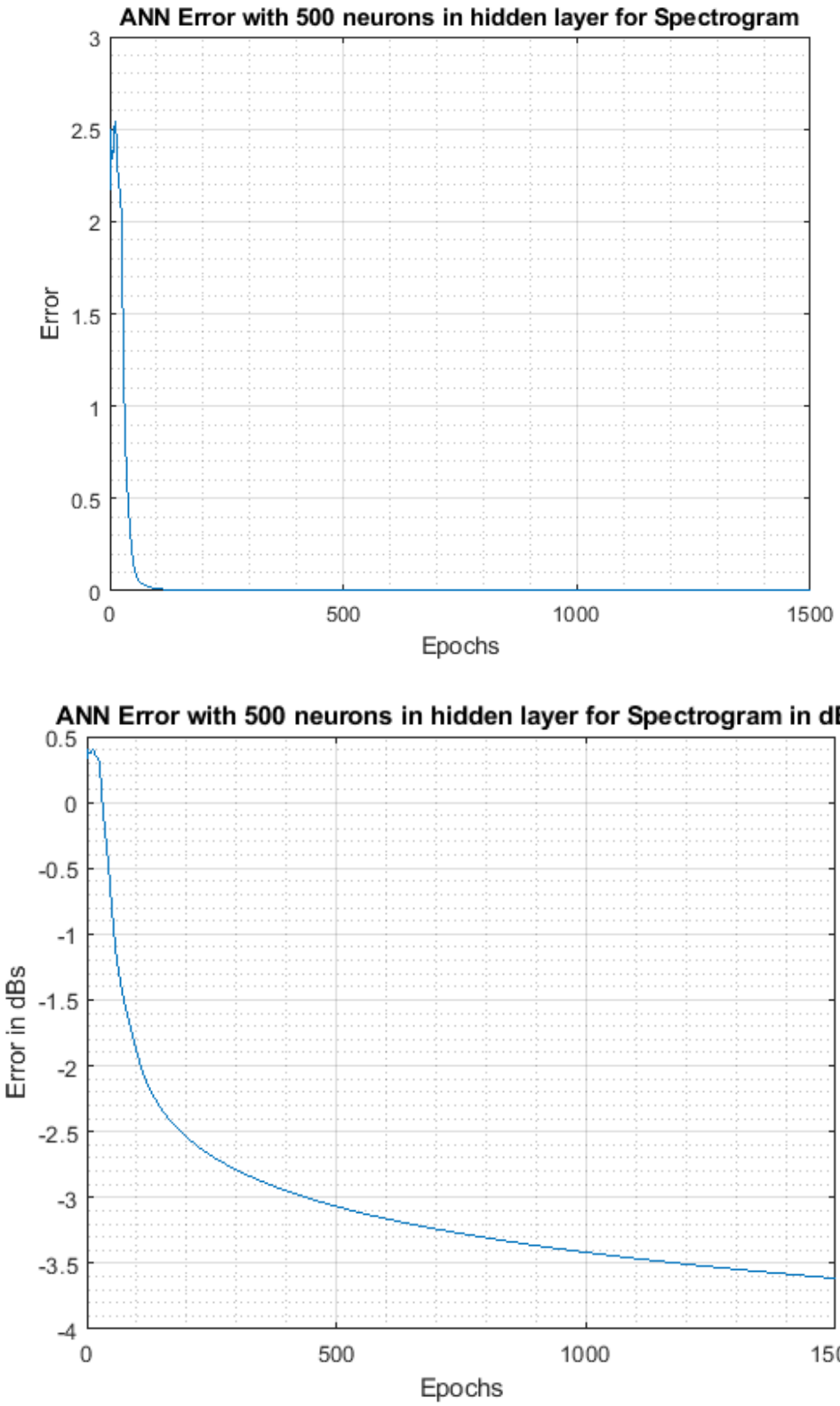


Figure 4.7: Error curves in lineal and logarithm scale for 500 neurons.

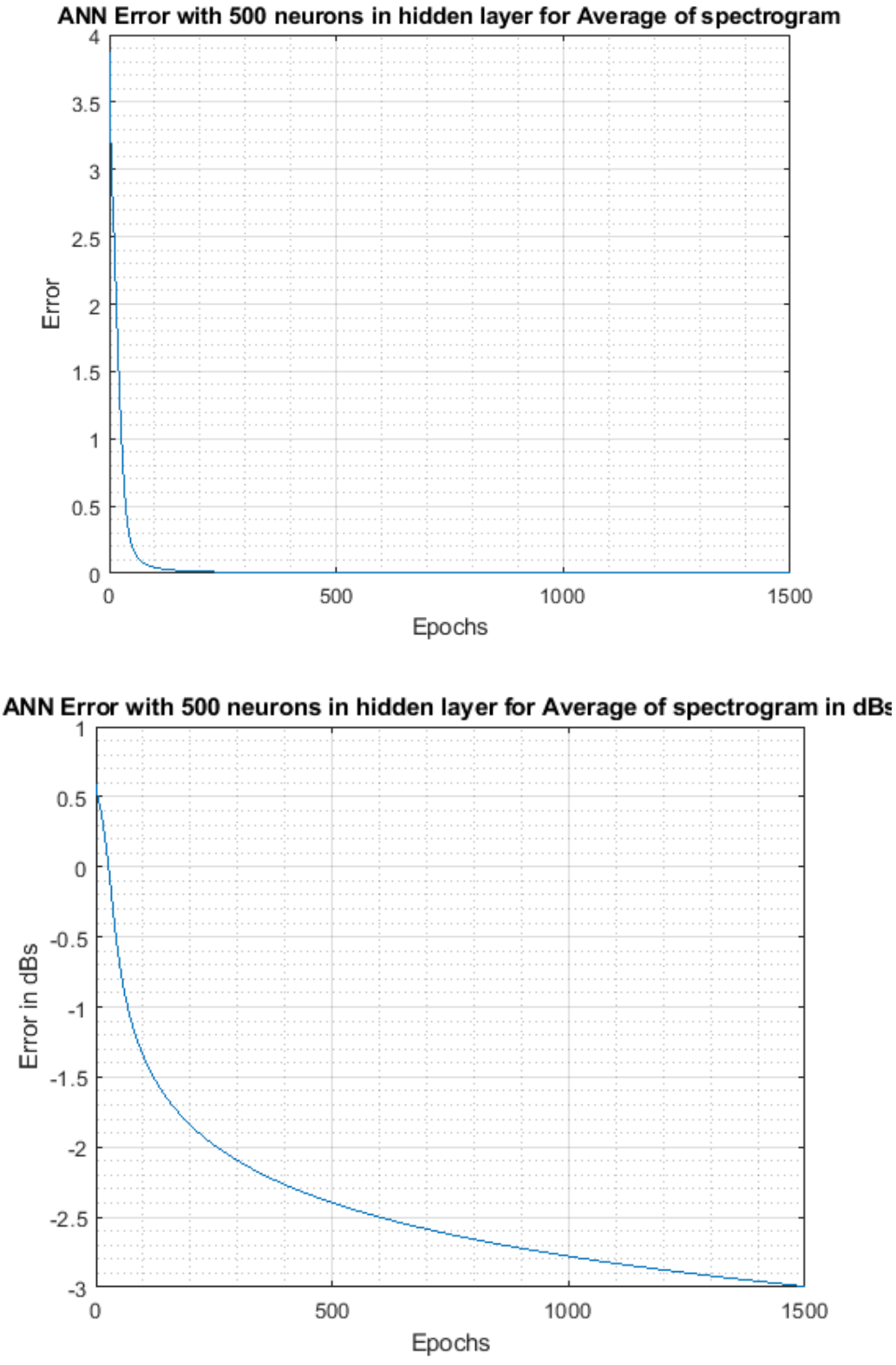


Figure 4.8: Error curves in lineal and logarithm scale for 500 neurons.

In the previous figures, was shown the retrieved error curves for the four types of topologies for the training of the ANN. In table 4.1, it can be found a summary of the

error behavior curves, that table is intended to give a glimpse of the behavior of the training process.

30 neurons in Hidden Layer	
Spectrogram	**Mean of spectrogram**
500 epochs: -2.5 dB	500 epochs: -2.2 dB
1500 epochs: -3.1 dB	1500 epochs: -2.8 dB

100 neurons in Hidden Layer	
Spectrogram	**Mean of spectrogram**
500 epochs: -2.9 dB	500 epochs: -2.3 dB
1500 epochs: -3.4 dB	1500 epochs: -2.9 dB

200 neurons in Hidden Layer	
Spectrogram	**Mean of spectrogram**
500 epochs: -3.0 dB	500 epochs: -2.4 dB
1500 epochs: -3.6 dB	1500 epochs: -3.0 dB

500 neurons in Hidden Layer	
Spectrogram	**Mean of spectrogram**
500 epochs: -3.1 dB	500 epochs: -2.4 dB
1500 epochs: -3.6 dB	1500 epochs: -3.0 dB

Table 4.1 Summary of the training error.

After the training process was done, the now trained neural network was tested with the same input data that was used to train the ANN, in order to verify that the found weights and bias coefficients of the ANN are working fine at least for the best scenario.

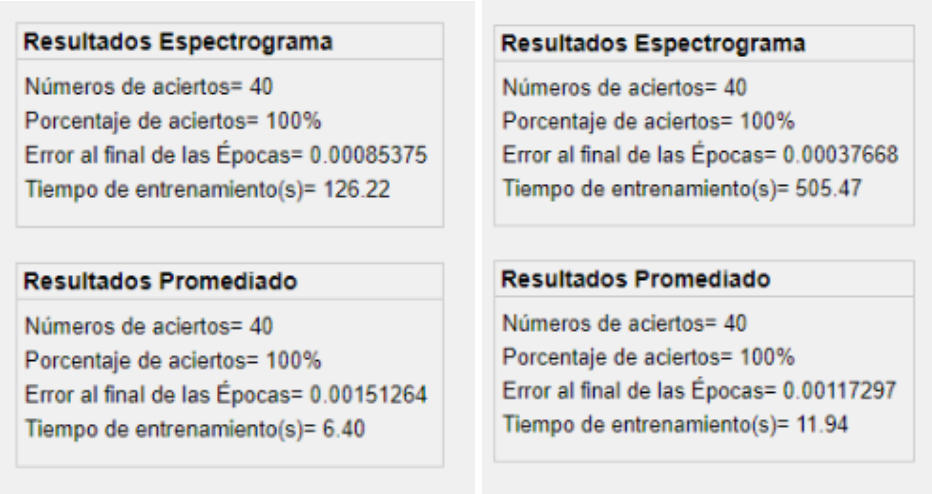


Figure 4.9: Summary with training information, for 30 and 100 neurons.

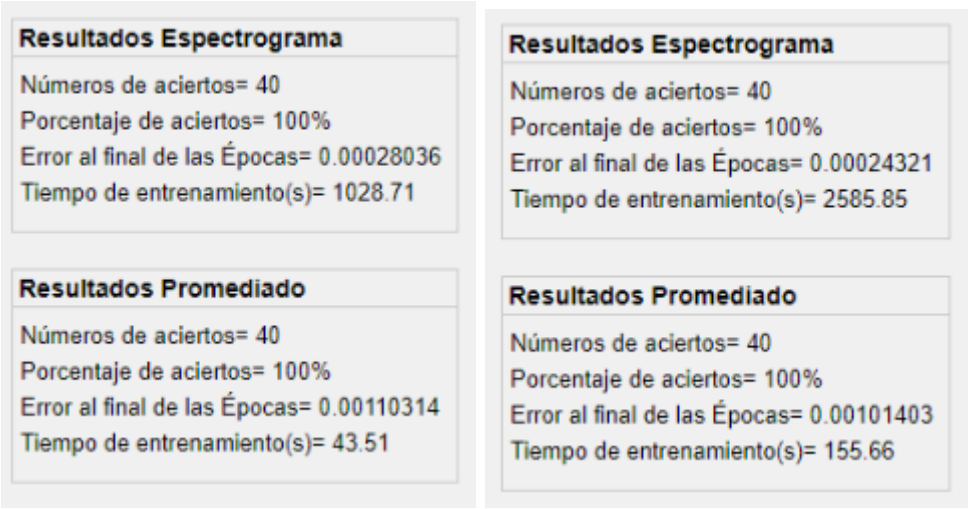


Figure 4.10: Summary with training information, for 200 and 500 neurons.

4.1.2 Classification metrics

In the field of machine learning and specifically for the problem of classification, a confusion matrix, also known as an error matrix[16], is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class. The name stems from the fact that it makes it easy to see if the system is confusing two classes (i.e. commonly mislabeling one as another). In table 4.6, it can be seen the results of the classification tests using a confusion matrix for the first topology of 30 neurons. It was used ten test signals per class (those signals are different than those that were used in the training process), so, for the class 1 was used an audio signal with a frequency of 775, 825, 875, 925, 975,1025, 1075, 1125, 1175, 1225 Hz, for the class 2 signals of 1775, 1825, 1875, 1925, 1975,2025, 2075, 2125, 2175, 1225 Hz, for the class 3 signals of 2775, 2825, 2875, 2925, 2975,3025, 3075, 3125, 3175, 3225 Hz and for the class 4 signals of 3775, 3825, 3875, 3925, 3975,4025, 4075, 4125, 4175, 4225 Hz. The same measurement process is applied to the ANNs topologies with 100, 200 and 500 neurons, which corresponds to topologies 2,3 and 4 respectively.

In the following tables 4.2, 4.3, 4.4 and 4.5, it can be seen the results of the output layers values of the ANNs for the classification tests previously described, all these tests are for the first topology consisting of 30 neurons in the hidden layer. The output layer neurons values are in a range of -1 to 1, where the unit value means a 100% of accuracy that the

audio signal belongs to this class, and a minus one means a 0% of accuracy that the audio signal belongs to this class.

Audio Signals (Hz)	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
775	0.98	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.99
825	0.98	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.99
875	0.98	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.99
925	0.98	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.99
975	0.98	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.97
1025	0.98	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.97
1075	0.97	-0.98	-0.99	-0.99	0.93	-0.99	-0.99	-0.91
1125	0.97	-0.98	-0.99	-0.99	0.96	-0.98	-0.99	-0.97
1175	0.97	-0.98	-0.99	-0.99	0.95	-0.98	-0.99	-0.98
1225	0.97	-0.97	-0.98	-0.99	0.95	-0.94	-0.99	-0.99

Table 4.2 ANN Topology 1 (30 neurons), for class 1 input signals.

Audio Signals (Hz)	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
1775	-0.98	0.98	-0.99	-0.99	-0.97	0.95	-0.99	-0.99
1825	-0.98	0.98	-0.99	-0.99	-0.97	0.94	-0.99	-0.99
1875	-0.98	0.99	-0.99	-0.99	-0.98	0.98	-0.99	-0.99
1925	-0.98	0.98	-0.99	-0.98	-0.98	0.95	-0.99	-0.99
1975	-0.98	0.98	-0.99	-0.99	-0.98	0.97	-0.99	-0.99
2025	-0.98	0.97	-0.99	-0.99	-0.98	0.97	-0.99	-0.99
2075	-0.99	0.95	-0.99	-0.97	-0.99	0.99	-0.99	-0.99
2125	-0.99	0.96	-0.96	-0.99	-0.99	0.98	-0.98	-0.99
2175	-0.99	0.95	-0.97	-0.99	-0.99	0.97	-0.98	-0.99
2225	-0.99	0.93	-0.93	-0.99	-0.99	0.84	-0.97	-0.99

Table 4.3 ANN Topology 1 (30 neurons), for class 2 input signals.

Audio Signals (Hz)	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
2775	-0.98	-0.98	0.96	-0.99	-0.99	-0.98	0.95	-0.99
2825	-0.99	-0.99	0.99	-0.99	-0.99	-0.98	0.97	-0.99
2875	-0.99	-0.99	0.99	-0.99	-0.99	-0.99	0.98	-0.98
2925	-0.99	-0.99	0.99	-0.99	-0.99	-0.99	0.97	-0.99
2975	-0.99	-0.99	0.98	-0.99	-0.99	-0.98	0.98	-0.99
3025	-0.99	-0.99	0.98	-0.99	-0.99	-0.99	0.97	-0.97
3075	-0.99	-0.99	0.96	-0.98	-0.99	-0.99	0.93	-0.95
3125	-0.98	-0.99	0.96	-0.98	-0.99	-0.99	0.97	-0.98
3175	-0.99	-0.99	0.95	-0.98	-0.99	-0.99	0.91	-0.92
3225	-0.99	-0.99	0.89	-0.96	-0.99	-0.99	0.92	-0.92

Table 4.4 ANN Topology 1 (30 neurons), for class 3 input signals.

Audio Signals (Hz)	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
3775	-0.99	-0.99	-0.98	0.98	-0.99	-0.99	-0.98	0.98
3825	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.98	0.98
3875	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.98	0.98
3925	-0.98	-0.99	-0.99	0.98	-0.99	-0.99	-0.98	0.99
3975	-0.96	-0.99	-0.99	0.98	-0.97	-0.99	-0.99	0.98
4025	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.99	0.99
4075	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.98	0.99
4125	-0.99	-0.99	-0.99	0.99	-0.99	-0.99	-0.99	0.99
4175	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.98	0.99
4225	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.99	0.99

Table 4.5 ANN Topology 1 (30 neurons), for class 4 input signals.

With the information of the previous four tables it is possible to build the confusion matrix for the first topology, that contains basically a hidden layer of 30 neurons for the ANN. In the table 4.6 can be seen two confusion matrices, the first one is for the Spectrogram

ANN and the other for the mean of the Spectrogram ANN, the four classes in the tables are marked with the signal frequencies of the class under test.

		Spectrogram Actual Class				
		1 Khz	2 Khz	3 Khz	4 Khz	**Effectiveness**
Predicted Class	1 Khz	10	0	0	0	100%
	2 Khz	0	10	0	0	100%
	3 Khz	0	0	10	0	100%
	4 Khz	0	0	0	10	100%
		Total Effectiveness = 100%				

		Mean of Spectrogram Actual Class				
		1 Khz	2 Khz	3 Khz	4 Khz	**Effectiveness**
Predicted Class	1 Khz	10	0	0	0	100%
	2 Khz	0	10	0	0	100%
	3 Khz	0	0	10	0	100%
	4 Khz	0	0	0	10	100%
		Total Effectiveness = 100%				

Table 4.6 Confusion matrix for topology 1 (30 neurons).

The results of the table 4.6 shows that for this first ANN topology the total classification effectiveness was of 100% for the proposed input audio signals. For the case of testing another frequencies (signals not included in the training dataset) there are possibilities to get correct classifications results depending on how similar is that signal with any of the training dataset.

Now that the testing of the first topology for the synthetic signals is done, it is time to test the second ANN topology, in the following tables 4.7, 4.8, 4.9 and 4.10, can be seen the results of the output layers values for the classification tests, all these tests are for the second topology consisting of 100 neurons in the hidden layer. The output layer neurons values are in a range of -1 to 1, where the unit value means a 100% of accuracy that the audio signal belongs to this class, and a negative one means a 0% of accuracy that the audio signal belongs to this class.

Audio Signals (Hz)	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
775	0.99	-0.99	-0.99	-0.99	0.98	-0.98	-0.99	-0.97
825	0.99	-0.99	-0.98	-0.99	0.99	-0.99	-0.99	-0.99
875	0.99	-0.99	-0.98	-0.99	0.99	-0.99	-0.99	-0.97
925	0.98	-0.98	-0.99	-0.98	0.97	-0.99	-0.99	-0.97
975	0.98	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.97
1025	0.97	-0.99	-0.99	-0.97	0.85	-0.99	-0.99	-0.64
1075	0.97	-0.98	-0.99	-0.98	0.95	-0.99	-0.99	-0.94
1125	0.98	-0.98	-0.99	-0.99	0.96	-0.98	-0.99	-0.98
1175	0.96	-0.97	-0.99	-0.99	0.96	-0.97	-0.99	-0.99
1225	0.97	-0.96	-0.99	-0.99	0.95	-0.96	-0.99	-0.99

Table 4.7 ANN Topology 2(100 neurons), for class 1 input signals.

Audio Signals (Hz)	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
1775	-0.97	0.98	-0.99	-0.99	-0.95	0.96	-0.99	-0.99
1825	-0.97	0.98	-0.99	-0.99	-0.96	0.95	-0.99	-0.99
1875	-0.99	0.99	-0.99	-0.99	-0.99	0.99	-0.99	-0.99
1925	-0.98	0.95	-0.99	-0.97	-0.98	0.97	-0.99	-0.99
1975	-0.98	0.98	-0.99	-0.99	-0.99	0.98	-0.99	-0.99
2025	-0.99	0.98	-0.99	-0.98	-0.99	0.98	-0.99	-0.99
2075	-0.99	0.98	-0.98	-0.99	-0.99	0.98	-0.98	-0.99
2125	-0.99	0.91	-0.97	-0.99	-0.99	0.96	-0.99	-0.99
2175	-0.99	0.96	-0.95	-0.99	-0.99	0.97	-0.97	-0.99
2225	-0.99	0.94	-0.97	-0.99	-0.99	0.92	-0.95	-0.99

Table 4.8 ANN Topology 2(100 neurons), for class 2 input signals.

Audio Signals (Hz)	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
2775	-0.98	-0.98	0.97	-0.99	-0.99	-0.97	0.96	-0.99
2825	-0.99	-0.99	0.99	-0.99	-0.99	-0.98	0.97	-0.99
2875	-0.99	-0.99	0.97	-0.99	-0.99	-0.99	0.98	-0.99
2925	-0.98	-0.99	0.99	-0.99	-0.99	-0.99	0.98	-0.99
2975	-0.99	-0.99	0.96	-0.99	-0.99	-0.99	0.99	-0.99
3025	-0.99	-0.99	0.98	-0.99	-0.99	-0.99	0.98	-0.98
3075	-0.98	-0.99	0.95	-0.99	-0.99	-0.99	0.97	-0.97
3125	-0.98	-0.99	0.95	-0.99	-0.99	-0.99	0.85	-0.98
3175	-0.98	-0.99	0.94	-0.98	-0.99	-0.99	0.93	-0.92
3225	-0.99	-0.99	0.88	-0.95	-0.99	-0.99	0.94	-0.93

Table 4.9 ANN Topology 2(100 neurons), for class 3 input signals.

Audio Signals (Hz)	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
3775	-0.99	-0.99	-0.98	0.98	-0.99	-0.99	-0.98	0.99
3825	-0.99	-0.99	-0.98	0.98	-0.99	-0.99	-0.98	0.99
3875	-0.99	-0.99	-0.99	0.99	-0.99	-0.99	-0.98	0.99
3925	-0.98	-0.99	-0.99	0.98	-0.98	-0.99	-0.99	0.99
3975	-0.95	-0.99	-0.99	0.98	-0.97	-0.99	-0.99	0.99
4025	-0.99	-0.99	-0.99	0.99	-0.99	-0.99	-0.99	0.99
4075	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.99	0.99
4125	-0.99	-0.99	-0.99	0.99	-0.99	-0.99	-0.99	0.99
4175	-0.99	-0.99	-0.99	0.99	-0.99	-0.99	-0.99	0.99
4225	-0.99	-0.99	-0.99	0.99	-0.99	-0.99	-0.99	0.99

Table 4.10 ANN Topology 2(100 neurons), for class 4 input signals.

With the information of the previous four tables it is possible to build the confusion matrix for the second topology, that contains basically a hidden layer of 100 neurons for the ANN. In the table 4.11 can be seen two confusion matrices, the first one is for the

Spectrogram ANN and the other for the mean of the Spectrogram ANN, the four classes in the tables are marked with the signal frequencies of the class under test.

		Spectrogram Actual Class				
		1 Khz	2 Khz	3 Khz	4 Khz	**Effectiveness**
Predicted Class	1 Khz	10	0	0	0	100%
	2 Khz	0	10	0	0	100%
	3 Khz	0	0	10	0	100%
	4 Khz	0	0	0	10	100%
		Total Effectiveness = 100%				

		Mean of Spectrogram Actual Class				
		1 Khz	2 Khz	3 Khz	4 Khz	**Effectiveness**
Predicted Class	1 Khz	10	0	0	0	100%
	2 Khz	0	10	0	0	100%
	3 Khz	0	0	10	0	100%
	4 Khz	0	0	0	10	100%
		Total Effectiveness = 100%				

Table 4.11 Confusion matrix for topology 2 (100 neurons).

The results of the table 4.11, shows that for this second ANN topology the total classification effectiveness was of 100% for the proposed input audio signals.

Now that the testing of the second topology for the synthetic signals is done, it is time to test the third ANN topology, in the following tables 4.12, 4.13, 4.14 and 4.15, can be seen the results of the output layers values for the classification tests, all these tests are for the third topology consisting of 200 neurons in the hidden layer. The output layer neurons values are in a range of -1 to 1, where the unit value means a 100% of accuracy that the audio signal belongs to this class, and a negative one means a 0% of accuracy that the audio signal belongs to this class.

Audio Signals (Hz)	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
775	0.98	-0.99	-0.99	-0.99	0.99	-0.99	-0.99	-0.98
825	0.99	-0.99	-0.98	-0.99	0.99	-0.99	-0.99	-0.99
875	0.99	-0.99	-0.97	-0.99	0.99	-0.99	-0.99	-0.99
925	0.98	-0.99	-0.99	-0.99	0.99	-0.99	-0.99	-0.98
975	0.98	-0.99	-0.99	-0.99	0.97	-0.99	-0.99	-0.96
1025	0.98	-0.99	-0.99	-0.98	0.88	-0.99	-0.99	-0.65
1075	0.98	-0.99	-0.99	-0.98	0.95	-0.99	-0.99	-0.94
1125	0.97	-0.97	-0.99	-0.99	0.97	-0.98	-0.99	-0.98
1175	0.97	-0.98	-0.99	-0.99	0.96	-0.97	-0.99	-0.99
1225	0.98	-0.95	-0.99	-0.99	0.96	-0.95	-0.99	-0.99

Table 4.12 ANN Topology 3 (200 neurons), for class 1 input signals.

Audio Signals (Hz)	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
1775	-0.98	0.93	-0.99	-0.98	-0.96	0.95	-0.99	-0.99
1825	-0.98	0.99	-0.99	-0.99	-0.98	0.93	-0.99	-0.99
1875	-0.99	0.98	-0.99	-0.99	-0.99	0.99	-0.99	-0.99
1925	-0.99	0.99	-0.99	-0.98	-0.98	0.95	-0.99	-0.99
1975	-0.99	0.98	-0.99	-0.99	-0.99	0.98	-0.99	-0.99
2025	-0.99	0.99	-0.99	-0.99	-0.99	0.98	-0.99	-0.99
2075	-0.99	0.99	-0.99	-0.99	-0.99	0.98	-0.99	-0.99
2125	-0.99	0.98	-0.95	-0.99	-0.99	0.98	-0.99	-0.99
2175	-0.99	0.97	-0.95	-0.99	-0.99	0.95	-0.97	-0.99
2225	-0.99	0.79	-0.96	-0.99	-0.99	0.90	-0.95	-0.99

Table 4.13 ANN Topology 3 (200 neurons), for class 2 input signals.

Audio Signals (Hz)	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
2775	-0.98	-0.99	0.97	-0.99	-0.99	-0.98	0.95	-0.99
2825	-0.98	-0.99	0.99	-0.99	-0.99	-0.98	0.98	-0.99
2875	-0.99	-0.99	0.99	-0.99	-0.99	-0.99	0.98	-0.99
2925	-0.98	-0.99	0.99	-0.99	-0.99	-0.99	0.97	-0.99
2975	-0.99	-0.99	0.98	-0.99	-0.99	-0.99	0.98	-0.99
3025	-0.99	-0.99	0.98	-0.99	-0.99	-0.99	0.98	-0.98
3075	-0.98	-0.99	0.96	-0.99	-0.99	-0.99	0.90	-0.95
3125	-0.99	-0.99	0.98	-0.98	-0.99	-0.99	0.98	-0.80
3175	-0.99	-0.99	0.94	-0.98	-0.99	-0.99	0.91	-0.92
3225	-0.99	-0.99	0.95	-0.98	-0.99	-0.99	0.90	-0.90

Table 4.14 ANN Topology 3 (200 neurons), for class 3 input signals.

Audio Signals (Hz)	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
3775	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.97	0.98
3825	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.98	0.98
3875	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.98	0.98
3925	-0.98	-0.99	-0.99	0.98	-0.99	-0.99	-0.99	0.99
3975	-0.95	-0.99	-0.99	0.98	-0.96	-0.99	-0.99	0.99
4025	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.99	0.99
4075	-0.99	-0.99	-0.99	0.99	-0.99	-0.99	-0.99	0.99
4125	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.99	0.99
4175	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.99	0.99
4225	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.99	0.99

Table 4.15 ANN Topology 3 (200 neurons), for class 4 input signals.

With the information of the previous four tables it is possible to build the confusion matrix for the third topology, that contains basically a hidden layer of 200 neurons for the ANN. In the table 4.16 can be seen two confusion matrices, the first one is for the Spectrogram

ANN and the other for the mean of the Spectrogram ANN, the four classes in the tables are marked with the signal frequencies of the class under test.

		Spectrogram Actual Class				
		1 Khz	2 Khz	3 Khz	4 Khz	**Effectiveness**
Predicted Class	1 Khz	10	0	0	0	100%
	2 Khz	0	10	0	0	100%
	3 Khz	0	0	10	0	100%
	4 Khz	0	0	0	10	100%
		Total Effectiveness = 100%				

		Mean of Spectrogram Actual Class				
		1 Khz	2 Khz	3 Khz	4 Khz	**Effectiveness**
Predicted Class	1 Khz	10	0	0	0	100%
	2 Khz	0	10	0	0	100%
	3 Khz	0	0	10	0	100%
	4 Khz	0	0	0	10	100%
		Total Effectiveness = 100%				

Table 4.16 Confusion matrix for topology 3 (200 neurons).

The results of the table 4.16 shows that for this third ANN topology the total classification effectiveness was of 100% for the proposed input audio signals.

Now that the testing of the third topology for the synthetic signals is done, it is time to test the fourth ANN topology, in the following tables 4.17, 4.18, 4.19 and 4.20, can be seen the results of the output layers values for the classification tests, all these tests are for the fourth topology consisting of 500 neurons in the hidden layer. The output layer neurons values are in a range of -1 to 1, where the unit value means a 100% of accuracy that the audio signal belongs to this class, and a negative one means a 0% of accuracy that the audio signal belongs to this class.

Audio Signals (Hz)	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
775	0.99	-0.99	-0.99	-0.99	.099	-0.99	-0.99	-0.98
825	0.99	-0.99	-0.99	-0.99	0.99	-0.99	-0.99	-0.99
875	0.99	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.98
925	0.99	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.99
975	0.99	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.98
1025	0.97	-0.99	-0.99	-0.98	0.80	-0.99	-0.99	-0.60
1075	0.98	-0.99	-0.99	-0.99	0.95	-0.99	-0.99	-0.93
1125	0.98	-0.99	-0.99	-0.99	0.96	-0.98	-0.99	-0.98
1175	0.98	-0.95	-0.99	-0.99	0.98	-0.96	-0.99	-0.99
1225	0.96	-0.95	-0.99	-0.99	0.95	-0.97	-0.99	-0.99

Table 4.17 ANN Topology 4 (500 neurons), for class 1 input signals.

Audio Signals (Hz)	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
1775	-0.97	0.98	-0.99	-0.99	-0.97	0.96	-0.99	-0.99
1825	-0.96	0.98	-0.99	-0.99	-0.97	0.96	-0.99	-0.99
1875	-0.97	0.99	-0.99	-0.99	-0.98	0.95	-0.99	-0.99
1925	-0.99	0.96	-0.99	-0.99	-0.98	0.97	-0.99	-0.99
1975	-0.99	0.98	-0.99	-0.99	-0.99	0.97	-0.99	-0.99
2025	-0.99	0.99	-0.99	-0.99	-0.99	0.98	-0.99	-0.99
2075	-0.99	0.99	-0.99	-0.99	-0.99	0.99	-0.99	-0.99
2125	-0.99	0.97	-0.97	-0.99	-0.99	0.98	-0.99	-0.99
2175	-0.99	0.97	-0.99	-0.99	-0.99	0.96	-0.98	-0.99
2225	-0.99	0.95	-0.95	-0.99	-0.99	0.90	-0.95	-0.99

Table 4.18 ANN Topology 4 (500 neurons), for class 2 input signals.

Audio Signals (Hz)	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
2775	-0.98	-0.99	0.95	-0.99	-0.99	-0.98	0.94	-0.99
2825	-0.99	-0.99	0.99	-0.99	-0.99	-0.98	0.98	-0.99
2875	-0.99	-0.99	0.98	-0.99	-0.99	-0.99	0.99	-0.99
2925	-0.99	-0.99	0.99	-0.99	-0.99	-0.99	0.99	-0.99
2975	-0.99	-0.99	0.99	-0.99	-0.99	-0.99	0.99	-0.99
3025	-0.99	-0.99	0.99	-0.99	-0.99	-0.99	0.99	-0.98
3075	-0.99	-0.99	0.97	-0.99	-0.99	-0.99	0.97	-0.97
3125	-0.99	-0.99	0.97	-0.99	-0.99	-0.99	0.96	-0.98
3175	-0.99	-0.99	0.79	-0.95	-0.99	-0.99	0.87	-0.92
3225	-0.99	-0.99	0.72	-0.92	-0.99	-0.99	0.90	-0.90

Table 4.19 ANN Topology 4 (500 neurons), for class 3 input signals.

Audio Signals (Hz)	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
3775	-0.99	-0.99	-0.98	0.98	-0.99	-0.99	-0.98	0.99
3825	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.99	0.99
3875	-0.99	-0.99	-0.99	0.99	-0.99	-0.99	-0.98	0.99
3925	-0.99	-0.99	-0.99	0.98	-0.99	-0.99	-0.99	0.99
3975	-0.98	-0.99	-0.99	0.98	-0.98	-0.99	-0.99	0.99
4025	-0.99	-0.99	-0.99	0.99	-0.99	-0.99	-0.99	0.99
4075	-0.99	-0.99	-0.99	0.99	-0.99	-0.99	-0.99	0.99
4125	-0.99	-0.99	-0.99	0.99	-0.99	-0.99	-0.99	0.99
4175	-0.99	-0.99	-0.99	0.99	-0.99	-0.99	-0.99	0.99
4225	-0.99	-0.99	-0.99	0.99	-0.99	-0.99	-0.99	0.99

Table 4.20 ANN Topology 4 (500 neurons), for class 4 input signals.

With the information of the previous four tables it is possible to build the confusion matrix for the first topology, that contains basically a hidden layer of 500 neurons for the ANN. In the table 4.6 can be seen two confusion matrices, the first one is for the Spectrogram

ANN and the other for the mean of the Spectrogram ANN, the four classes in the tables are marked with the signal frequencies of the class under test.

		Spectrogram Actual Class				
		1 Khz	2 Khz	3 Khz	4 Khz	**Effectiveness**
Predicted Class	1 Khz	10	0	0	0	100%
	2 Khz	0	10	0	0	100%
	3 Khz	0	0	10	0	100%
	4 Khz	0	0	0	10	100%
		Total Effectiveness = 100%				

		Mean of Spectrogram Actual Class				
		1 Khz	2 Khz	3 Khz	4 Khz	**Effectiveness**
Predicted Class	1 Khz	10	0	0	0	100%
	2 Khz	0	10	0	0	100%
	3 Khz	0	0	10	0	100%
	4 Khz	0	0	0	10	100%
		Total Effectiveness = 100%				

Table 4.21 Confusion matrix for topology 4 (500 neurons).

The results of the table 4.21 shows that for the fourth ANN topology the total classification effectiveness was of 100% for the proposed input audio signals.

4.2 Performance results for the ANN with Real-World signals.

In section 3.2.2, it was described the proposed training and classification tests for an ANN with real-world signals, for that case was used just one topology (the first one of 30 neurons in the hidden layer, there is more details in the section 5.2 about the reason of this topology selection). Those real-world signals are the final purpose of this working thesis, to know the feasibility for using an artificial neural network in order to classify four kinds of maritime vessels with a high level of effectiveness using an acquisition system that was developed for this project. In the following sections it will be summarized the results for both tests, the training and classification ones.

4.2.1 Training metrics

The training metrics used for this section will be the following: error curve and a summary table, for both spectrogram and mean of spectrogram for the proposed topology that was 30 neurons in hidden layer for the neural network. As can be seen in the linear figures (for example in the figure 4.1), the error tends to drop suddenly in the first epochs and is hard to see the decline curve in the further epochs, for that reason was added a logarithmic plot in order to observe the decline of the curve in further epoch values.

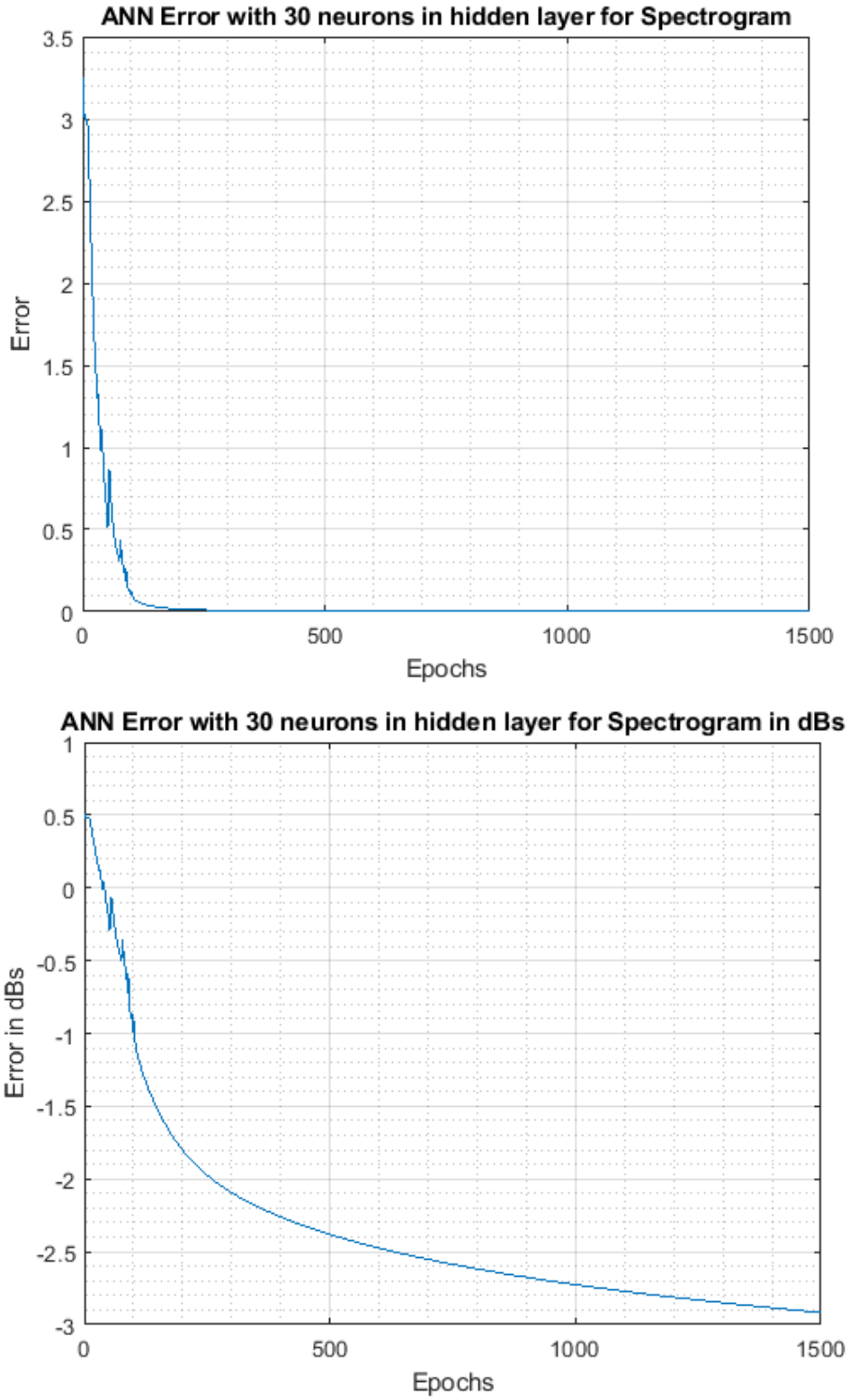


Figure 4.11: Error curves in lineal and logarithm scale for 30 neurons.

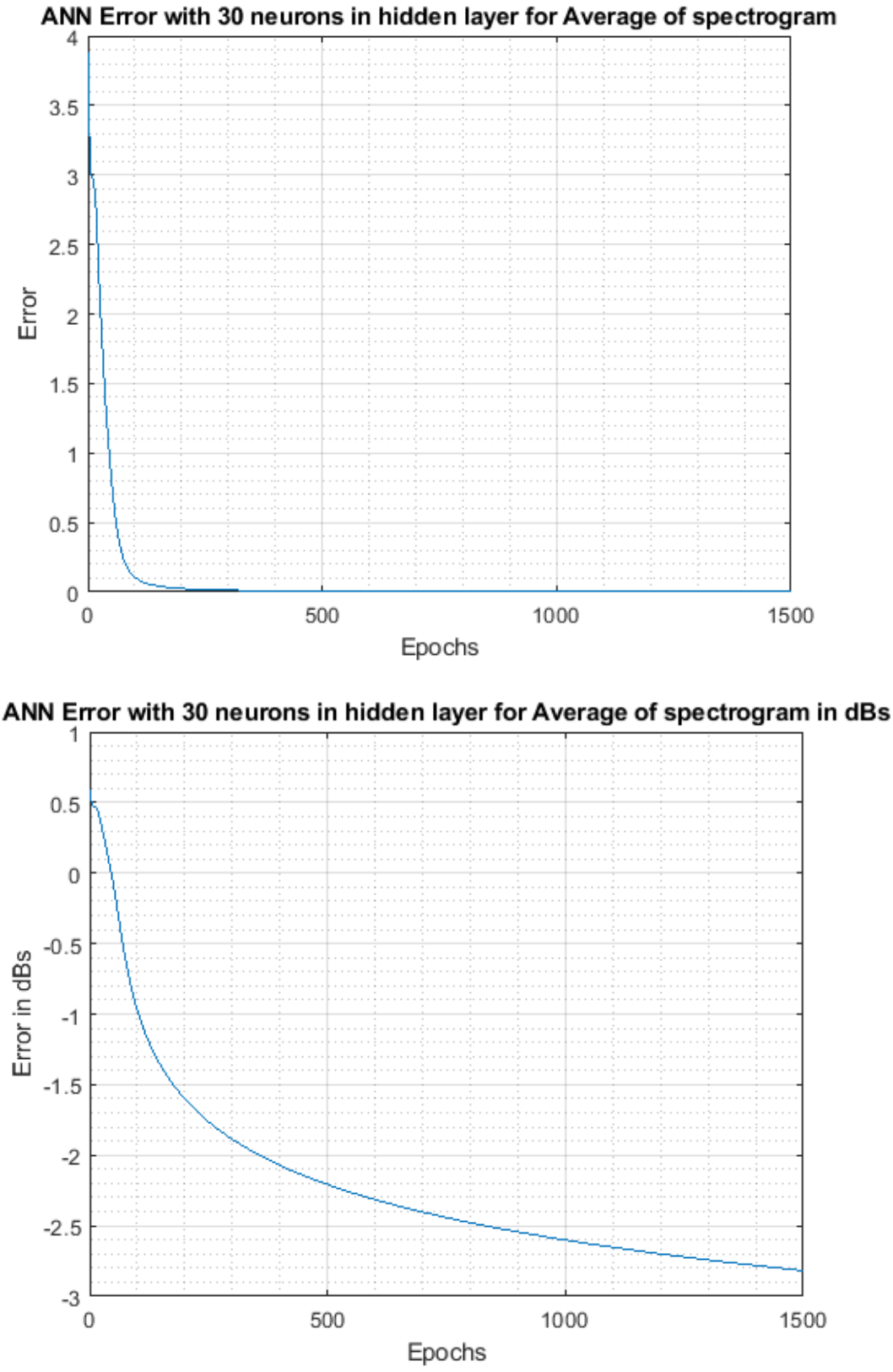


Figure 4.12: Error curves in lineal and logarithm scale for 30 neurons.

In the previous figures, was shown the training error curve of the proposed ANN. In the table 4.22 can be found a summary of the error behavior curves, that table is intended to give a glimpse of the behavior of the training process.

30 neurons in Hidden Layer	
Spectrogram	**Mean of spectrogram**
500 epochs: -2.4 dB	500 epochs: -2.2 dB
1500 epochs: -2.9 dB	1500 epochs: -2.8 dB

Table 4.22 Summary of the training error.

After the training process was done, the now trained neural network was tested with the same data that was used to train the ANN, in order to verify that the found weights and bias coefficients for the ANN are working fine at least for the best scenario.

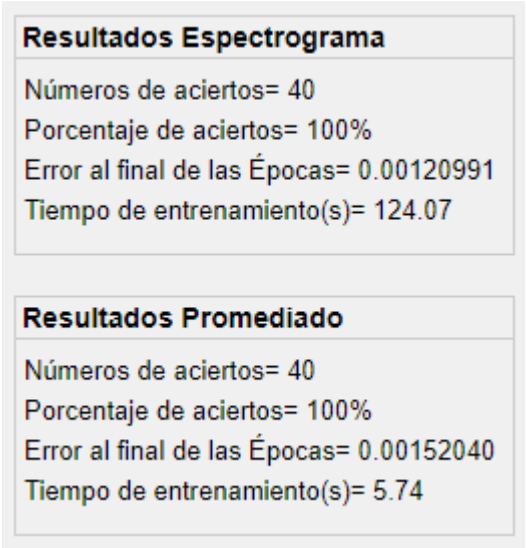


Figure 4.13: Training summary, for ANN.

4.2.1 Classification metrics

In the table 4.27, it can be seen the results of the classification tests using a confusion matrix for the ANN with of 30 neurons in the hidden layer. It was used ten test signals per class (those signals are different than those that were used in the training process).

In the following tables 4.23, 4.24, 4.25 and 4.26, it can be seen the results of the output layers values of the ANNs for the proposed classification tests. The output layer neurons values are in a range of -1 to 1, where the unit value means a 100% of accuracy that the

audio signal belongs to this class, and a minus one means a 0% of accuracy that the audio signal belongs to this class.

Audio Signals	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	ROV	**Merchant**	**Fishing**	**Boat**	**ROV**	**Merchant**	**Fishing**	**Boat**
ROV 1	0.97	-0.99	-0.99	-0.97	0.99	-0.98	-0.99	-0.99
ROV 2	0.98	-0.99	-0.99	-0.94	0.98	-0.99	-0.99	-0.99
ROV 3	0.98	-0.99	-0.99	-0.93	0.98	-0.99	-0.99	-0.99
ROV 4	0.97	-0.99	-0.98	-0.97	0.97	-0.99	-0.98	-0.99
ROV 5	0.96	-0.99	-0.95	-0.99	0.98	-0.99	-0.98	-0.99
ROV 6	0.96	-0.99	-0.97	-0.99	0.98	-0.99	-0.99	-0.99
ROV 7	0.97	-0.99	-0.98	-0.97	0.98	-0.99	-0.99	-0.99
ROV 8	0.98	-0.99	-0.99	-0.96	0.98	-0.89	-0.99	-0.99
ROV 9	0.96	-0.99	-0.97	-0.99	0.98	-0.99	-0.99	-0.99
ROV 10	0.97	-0.99	-0.97	-0.99	0.99	-0.99	-0.99	-0.99

Table 4.23 Results for class 1 input signals (ROV Submersible).

Audio Signals	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	ROV	**Merchant**	**Fishing**	**Boat**	**ROV**	**Merchant**	**Fishing**	**Boat**
Merchant 1	-0.99	0.97	-0.98	-0.96	-0.98	0.96	-0.99	-0.96
Merchant 2	-0.99	0.75	-0.99	-0.55	-0.97	0.97	-0.99	-0.96
Merchant 3	-0.99	0.97	-0.99	-0.98	-0.99	0.96	-0.99	-0.97
Merchant 4	-0.99	0.80	-0.98	-0.83	-0.98	0.96	-0.99	-0.97
Merchant 5	-0.99	0.85	-0.99	-0.89	-0.98	0.96	-0.99	-0.95
Merchant 6	-0.99	0.95	-0.99	-0.93	-0.98	0.97	-0.99	-0.97
Merchant 7	-0.99	0.96	-0.98	-0.96	-0.99	0.98	-0.99	-0.98
Merchant 8	-0.99	0.85	-0.98	-0.87	-0.98	0.96	-0.99	-0.97
Merchant 9	-0.99	0.95	-0.98	-0.98	-0.98	0.95	-0.99	-0.98
Merchant 10	-0.99	0.80	-0.99	-0.83	-0.98	0.97	-0.99	-0.97

Table 4.24 Results for class 2 input signals (Merchant).

Audio Signals	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	ROV	**Merchant**	**Fishing**	**Boat**	**ROV**	**Merchant**	**Fishing**	**Boat**
Fishing 1	-0.98	-0.99	0.95	-0.98	-0.98	-0.99	0.93	-0.97
Fishing 2	-0.97	-0.99	0.90	-0.95	-0.98	-0.99	0.98	-0.98
Fishing 3	-0.98	-0.97	0.96	-0.99	-0.98	-0.99	0.95	-0.99
Fishing 4	-0.98	-0.98	0.97	-0.99	-0.98	-0.99	0.97	-0.98
Fishing 5	-0.98	-0.98	0.97	-0.99	-0.98	-0.99	0.97	-0.97
Fishing 6	-0.95	-0.99	0.80	-0.84	-0.98	-0.99	0.97	-0.96
Fishing 7	-0.98	-0.99	0.95	-0.95	-0.98	-0.99	0.98	-0.95
Fishing 8	-0.95	-0.99	0.82	-0.75	-0.98	-0.99	0.98	-0.96
Fishing 9	-0.95	-0.99	0.83	-0.91	-0.98	-0.99	0.97	-0.95
Fishing 10	-0.89	-0.99	0.54	-0.76	-0.98	-0.99	0.97	-0.96

Table 4.25 Results for class 3 input signals (Fishing boat).

Audio Signals	Spectrogram (Output layer Values)				Mean of Spectrogram (Output layer Values)			
	ROV	**Merchant**	**Fishing**	**Boat**	**ROV**	**Merchant**	**Fishing**	**Boat**
Boat 1	-0.98	-0.98	-0.98	0.96	-0.99	-0.98	-0.95	0.94
Boat 2	-0.99	-0.93	-0.71	0.01	-0.99	-0.98	-0.90	0.89
Boat 3	-0.99	-0.98	-0.98	0.90	-0.99	-0.98	-0.98	0.97
Boat 4	-0.98	-0.98	-0.98	0.96	-0.99	-0.97	-0.98	0.95
Boat 5	-0.99	-0.99	-0.98	0.95	-0.99	-0.97	-0.95	0.96
Boat 6	-0.90	-0.99	-0.99	0.95	-0.99	-0.98	-0.97	0.94
Boat 7	-0.98	-0.98	-0.95	0.88	-0.99	-0.99	-0.83	0.84
Boat 8	-0.98	-0.98	-0.97	0.93	-0.99	-0.98	-0.95	0.89
Boat 9	-0.99	-0.91	-0.98	0.87	-0.99	-0.94	-0.99	0.96
Boat 10	-0.99	-0.95	-0.98	0.90	-0.99	-0.95	-0.99	0.95

Table 4.26 Results for class 4 input signals (Motorboat).

With the information of the previous four tables it is possible to build the confusion matrix for the first topology, that contains basically a hidden layer of 30 neurons for the proposed ANN. In the table 4.27 can be seen two confusion matrices, the first one is for the

Spectrogram ANN and the other for the mean of the Spectrogram ANN, the four classes in the tables are marked with the signal frequencies of the class under test.

		Spectrogram Actual Class				
		ROV	Merchant	Fishing	Boat	**Effectiveness**
Predicted Class	ROV	10	0	0	0	100%
	Merchant	0	10	0	0	100%
	Fishing	0	0	10	0	100%
	Boat	0	0	0	10	100%
		Total Effectiveness = 100%				

		Mean of Spectrogram Actual Class				
		ROV	Merchant	Fishing	Boat	**Effectiveness**
Predicted Class	ROV	10	0	0	0	100%
	Merchant	0	10	0	0	100%
	Fishing	0	0	10	0	100%
	Boat	0	0	0	10	100%
		Total Effectiveness = 100%				

Table 4.27 Confusion matrix for topology 1 (30 neurons).

The results of the table 4.6 shows that for this first ANN topology the total classification effectiveness was of 100% for the proposed input audio signals.

4.3 Performance Analysis.

In the previous sections in 4.1 and 4.2, were shown the results for both the training and classification process, for the proposed ANN topologies and input signals. Now in this section it will be summarize it all the retrieved data to finally compare the performance for every presented case.

4.3.1 Synthetic signals for training and classification of ANN

In the table 4.28, it can be seen the summarized data of the figures 4.9 and 4.10. That info is related to the training metrics of the four proposed topologies (30, 100,200, 500 neurons in the hidden layer) for both inputs the full spectrogram and the mean of the spectrogram. The first metric is "Test Pass" that is intended to measure how well the found training coefficients (weight and bias) are performing when the training data is introduced in the new ANN and run a classification test, for this case was used 40 training samples. The second metric is "Final Error" in the figures 4.1 until 4.8 were presented the training error graphs and there can be seen the progress of the error in function of the epochs, but for this table just the last data is taking in account. And the third metric is "Time" this value informs to the user, how long time take to the training algorithm (backpropagation) complete the training for a certain number of epochs (for this case was 1500). For the "Test Pass" metric was not necessary to create a graph because there is not variation of the results for all cases (100%).

ANN TOPOLOGY	Spectrogram			Mean of Spectrogram		
	Test Pass	Final Error	Time(s)	% Test Pass	Final Error	Time(s)
30	100%	0.00085	126.22	100%	0.00151	6.40
100	100%	0.00037	505.47	100%	0.00117	11.94
200	100%	0.00028	1028.71	100%	0.00110	43.51
500	100%	0.00024	2585.85	100%	0.00101	155.66

Table 4.28 Comparative of final training metrics.

In the figure 4.14, can be seen the results of plotting the final training error for both inputs the full spectrogram and the mean of the spectrogram, in function of the proposed topologies. And for figure 4.15, it can be seen the results of plotting the final training time for both inputs the full spectrogram and the mean of the spectrogram, in function of the proposed topologies. The final conclusions of those graphs will be discussed in the next chapter.

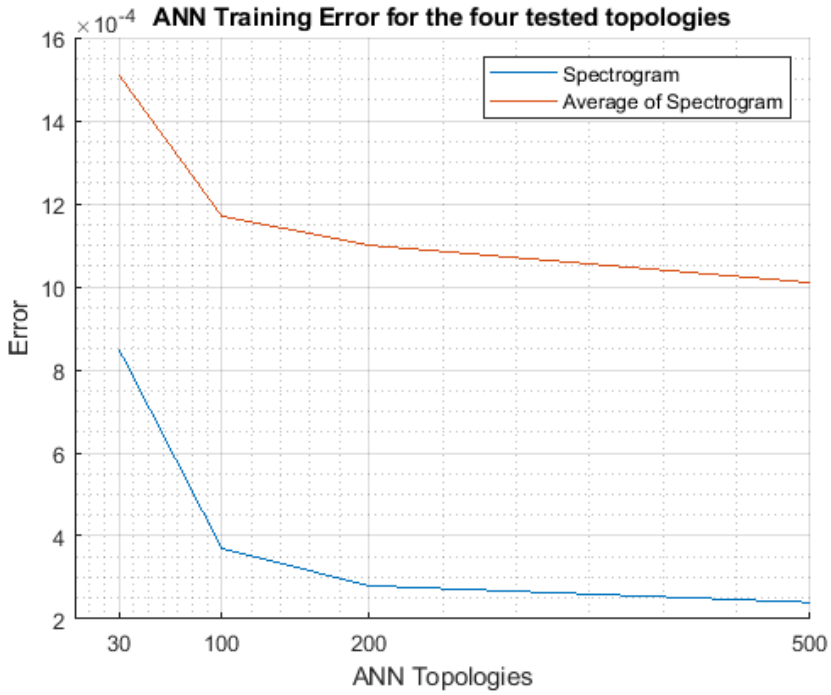


Figure 4.14: Training error summary, for ANN.

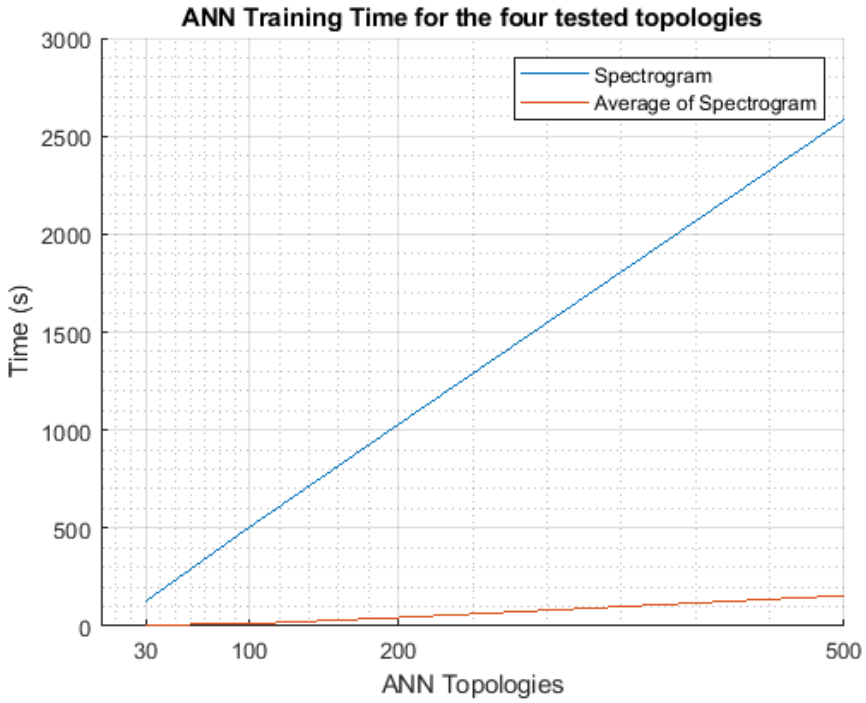


Figure 4.15: Training time summary, for ANN.

In the table 4.29, can be seen the summarized data of the tables 4.2 to 4.21. That info is related to the classification metrics of the four proposed topologies (30, 100,200, 500 neurons in the hidden layer) for both inputs the full spectrogram and the mean of the spectrogram. The first metric is "Effectiveness" that is intended to measure how well the classificator agent is performing when the synthetic acoustic signals are put under a classification test, the effectiveness data was taken from the confusion matrices in the tables 4.6, 4.11, 4.16, 4.21. The second metric is "True Positive" that is intended to measure the sensitivity of the output layer neuron that belongs to the true positive class (the correct class). And due there are four different classes under test per ANN topology, and every class has its own sensitivity, it was used the arithmetic mean and standard deviation of those classes, and that final result was put in the table. And the third metric is "False Negative" that is intended to measure the sensitivity of the output layer neurons that belongs to the false negative class (the incorrect classes). And due there are four different classes under test per ANN topology, and there are also three false negative classes with its own sensitivity, it was used the arithmetic mean and the standard deviation of those twelve sensitivities, and that final result was put in the table.

ANN TOPOLOGY	Spectrogram (Output layer Values)			Mean of Spectrogram (Output layer Values)		
	Effectiveness	True Positive	False Negative	Effectiveness	True Positive	False Negative
30	100%	0.9722 ± 0.0188	-0.9866 ± 0.0082	100%	0.9655 ± 0.0289	-0.9844 ± 0.0135
100	100%	0.9715 ± 0.0230	-0.9858 ± 0.0084	100%	0.9672 ± 0.0326	-0.9819 ± 0.0335
200	100%	0.9732 ± 0.0324	-0.9867 ± 0.0084	100%	0.9660 ± 0.0305	-0.9806 ± 0.0374
500	100%	0.9695 ± 0.0516	-0.9870 ± 0.0103	100%	0.9442 ± 0.1423	-0.9824 ± 0.0375

Table 4.29 Comparative of final classification metrics.

In the figure 4.16, can be seen the results of plotting the mean of the sensitivities of the output neurons for the true positives values (the correct classification), for both inputs the full spectrogram and the mean of the spectrogram, in function of the proposed topologies. And for the figure 4.17, can be seen the results of plotting the mean of the sensitivities of the output neurons for the true False Negatives (the wrong classifications), for both inputs the full spectrogram and the mean of the spectrogram, in function of the proposed topologies. The final conclusions of those graphs will be discussed in the next chapter.

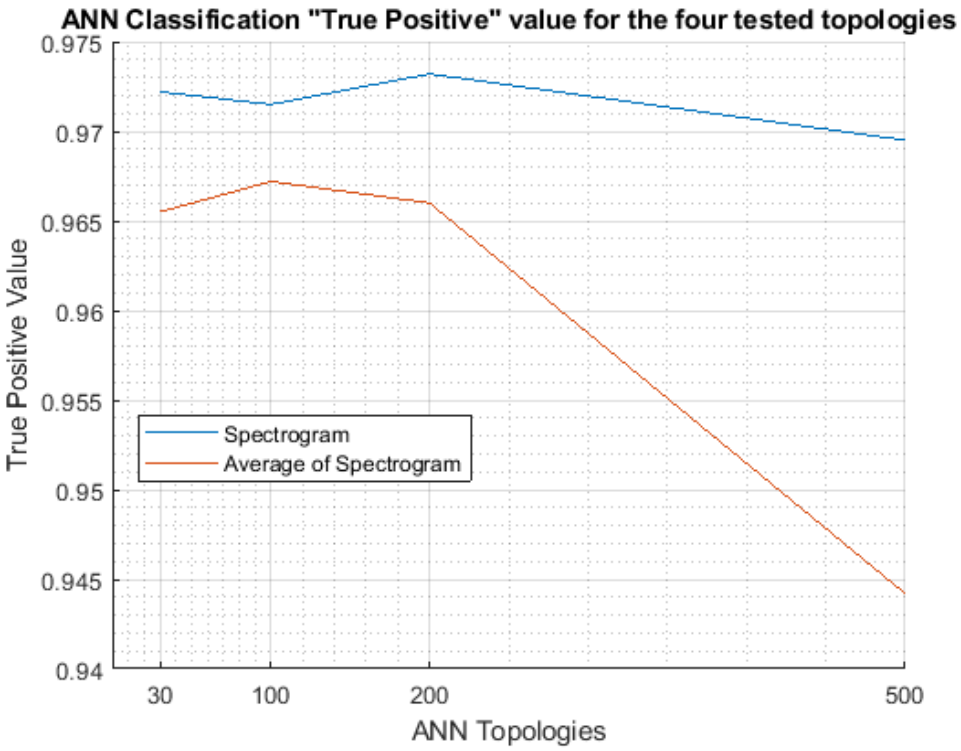


Figure 4.16: Classification quality summary for "True Positive".

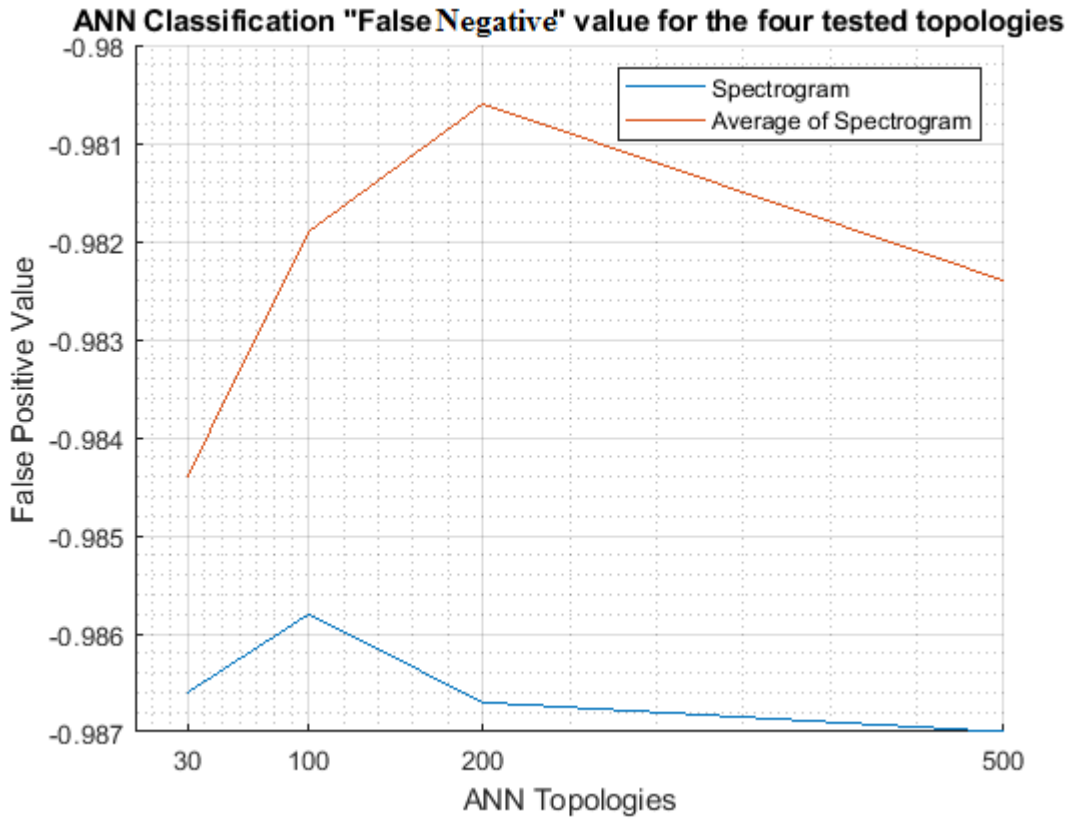


Figure 4.17: Classification quality summary for "False Negative".

4.3.2 Real-world signals for training and classification of ANN

Now, for the case of using real-world signals the same procedure that was explained in the section 4.3.1 was implemented but now just for one single ANN topology in this section. The training results can be seen in the table 4.30. As was seen in the previous sections, all that info is related to the training metrics of the proposed ANN topology (30 neurons in the hidden layer) for both inputs the full spectrogram and the mean of the spectrogram.

ANN TOPOLOGY	Spectrogram (Output layer Values)			Mean of Spectrogram (Output layer Values)		
	Test Pass	Final Error	Time(s)	% Test Pass	Final Error	Time(s)
30	100%	0.0012	124.7	100%	0.0015	5.74

Table 4.30 Comparative of final training metrics.

In the table 4.31, can be seen the summarized data of the tables 4.23 to 4.26. That info is related to the classification metrics of the selected ANN topology (30 neurons in the

hidden layer) for both inputs the full spectrogram and the mean of the spectrogram for 40 samples tests.

ANN TOPOLOGY	Spectrogram (Output layer Values)			Mean of Spectrogram (Output layer Values)		
	Effectiveness	True Positive	False Negative	Effectiveness	True Positive	False Negative
30	100%	0.8887 ± 0.1668	-0.9619 ± 0.0616	100%	0.9602 ± 0.0293	-0.9786 ± 0.0212

Table 4.31 Comparative of final classification metrics.

The final conclusions of those tables will be discussed in the next chapter.

5 CONCLUSIONS AND FUTURE WORK.

In this thesis work, was developed and evaluated an Artificial Neural Network with different topologies and acoustic input signals, with the chief end to classify those signals. Another important purpose for this work is related to provide metrics to evaluate the viability of using the mean of the spectrogram input data instead of a full spectrogram image, the averaged spectrogram contains much lesser data, and that for an ANN means a smaller network, and a smaller network means lesser training time and another benefits.

Now that the final metrics were gotten, it is possible to give some conclusions about performance, advantages and disadvantages, etc.

5.1 Synthetic signals training and classification conclusions.

One of the most difficult tasks for this thesis project was to collect as many real-world acoustic signals from maritime vessels as possible, the difficulty lies in the logistics costs, time expended, weather conditions, good positioning of the hydrophone in relation to the vessel in order to acquire the correct acoustic signature, the ever-changing environment noise in underwater environment, maritime vessels traffic flow, etc. So, the first attempt in this project was to work with synthetic audio signals, in order to overcome this initial limitation, so, with those signals, can be easily test the implementations of the ANNs, and correct the design-implementation failures in a quickly way, and also was intended to test rapidly different ANN topologies and get the first results and some implementation experiences, in order to finally have better references in order to select the better ANN topology, for testing the classificator with real-world acoustic signatures.

So, the performance conclusions for the *training* of the ANNs with synthetic signals, can be taken from the table 4.28, there can be seen clearly that the training algorithm was able to find the proper weights and bias coefficients for all ANNs proposed topologies and for both Spectrogram and mean of Spectrogram. Even though all training cases were successful, yet there are differences in the final training errors and time consumed. In the figure 4.14 can be seen the graph that describes the error behavior in function of the

topologies, and is relatively easy to conclude that bigger neural networks reduce the training error in comparison than smaller ones for both cases Spectrogram and mean of Spectrogram. But still the error difference between topologies is very small to the point of ten thousandths of unity. And in regards of the training time the figure 4.15 helps to see that bigger neural networks increments the training time in comparison than smaller ones for both cases Spectrogram and mean of Spectrogram. For this case the training time difference between topologies is significant, especially for the Spectrogram ANNs.

Now, the classification conclusions for the four proposed ANNs with synthetic signals, can be taken from the table 4.29, there can be seen clearly that the classification agent was able to classify in a proper way with an effectiveness of 100% for all ANNs proposed topologies and for both Spectrogram and mean of spectrogram for the proposed audio signals. Even though all the classification tests were successful, yet there are differences in the quality of the classifications, and those metrics were labeled as “True Positive” and “False Negative”. In the figure 4.16, it can be seen the graph that describes the quality of the classificator for “True Positive” in function of the topologies, and it can be concluded that for the case of the Spectrogram topology the quality seems to stay constant but for the mean of spectrogram topology looks constant in the first three topologies but for the last one tends to decrease with a significant negative slope, this behavior is probably due to the neural network hidden layer with many neurons (500) compared with not many input neurons (1024) for this scenario, and that make the quality of the classification decrease. And the last quality metric can be found in the figure 4.17 that graph describes the quality of the classificator for “False Negative”, and it can be concluded that for the case of the Spectrogram topology the quality seems to stay constant but for the mean of spectrogram topology looks no really constant but the values differences are not very spread.

So, with the base of the previous conclusions using synthetic audio signals, the chosen ANN to be trained and tested with Real-World signals was the first topology that is built with 30 neurons in the hidden layer, even though that topology is the one with more training error, but the difference between this topology and the one with lesser error is almost insignificant, and by choosing this, the training speed is the faster one (between

the proposed topologies) for many units of time. And the quality of the classification for both spectrogram and mean of spectrogram don't change significantly for this first topology.

5.2 Real-world signals training and classification conclusion.

In the previous section 5.1, was described the conclusion process for different ANN topologies in order to select the most suitable, for put under test. And the selected topology was the ANN with 30 neurons in the hidden layer.

For this thesis project was possible to acquire approximately 40 different audio signals per class, and for the first class (Submersible ROV) was just possible to use one kind of this class, but the ANN was trained with different speed of the motor, so the acoustic signatures looked no exact the same. For the second class (Merchant vessel), was used acoustic signatures of two different kinds of that class. For the third class (Fishing boat), was used acoustic signatures of only one class. And for the last class (Motorboat) also was used acoustic signatures of only one class.

So, the performance conclusions for the *training* of the ANNs with Real-World signals, can be taken from the table 4.30, there can be seen clearly that the training algorithm was able to find the proper weights and bias coefficients for the proposed topology and for both spectrogram and mean of spectrogram. The metrics results are very similar compared with the results of the same topology but with synthetic signals, the only meaningfully difference is the error metric that was increased for Real-World signals for the Spectrogram input, but even though the error still is low.

Now, the *classification* conclusions for the proposed ANN with Real-World signals, can be taken from the table 4.31, there can be seen clearly that the classification agent was able to classify in a proper way with an effectiveness of 100% for both spectrogram and mean of spectrogram for the proposed audio signals. The metrics results are very similar compared with the results of the same topology but with synthetic signals, the only meaningfully difference is the "True Positive" metric that was decreased for Real-World signals for the Spectrogram input, but even though the metric still is pretty good.

5.3 Hypothesis conclusions.

The proposed hypothesis for this thesis work is the following: "It is possible to create a classifier algorithm using acoustic signals for training an artificial neural network, for maritime vessels purposes. And verify the accuracy of the results for different kinds

of morphologies, which are in function of the input and hidden layers of the neural network".

With the support of the metrics that were retrieved for the different tested ANN topologies and input signals, can be stated that this hypothesis is **accepted**, due for all the training and classification cases was gotten a 100% of effectivity. But it has to take into account that the audio database was not large enough and with a wide variety of maritime vessels.

5.4 Final conclusions.

It was proposed and implemented an own acquisition system, and the results shows, that system possess a pretty good quality for acquiring underwater acoustic signals, and is a very portable, so, can be used it in small boats without the need of AC electric energy, due that can be operated on batteries, all those features are a great asset for the task of collecting acoustic signals in the sea.

Even though all the proposed training and classification tests for the ANNs were successful 100% of the time, it is clear that the true potential of this implementation was not really tested due of the lack of a big database of acoustic signals.

Another important conclusion that can be taken out of this work: It is about the no need to work with a complete image of the spectrogram as an input for an ANN, it can be gotten good training and classification results(almost the same results as using the full image) just using one averaged spectrogram (one single input vector), with the advantage of reducing the number of neurons in the ANN, and the training speed increase and others benefits. The main reason observed for this particular behavior for maritime vessels acoustic signatures is that most of those signals repeats its pattern over identical subsequent periods of time, so when a mean to the full spectrogram is made, there is not lost information, even it is possible to see an enhanced graph.

5.5 Future work.

There are many areas of improvement for this work. In the following points are described the suggested ones.

- Get a big database of acoustic signatures for different kinds of maritime vessels.
- Perform more training and classification tests now altering the following ANN variables: learning rate, number of layers for hidden layer, number of training epochs, and register the new metrics in order to get new conclusions.
- Compare the ANNs implemented in this work with another ANNs embedded in libraries like Tensorflow, Caffe, Theano, and many others.
- Implement this classificator in an embedded system in order to use it in a cheap electronic application.
- Migrate this implementation to an open source language like python or java, in order to improve the portability of the code. And increase the possibility to be used in a real sonar equipment.
- Create an own electronic preamplifier instead of the proposed instrumentation equipment due its cost is high.

6 REFERENCES

[1] Qihu Li. (2011). Digital Sonar Design in Underwater acoustics principles and applications. Beijing, China

[2] J-C Di Martino, S. Tabbone (1994). An approach to detect lofar lines. Vandauvre-les-Nancy, France.

[3] Talmor Meir, Mikhail Tsionskiy, Dr. Alexander Sutin, Dr. Hady Salloum (2012). Decision Learning Algorithm for Acoustic Vessel Classification. New York, USA.

[4] N.Leal, E. Leal and G. Sanchez (2015), Marine Vessel Recognition By Acoustic Signature. Barranquilla Colombia.

[5] Hamed Komari Alaie, Hassan Farsi. (2018). Passive Sonar Target Detection Using Statistical Classifier and Adaptive Threshold. Birjand, Iran.

[6] Chin-Hsing Chen, Jiann-Der Lee, Ming-Chi Lin (2000). Classification of Underwater Signals Using Neural Networks. Taiwan.

[7] Richard G. Lyons (consulted 2018). DSP Tricks: Computing Fast Fourier Transform Twiddle Factors. https://www.embedded.com/print/4007620.

[8] L. Castellanos, J. Aguilar, M. Alvarado (2016). A LOFAR and beamforming implementation in a FPGA for a digital passive sonar. Mexico.

[9] Trearen, P., Pacheco M., Vellasco M., VLSI architectures for neural networks. IEEE Micro, pp. 8-27 (1989).

[10] Grenie M. Acoustic detection of propeller cavitation. ICASSP'90, pp. 2907-2910 (1990).

[11] Alex Holehouse (consulted 2018). Classification using Logistic Regression. http://www.holehouse.org/mlclass/06_Logistic_Regression.html

[12] Daniel Shiffman (consulted 2018). The nature of Code, Chapter 10, Neural Networks. https://natureofcode.com/book/chapter-10-neural-networks/.

[13] Electronics Tutorials (consulted 2018). Zener Diode Clipping Circuits. https://www.electronics-tutorials.ws/diode/diode_7.html.

[14] Function Generator App from KEUWLSOFT (consulted 2018). Operation Manual. http://www.keuwl.com/FunctionGenerator/

[15] Seawolf by TT Robotix (consulted 2018). Product Description. http://www.ttrobotix.com/product/seawolf-sport

[16] Stehman, Stephen V. (1997). "Selecting and interpreting measures of thematic classification accuracy". Remote Sensing of Environment. 62 (1): 77–89

[17] Ministry of Defence Govt of India (2018), "Sonar Technology" https://www.drdo.gov.in/drdo/pub/techfocus/oct2000/sonar%20technology.htm

[18] N. N. de Moura, J. M. de Seixas and Ricardo Ramos, "Passive Sonar Signal Detection and Classification Based on Independent Component Analysis". Brazil. http://cdn.intechopen.com/pdfs/18872/InTech-Passive_sonar_signal_detection_and_classification_based_on_independent_component_analysis.pdf

[19] Radionerds.com (2018), "PPI Scope", http://radionerds.com/index.php/PPI_Scope

[20] Mathworks, Matlab (2018). "Applying Supervised Learning", https://www.mathworks.com/content/dam/mathworks/tag-team/Objects/i/90221_80827v00_machine_learning_section4_ebook_v03.pdf

[21] Miguel Alvarado (2017), "Tecnología subacuática para protección y aprovechamiento de los intereses marítimos nacionales", http://repositorio.uninav.edu.mx:8080/xmlui/bitstream/handle/23000/340/da_44-17.pdf?sequence=1&isAllowed=y

[22] J.D. Kraus, (1966), "Radio Astronomy", McGraw-Hill.

7 APPENDIX

APPENDIX 1 IMPLEMENTING A MULTILAYER ARTIFICIAL NEURAL NETWORK

In this appendix, it will be described the process of implementing one artificial neural network that was used in this thesis project.

There are a set of questions that need to be answered in order to start de design of the ANN.

- How many neurons has the ANN input layer? R
- How many neurons the hidden layer has? S
- How many neurons the output layer has? Number of classes
- How many layers the ANN has? M
- How many training samples for all classes? Q
- What activation function it will be used? F
- What weights and bias values are needed? W and B.

In order to simplify this design exercise some fixed values are proposed, R=1024, S=30, Classes = 4, M= 3, Q= 40. In order to visualize the ANN and its proposed values was added the figure 7.1.

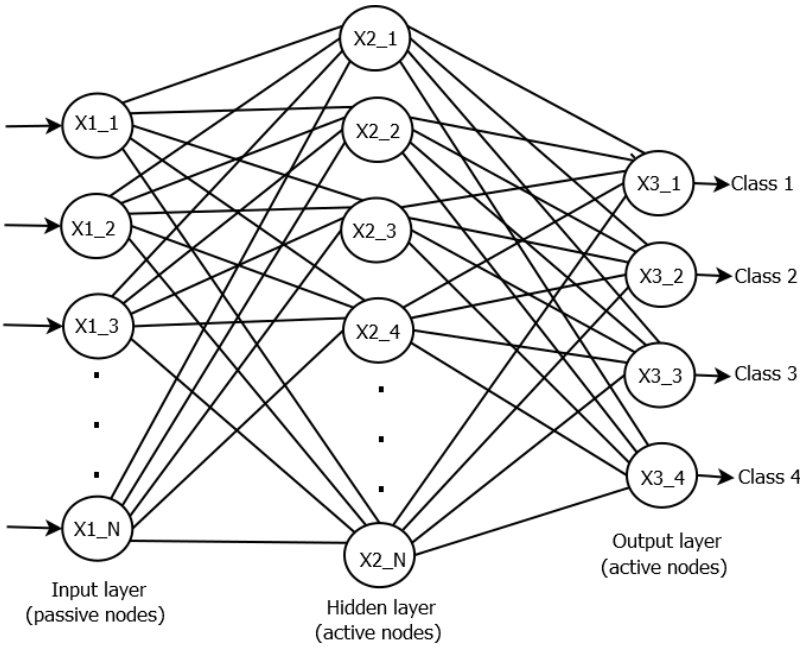


Figure 7.1: Artificial Neural Network schema.

In order to represent a multilayer ANN in a mathematical form in such a way that can be put in a computer software language, it is necessary to understand that many mono layer networks connected in series give the result of a multilayer ANN. For example, in the

figure 7.2 there is a representation of an ANN with two layers with active nodes connected in series, those layers represent the hidden layer and the output layer. The first layer that corresponds to the input layer the nodes or neurons are passive, so there is no need to add a monolayer neural network to represent it in the schema. So, the number of total layers for this topology is M=3, the input layer is represented with the vector p (patrons), the hidden layer output represented with the vector a^1 (axon), and the output layer represented with the vector a^2 (axon). The others vector values "W" are for the synaptic weights, and "b" for the bias of the respective monolayer networks in the schema. And finally, the letter "n" are the net outputs and the "f" blocks represents the employed activation functions.

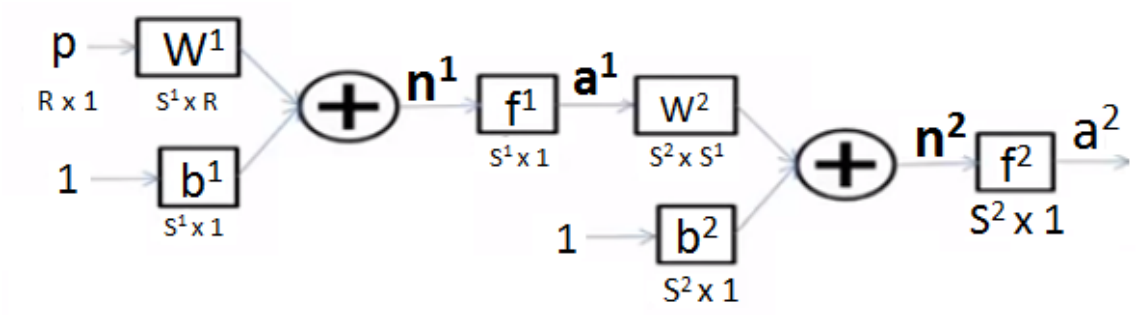


Figure 7.2: Artificial Neural Network.

The final output of the multilayer network can be represented as a linear combination with the activation functions embedded, as described in the equation 7.1.

$$a^2 = F^2(W^2F^1(W^1p + b^1) + b^2) \tag{7.1}$$

The equation 7.1, can be decoded to software lines in order to be executed by a computer. In the following lines, it is shown one example of a way to represent a multilayer ANN using MATLAB code.

```
% Propagación de la entrada hacia la salida (adelante)
a1 = tg_sig(W1*P(:,q) + b1);
a2 = tg_sig(W2*a1 + b2);
```

The next question to be answered is, what activation function it will be used? There are many activations functions that can be used, but a popular recommendation for

classification problems for ANN is to use codification -1 and 1, using activation functions of type tangential sigmoid. It was proposed to use a hyperbolic sigmoid tangential, that can be described with the equation 7.2.

$$f(\mathrm{n}) = \frac{e^{n} - e^{-n}}{e^{n} + e^{-n}} \tag{7.2}$$

In the figure 7.3 can be see the behavior of the hyperbolic sigmoid tangential, based on the equation 7.2, and the codification is for a maximum value 1 and minimum -1.

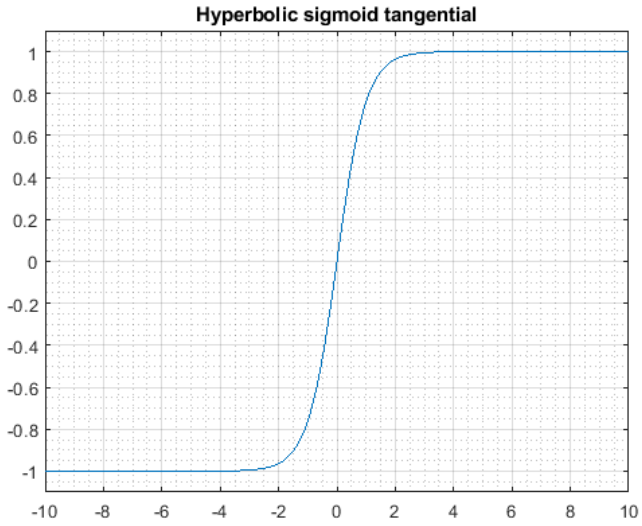


Figure 7.3: Activation function.

The Matlab implementation of this activation function can be seen in the following lines.

```
function y = tg_sig(n)
      y = (exp(n) - exp(-n))./(exp(n) + exp(-n));
```

The last and the most challenging question to answer is, What weights and bias values are needed for the ANN? In order to provide the needed weights and bias values to use in the proposed ANN, it is a very impractical way to start to calculate those values by hand, especially when it is in use a multilayer network with many neurons. So, it is necessary to use an algorithm that can be able to calculate those values in an automatic way, this process is generally known as the training stage.

The chosen algorithm for the training process is the backpropagation, this algorithm is intended to use in multilayer networks with many Q input training patrons, and the goal of this process is to minimize the squared error function, using the method of gradient descent. The combination of weights and bias which minimized the error function is considered to be a solution of the learning problem.

There are three basic steps in order to implement the backpropagation algorithm:

1) Propagate forward the stimulus of the ANN, initializing the values for W and b randomly. Using the recursive 7.1 equation, in order to calculate the output of the first layer and then calculate the output of the second layer with the previous retrieved values.

```
% Propagación de la entrada hacia la salida (adelante)
a1 = tg_sig(W1*P(:,q) + b1);
a2 = tg_sig(W2*a1 + b2);
```

2) Backpropagate the sensitivity of every single layer, in order to accomplish this task, it must be calculated the error, and that error can be defined as the difference between the desired value and the actual value, for the desired value was used a targets (T) vector.

```
e = T(:,q) - a2;      %error = deseado - actual
```

Now with this error vector, can be computed the sensitivity of the last layer and later the penultimate, and so on until reach the first layer. In this section of the code is where the method of the gradient descend is applied, it's not part of the scope of work of this appendix to demonstrate the origin of those lines of code.

```
s2 = -2*diag(1-a2.^2)*e;
s1 = diag(1-a1.^2)*W2'*s2;
```

3) Now that the sensitivities of the layers are known it is time to update the weights and bias values of the ANN, where "alpha" is the learning rate.

```
% Actualización de pesos sinapticos y polarizaciones
W2 = W2 - alfa*s2*a1';
b2 = b2 - alfa*s2;
W1 = W1 - alfa*s1*P(:,q)';
b1 = b1 - alfa*s1;
```

To finally to calculate the squared error and saved it in a vector (ec).

```
% Error Cuadrático
ec(k) = e'*e;
```

And when the training is done for one epoch, it can be calculated the mean squared error and save it in a vector (ecm).

```
%Error cuadratico medio
ecm(Epocas)= sum(ec)/Q;
```

In the below lines can be seen the code in MATLAB on charge of the implementation of the training process using the backpropagation algorithm for a neural network with 3 layers, M=3 (one passive and two active layers).

```
for Epocas = 1:EpocaLimit
    for k = 1:Q
        q=k;
        % Propagación de la entrada hacia la salida (adelante)
        a1 = tg_sig(W1*P(:,q) + b1);
        a2 = tg_sig(W2*a1 + b2);
        % Retropropagación de la sensibilidades
        e = T(:,q) - a2;            %error = deseado - actual
        s2 = -2*diag(1-a2.^2)*e;
        s1 = diag(1-a1.^2)*W2'*s2;
        % Actualización de pesos sinapticos y polarizaciones
        W2 = W2 - alfa*s2*a1';
        b2 = b2 - alfa*s2;
        W1 = W1 - alfa*s1*P(:,q)';
        b1 = b1 - alfa*s1;
        % Error Cuadrático
        ec(k) = e'*e;
    end
    ecm(Epocas)= sum(ec)/Q;     %error cuadratico medio
end
```

Lastly, in order to test the previously trained ANN and verify the performance, it is needed to load the weights and bias coefficients found by the backpropagation algorithm and use them in the recursive equation. The result will be a vector with the final values of the four neurons for the output layer (for this example case), those values are the final results of the classification. For this example “EspectrogramaNorm” represents the input vector to be tested by the classificator.

```
 %---resultados clasificador
result_neuronas = tg_sig(W2*tg_sig(W1*EspectrogramaNorm + b1)+ b2);
```