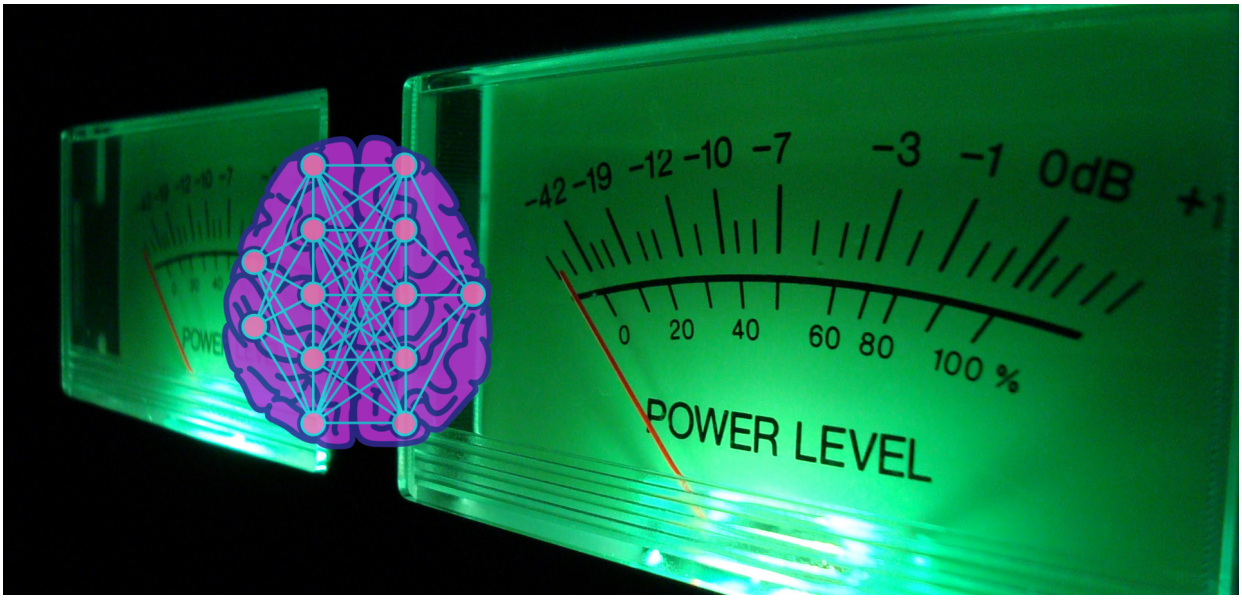
CHALMERS
UNIVERSITY OF TECHNOLOGY



Crest Factor Reduction Based On Artificial Neural Networks

Master's thesis in Complex Adaptive Systems

EDUARDO SESMA CASELLES

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Master's thesis 2017:108

Crest Factor Reduction Based On Artificial Neural Networks

EDUARDO SESMA CASELLES

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2017

Crest Factor Reduction Based On Artificial Neural Networks
EDUARDO SESMA CASELLES

Supervisor: Amir Eghbali, Lab.gruppen AB
Examiner: Henk Wymeersch, Department of Electrical Engineering

Master's Thesis 2017:108
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Decibel level meter along with a symbolic representation of an Artificial Neural Network at the centre.

Typeset in LaTeX
Gothenburg, Sweden 2017

Crest Factor Reduction based on Artificial Neural Networks
EDUARDO SESMA CASELLES
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Nowadays most audio amplifiers for professional use include advanced signal processing techniques. While most of these signal processing relate to adjusting the spectral characteristics of the sound (like equalization), many more hidden features aim to guarantee proper function of the amplifier in conditions close to its operational limits. For example, how to handle situations like the main supply being insufficient or the input signal amplitude is too high for the amplifier to generate a proportional output. Notice that in both aforementioned problems, the output signal ends up being clipped with obvious consequence on the perceived audio quality. To counteract this effect, specific signal processing is introduced: a so called "limiter" has the purpose of re-shaping the signal in order to produce a sort of instantaneous compression of the signal itself, aiming to minimize the perception of distortion.

A common implementation of such algorithms involve the introduction of look-ahead buffers, i.e. the signal is delayed in order to accommodate for its processing. The longer the delay, the better the efficacy of the algorithm to contain the audible distortion. Needless to say, such delay is in other regards undesirable. Therefore the research about an instantaneous algorithm, which is the topic of this thesis work.

The algorithm we have developed is based on an Artificial Neural Network (ANN) trained by a Genetic Algorithm (GA). The ANN is trained and its behaviour shaped according to a set of rules embedded in the GA. Different set of rules are evaluated and the result exposed in this report.

The result is a non-deterministic system: being based on an ANN, each training session may lead to different outcomes. Nevertheless, the methods developed yield reliable convergence toward the desired behaviour, i.e. the reduction of the harmonic distortion. A comparison between the new method and the existing algorithms is also presented. It was also observed that such new methodology opens so far unexplored research paths about signal processing in general.

Keywords: Crest Factor Reduction, Audio Limiter, Audio Compressor, Audio Signals Processing, Artificial Neural Networks, Genetic Algorithms.

Acknowledgements

This thesis has been carried out at LabGruppen AB and in collaboration with Chalmers University of Technology.

I would like to express my outmost gratitude to all my skilled and creative colleagues at LabGruppen who have taken part in the development of this thesis. I would like to mention my supervisor Amir Eghbali for all of his feedback and Axel Lindholm for developing the initial and basic idea with me. Big thanks to Marco Monzani for providing the opportunity to be a member of all of this, share his knowledge and spend such a good time together.

Secondly, I thank my examiner at Chalmers Henk Wymeersch for making this thesis possible, Rahul Devassy for his support and Luca Caltagirone for all the valuable advice.

It feels important for me to mention my home university UNIZAR (University of Zaragoza, Spain) where the basis of my knowledge was set and Chalmers University of Technology and Sweden where I could develop myself further. Tack så mycket.

Para finalizar, quiero agradecer todo el apoyo que siempre he encontrado en mi familia, a los cuales les debo todo. Y como no, a mi mitad. Gracias de corazón.

Eduardo Sesma Caselles, Gothenburg, December 2017

Contents

List of Figures

List of Tables

1

Introduction

This chapter describes the background about limiters, crest factor reduction and signal reconstruction which help to avoid saturation and distortion in audio signals. Furthermore, it specifies the scope and limitations of the thesis.

1.1 Background

In acoustic signal processing, we often encounter the need to limit the peak amplitude of the waveform to adapt the waveform itself to the dynamic range of the processing device. A typical case is that of sound amplifiers: in such applications, lack of more sophisticated signal limiting results in the signal being clipped, causing audible distortion [1]. The concept of limiting is any process by which a specified characteristic (usually amplitude) of the output of a device is prevented from exceeding a predetermined value.

Limiters have similar behaviour as compressors, but each one uses the concept of compression for different purposes. A compressor reduces gradually the signal level above a certain threshold. Limiters are extreme forms of compressors that prevent signals from exceeding certain limits [2].

In the Crest Factor Reduction (CFR) algorithm, the input signal is processed so that the output voltage is kept within predefined limits, yet the audible harmonic distortion caused by clipping is reduced to the minimum. CFR algorithms are based on analysing an amount of samples before their reproduction. The current systems for CFR are well known, but it still implies a certain latency in the signal propagation.

The tool that will be used in this thesis is an Artificial Neural Network (ANN). In simple words, ANNs are systems where some inputs are fed to the network and propagated through the neurons (middle nodes excited in different ways depending of the input signal) to generate an output. These systems are widely known to be trained in order to generate adaptive behaviours. In the training phase, the network can be guided with different methods trying to fit a target behaviour as close as possible.

1.2 Related Work

In general, the systems studied in prior art implement a look-ahead limitation of the signal, therefore all of these approaches include some delay. The system that is going to be developed in this thesis does not imply a delay line and, to best of our knowledge, it is a new model that has never been used for this application.

There are some studies in prediction of signals [3] in general and audio reconstruction [4] . A system able to predict future samples in an audio source could interact in the same way as the look-ahead application, but no evidence could be found in literature that this kind of method had been actually tested. Also, the outcome of this project will lead to a unique system that implies prediction and signal attenuation in the same step. Then, a nonlinear limiter with two states will be created, differentiating when attenuation should be applied and when not. Therefore, this differentiation should be equivalent to prediction. Whenever prediction is applied with the purpose to identify whether the following samples would cause clip or not.

1.3 Purpose

The aim of this thesis is to implement an innovative CFR algorithm to process digital sound from voice and music sources. The proposed idea is to make use of an ANN to identify a prediction model of the signal. This predictive model can be applied to implement a CFR algorithm. The main ambition is to study if this method could work properly for the aforementioned purpose since it has not been possible to find similar approaches in literature.

When programming real-time devices involving sound applications, latency is critical. The instantaneous response of the whole system is compromised. If one block adds some delay to the audio flow, the time needed to wait between the signal injection to the system and its extraction at the output increases. The proposed system could eliminate the problem of latency in the audio signal flow generated in the current look-ahead limiter, probably the computational load could be decreased and it could also open new pathways leading to different implementation of a number of audio processing applications.

1.4 Scope and Limitations

This report will outline some basic knowledge about limiters and more specifically digital sound limitation systems. It will also report the tools used for the new approach as Neural Networks and the “Gravity model” (applied for this thesis and explained in next chapters).

The aim is to find an intelligent/adaptive model that can be applied to input signals so that the output will be attenuated in advance before its amplitude goes above a predefined threshold level.

The best model and parameter extraction of the input signals in order to feed the ANN and the best topologies have to be found. The next step is to find a coherent data set and generate the best training and validation sets (or at least data with enough content for the general purposes of the real application). The proposed system should work for any kind of sound input sources or types of music in a defined frequency range.

Different training methods for the ANN will be found and evaluated during this thesis. The training methods will be based on Genetic Algorithms (GA) and different types of fitness functions with different penalties will be explained and discussed in future chapters.

The results will be simulated using MATLAB and high level simulation will be made for the proposed method as well some existing techniques within audio community. Once the implementation is ready, it has to be tested and compared with previous solutions. Such factors as no clipping in the output signal or minimum distortion between input and output should be targeted.

This thesis will provide the information to know the methods and parameters involved in the realisation of the system. However, due to the stochasticity of the results and the final subjective evaluation of the solutions (listening test), the ambition is not to provide the absolutely best configuration for the system implementation. Different solutions could be valid for different listeners.
Nevertheless, the system has to show the correct behaviour of the methods and the implementation.

The study will not cover comparisons with the real ISVPL (Inter-Sample Voltage Peak Limiting, zero-overshoot digitally implemented limiter) developed and applied at Labgruppen AB, but it will be compared with a similar implementation based on a look-ahead limiter. This project could lead to a big number of possible combinations of parameters and different implementations or topologies. Due to time constraints, after a preliminary research, the range of possible combinations was narrowed as much as possible to have a good enough knowledge of the problem and the possible outcomes. For example, only one hidden layer is expected in the ANN. In the same way, the choice of using GA instead of back-propagation algorithms to train the network is explained in this document, but it has not been implemented and compared. Improvements for real-time processors will be explained theoretically based on complexity, but no implementation has been created for the purpose of testing the algorithms in those processors.

2

Theory

This chapter describes the theory needed to understand the solution to the problem formulated in the previous chapter. It is split into three sections: First section, about limiters where the concept of limitation and CFR is explained. Second, about ANNs since it will be the main tool used to calculate the gain control for the limiter algorithm. At the end, GAs as the process to train the neural network in order to obtain its correct behaviour.

2.1 Limiters

A limiter is a tool that ensures that the output signal does not exceed a specified amplitude level. It is usually implemented based on a compressor algorithm, but in a form that prevents the signal from exceeding a threshold. This is very important in order to protect the system going above its limits. If a sound amplifier tries to produce more power than it can produce, the signal will be distorted. Also, amplifiers set limitations for the output power in order to do not burn the loudspeaker.

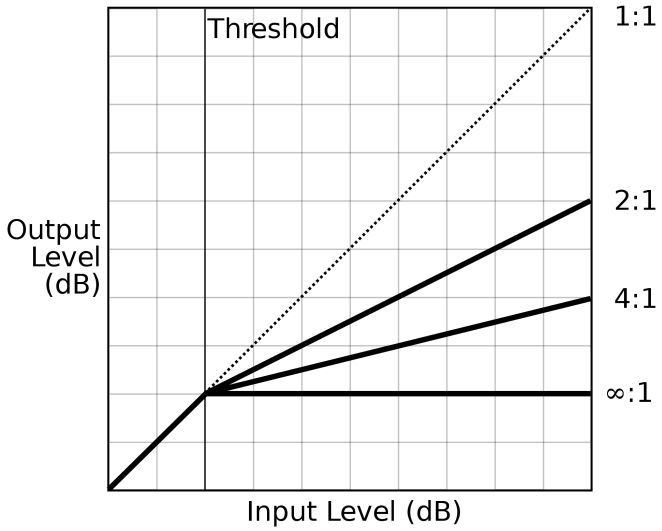


Figure 2.1: Compressor-limiter curve showing the input-output relation of the system. From a threshold level, compressors will compress the signal with some ratio, while limiters (hard limiter) have infinite ratio since every input above the threshold will derive in an output equal to the threshold. [5]

The digital model of the limiter follows the analog circuit that allows signals below

a specified input power or level to pass unaffected while attenuating the peaks of stronger parts of the signal envelope that exceeds this limit. Current technology in digital signal processing allows sophisticated algorithms in sound limiters with much improved precision, no added noise and considerably reduced distortion as compared to analog designs (due to analog components among other reasons).

Just as an introduction to understand better how a limiter works, the concept of compression will be explained. Basically, compression reduces the dynamic range of the input by bringing down the level of the loudest parts. Meaning the loud and quiet parts are now closer together in volume and the natural volume variations are less obvious.

As can be seen in Figure 2.1, there are some parameters to determine how the compressor works. Threshold sets the amplitude level where the compressor starts to act, which means that from that level the input is louder than desired. Every input above the threshold will be compressed with some ratio. Ratio parameter sets how much the input signal will be compressed. In other words, the compression ratio is the relation between the input and the output when the input goes above the threshold. Higher ratio means more compression. If the ratio is 1:1 there is no compression, but if for example, the ratio is 4:1 that means for every 4 dB of sound that goes over the threshold, the output brings 1 dB above the threshold.

$$x_{out_{dB}}[n] = \begin{cases} \frac{x_{in_{dB}}[n] - \text{threshold}_{dB}}{\text{ratio}} + \text{threshold}_{dB}, & \text{if } x_{in_{dB}}[n] > \text{threshold}_{dB} \\ x_{in_{dB}}[n], & \text{otherwise} \end{cases} \tag{2.1}$$

Attack time is the time to act on the input when it is above the threshold, while the release time is the time to let the signal return to the input level after it falls below the threshold. Figure 2.2 shows the two types of transitions to the compression state, one is called soft-knee where the transition is progressive and the hard-knee where the change goes straight from the linear state to the compression.

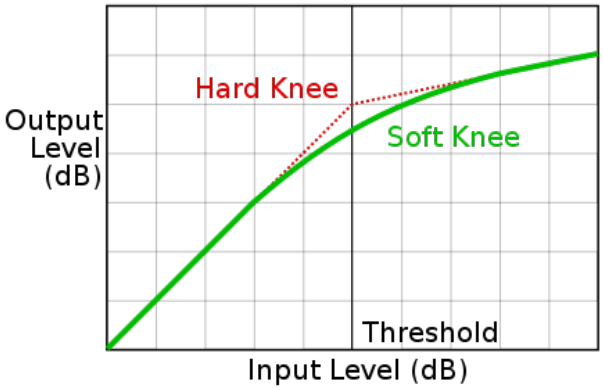


Figure 2.2: Difference between soft-knee and soft-knee limiters/compressors.[6]

Simple limiters add some attack and release time reaction to the dynamics (amplitude variations) of the input signal (similar process to a low pass filter), as can be seen in Figure 2.3.

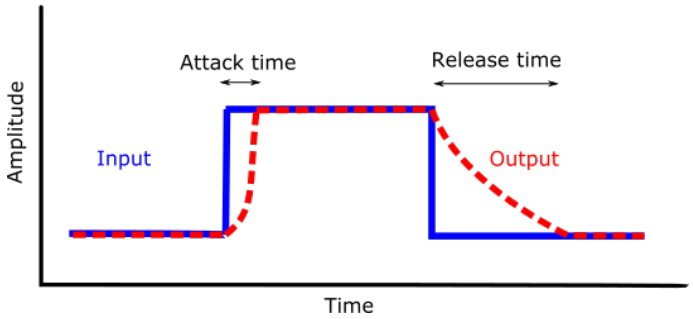


Figure 2.3: Time reaction to the dynamic change of the input signal due to the attack and release time.

The resulting signal (x_{peak} or envelope) with the threshold is compared in order to apply linear behaviour (ratio 1, $1:1$) or compression (limitation, ratio a, $a:1$ with $a > 1$). One of the simpler implementations of an envelope detector is to use a first order filter over the absolute value of the input signal $x(n)$ (since both peaks, positive and negative want to be compressed). The mathematical expression, taking AT as attack coefficient and RT as release coefficient works as follows (where the coefficients are precalculated as the percentage of the current sample that needs to be taken in order to reach the desired one in the time specified by the attack or release time [1]):

$$x_{peak}[n] = \begin{cases} (1-AT)x_{peak}[n-1] + AT|x[n]|, & \text{if } |x| > x_{peak}[n-1] \\ (1-RT)x_{peak}[n-1] + RT|x[n]|, & \text{otherwise} \end{cases} \tag{2.2}$$

The attack time needs to be very fast on a limiter, so AT must be close to 1 (or 1 directly when attack time is 0). These settings can easily colour the sound of the signal and also cause audible "pumping" if the signal goes over the limit with very fast transitions.

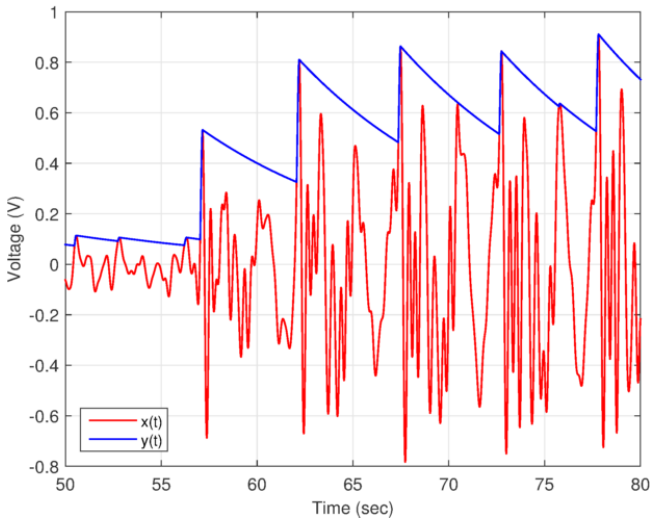


Figure 2.4: Envelope estimator with short attack time and larger release. The signal in red is the input and in blue is the envelope. Envelope follows the signal quickly when the input increase due to the short attack time and goes down slowly when the input decrease due to large release time.

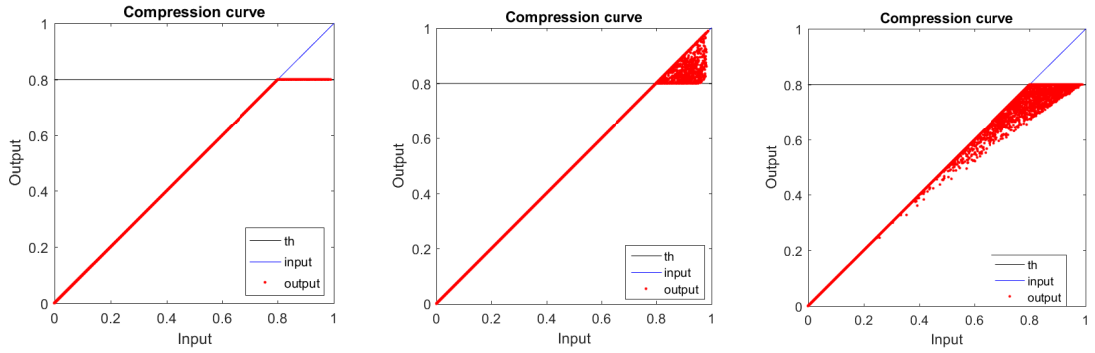


Figure 2.5: Input-Output samples relation. Each red dot maps the relation between the value of an input sample and the value of that sample in the output when it is attenuated by the limiter. On the left, the ideal limiter response with no attack and release. In the middle, with 1 ms attack time and no release. On the right, the opposite procedure with no attack and 1 ms release time.

In Figure 2.5, as can be seen on the left, the input/output relation for all the samples (each red dot) maps exactly the ideal behaviour of the limiter when there is no attack and release time. In the middle figure with some attack time, the limiter does not reach on time to attenuate the signal and that generates some clipping above the threshold. That is the reason of the red dots above the threshold: when the input signal is above it, the limiter sometimes is not able to attenuate it and, hence the output is still above. In figure on the right with some release time, the limiter is still applying attenuation after the signal is clipping and some samples receive more attenuation than it is needed. Short times result in higher concentration of samples close to the ideal case. Longer times will result in samples farther from the ideal case. Set different values of attack and release times will generate a non-ideal behaviour, but the result will be better in terms of saving the dynamic relation of the signal.

In order to avoid these effects, look-ahead limiters (and CFR algorithms) apply some delay to the input and preprocess the compression starting to attenuate the signal before it is going to clip. This behaviour is similar to Figure 2.7.

Look-ahead limiters are very important in digital application since one is able to catch peaks even as short as a sample. Delay gives time to the algorithm to know exactly when the peak is going to be in the future and the algorithm is ready to apply the required attenuation when the delayed input comes.

2.2 Crest Factor Reduction

CFR algorithms are used to reduce Peak to Average Power Ratio (PAPR) of a signal. These systems are widely used in telecommunications for OFDM systems [8], where the small distortion in the low noise amplifiers could cause significant noise in the different frequency bands. Therefore these peaks need to be attenuated and at the same time try to keep the maximum RMS power in the signal for a better

Signal to Noise Ratio (SNR). In this thesis similar approach is taken, working with lower frequency bands and trying to avoid audible distortions and over-excursion in speakers (the loudspeaker membrane being pushed outside the acceptable excursion range with consequent risk of permanent damage). Also, higher power demand in the sound amplifier than it can handle could create distortions.

This system does not have to be confused with reconstruction algorithms [4]. CFR algorithms prevent clipping while in reconstruction algorithms the input to the system is already clipped and the system tries to predict and restore the shape above the threshold. Usually, these systems are used for telecommunication purposes.

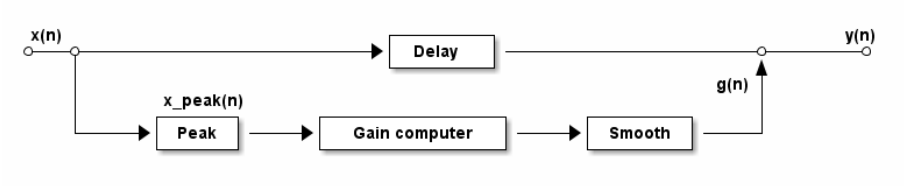


Figure 2.6: Blocks of a basic look-ahead limiter.

A basic model for a look-ahead limiter is shown in Figure 2.6. Input signal $x(n)$ is delayed and processed in parallel. This processing branch includes a peak (envelope) detector, a gain computer and a gain smoother block. At the end, the gain $g(n)$ is applied to the delayed signal. Once the envelope of the input is calculated, the gain computer block calculates the gain needed to attenuate the signal when it clips. The formulas in logarithmic domain are as follow:

$$g_{db}(n) = max(0, threshold_{db} - x_{peak_{dB}}(n)) \tag{2.3}$$

and instead of using a soft-knee implementation, the gain signal is filtered over the time at the gain smoother block. This block acts similar to the peak detector being, in most of the cases, a first or second order low pass filter. This block avoids the discontinuities in the gain, having soft transitions in the attenuation of the signal and trying to keep the dynamic range of the processed samples between them [2].

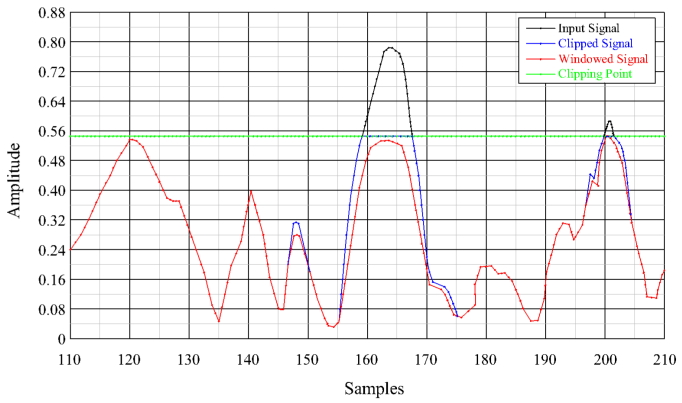


Figure 2.7: CFR algorithm. An input signal (black) clips over a threshold level (blue) and it is reshaped with low distortion under the threshold avoiding saturation (red).

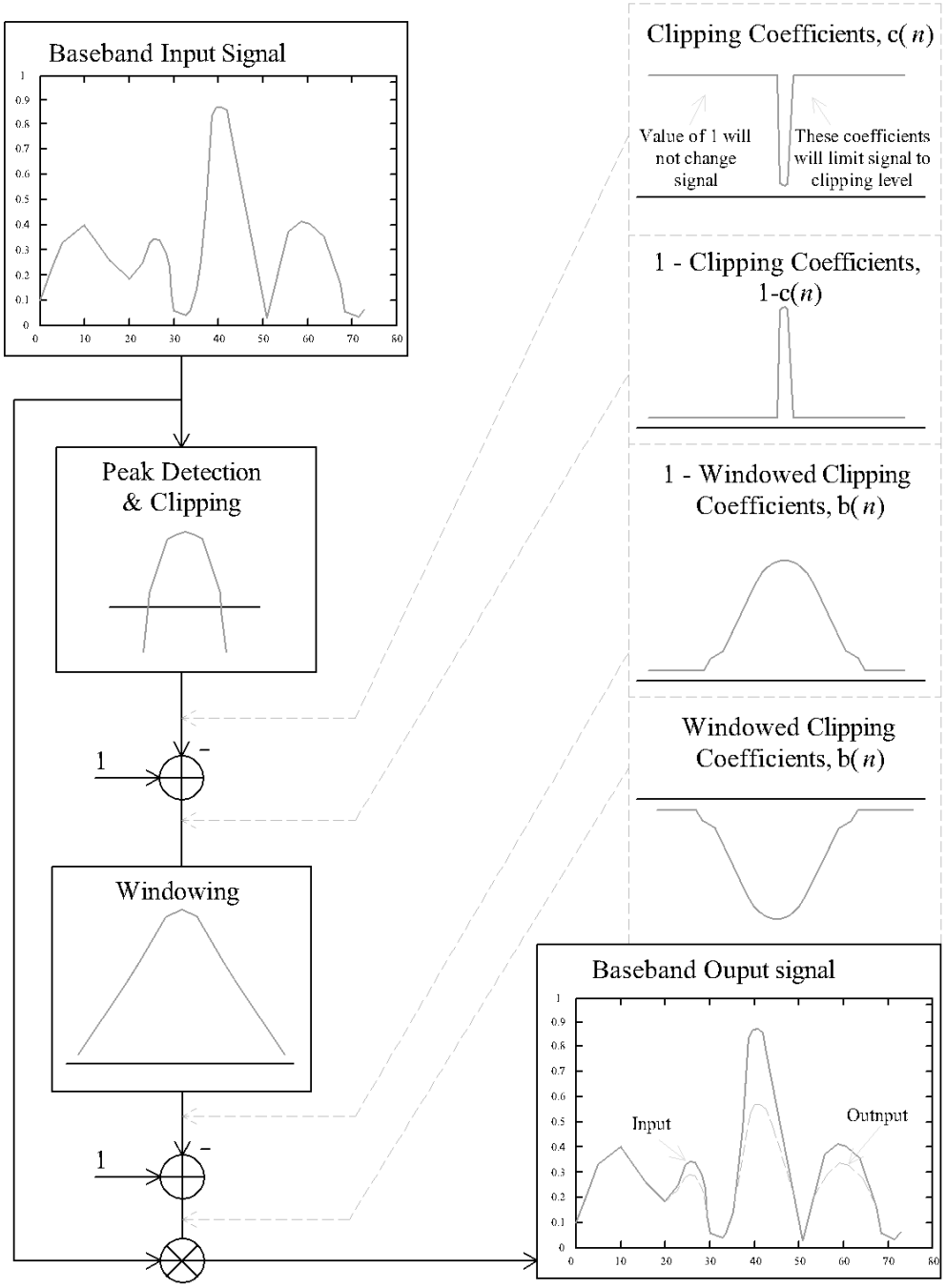


Figure 2.8: Example of a high level system block diagram of Peak Windowing Algorithm.

Another similar and also effective system is the Peak Windowing Algorithm. The block diagram and signal path is represented in Figure 2.8 and it is descriptive enough for the good understanding of both systems.

Here a frame (a set of input signal samples) is read and a peak search algorithm creates the signal $c(n)$ (Clipping Coefficients) with the samples above the threshold.

Later this signal is convolved with a window that stretches the sharp edges of $c(n)$. The result of the convolution is the gain that is applied for each sample [7].
The result is similar to the look-ahead limiter in the sense that thanks to the windowing the signal is starting to be attenuated softly before the peak. This is equivalent to the smoother gain block in the look-ahead.

In order to analyse the outcome of this systems, Figure 2.7 shows the signals that take part when running the algorithm. The input signal in black triggers x_{peak} signal with the samples above the threshold at 0.5 amplitude. The gain reduction is computed following equation 2.3 and later is filtered/stretched in time. The result is used to apply the attenuation over the input signal and it generates the output in red.

As can be seen clearly, the algorithm starts to attenuate the input signal before it is clipping and the reconstruction of the signal below the threshold keeps the dynamic range for these samples inside the new range from 0 to the threshold.

2.3 Artificial Neural Networks

ANNs are structures which try to imitate the behaviour of biological neural networks present in the brains of animals and humans. Usually, the capacity of the brain is correlated with its size, so that bigger brains, and, transitively, larger neural networks are able to execute more complex tasks

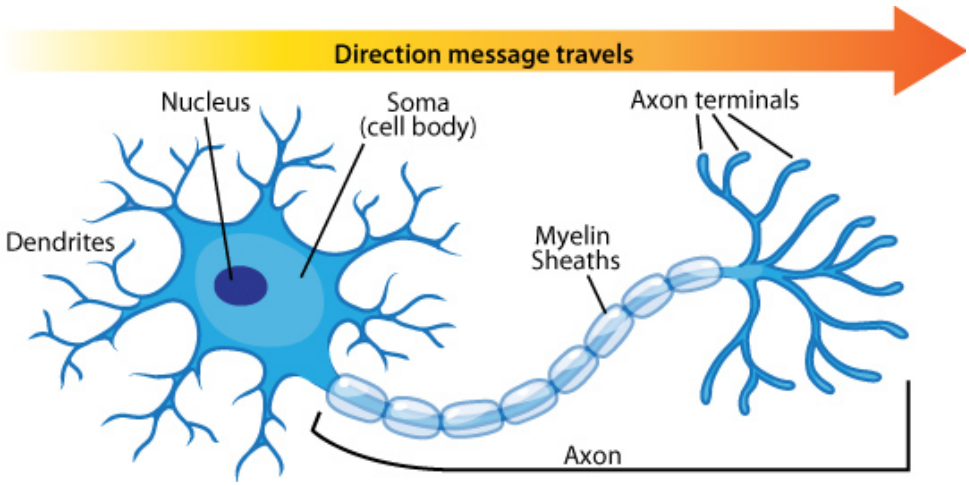


Figure 2.9: Schematic of a biological neuron.[9]

Biological neurons are a cell type of the nervous system whose main property is the electric excitability of its membrane. It is specialised in the reception of stimulus and transfer the nervous impulse between them.

As can be seen in Figure 2.9, there are several morphological characteristics to sustain its functions: Nucleus; one or several short prolongations that generally transmit impulses to the nucleus, called dendrites; and a long filament called axon

that drives the impulses from the nucleus to the next neurons. This is the output of the neuron. After some distance, the axon splits up into several smaller terminals which end on synapses. Input signals are received through the dendrites with an electric potential or spike. After reaching the axon, the synapses release transmitter substances to the next dendrites, as can be seen in Figure 2.10.

The synapses that connect the neurons can inhibit the signal or excite it. Synapses determine the computation carried out by the neural network: if synapses are added or removed, the network architecture and its computations will change. Synapses can be seen as information storage and also, it keeps another properties as the timing between signals that may play an important role. A very important feature of the human brain is the ability to memorise and learn, which means store information about past events and to modify the behaviour as a result of those experiences [10].

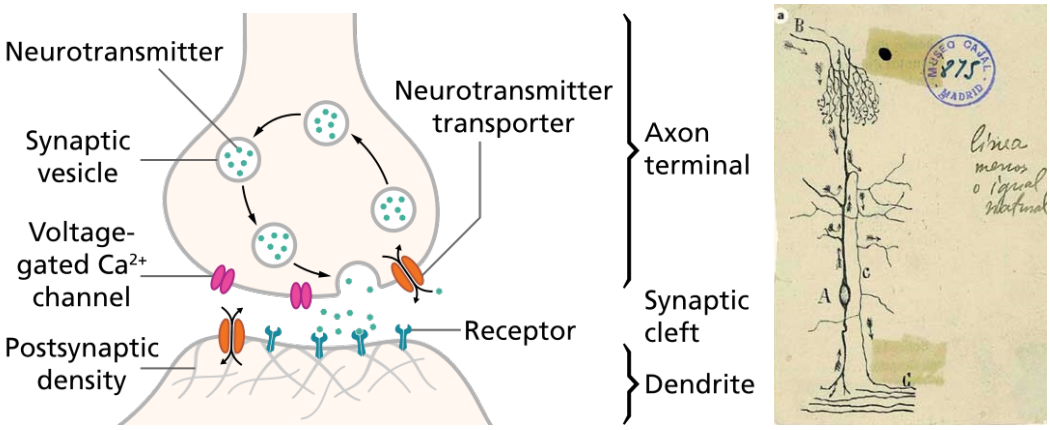


Figure 2.10: "Synapse" (from the Greek "synapsis", means "conjunction") [11]. Santiago Ramon y Cajal proposed that neurons are not continuous throughout the body, yet still communicate with each other. Synapses are the means by which the neurons pass signals [12].

Memory provides to humans and animals the ability to weigh actions from the past in order to assess the probable outcome of a situation, while learning allows the modification of the behaviour.

2.3.1 Implementation

The computational model is based on a set of simple neural units (artificial neurons, called perceptrons) interconnected through neuronal layers. Each neural unit is connected to the neurons in the next layer. This connection is weighted to increase or decrease the activation level among them.

McCulloch-Pitss model of a neuron [14] describes the simple information-processing unit fundamental for the operation of a neural network. Figure 2.11 shows the model of a neuron and three basic elements could be identified: Synapses as the weights for each input, accumulator as the summation of the transfer function and the activation function.

Synapses or connecting links are characterised by a weight where the signal x_i at the input synapse i, connected to neuron j, is multiplied by the synaptic weight $w_{i,j}$. The accumulator is used to sum the input signals, weighted by the respective synapse; this operation constitutes a linear classifier.

The activation function limits the amplitude of the output and it can be called "squashing function" or "sigmoid function" as well. It squashes the amplitude range of the output signal to some finite value [15].

Mathematically, the calculation that describes the output of a neuron j, from inputs $x_i, ..., x_n$ becomes:

$$o_j = \varphi\left(\sum_{i=1}^{n} w_{i,j} x_i \right) \tag{2.4}$$

Usually, an extra input to the neuron is applied in terms of bias, but it is simpler just to add this term as an extra input to the accumulator.

There are several types of activation functions, but in general, they are modelled by taking value 1 if the sum of the accumulator exceeds some threshold (represented in Figure 2.11 as θ_j) or 0 otherwise (step function). In that case, the bias term determines the propensity of the neuron to fire (to be activated) in the absence of any input [10]. However, this model does not always fit the purposes of the desired application and it can be a better method to map the response into a finite range of values. In this case, for example, the logistic sigmoid function, among others, could be used. There are several studies to determine how these activation functions work and which has better performance [17].

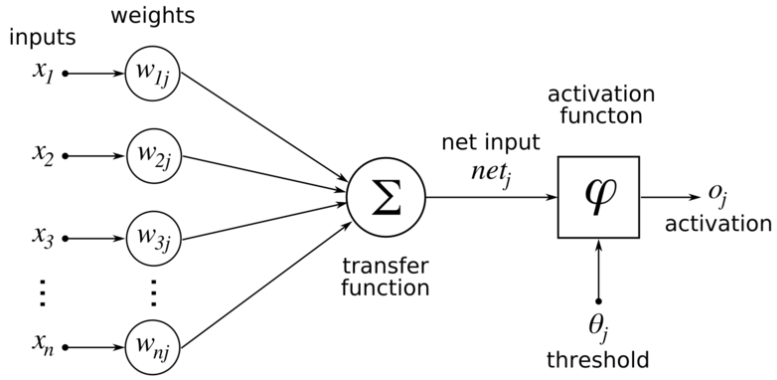


Figure 2.11: Perceptron. An artificial neuron, a linear classifier. All inputs are weighted and sum, the result is evaluated in the activation function and it derives in the activation output.[18]

The Hebbian learning rule describes the basis of learning algorithm: " If a neuron j receives an input from neuron i and if both neurons are strongly active at the same time, then increase the weight $w_{i,j}$ (i.e. the strength of the connection between i

and j)" [13]. This derives:

$$\Delta w_{i,j} \sim \eta o_i a_j \tag{2.5}$$

with $\Delta w_{i,j}$ as the adjustment of the weight from i to j, proportional to the output o_i of the previous neuron, the activation a_j of the successor neuron and a constant η, i.e. the learning rate [16].

There are several topologies for neural networks. The most common and simple are the Feed-Forward Neural Networks (FFNN) topology. This topology has a first layer with inputs elements, one or several hidden layers and an output layer. In each layer (except the first one that there are only input values), each element corresponds with an artificial neuron, that computes the inputs from the previous layer and propagates its output to the next layer. Figure 2.12 is an example of FFNN. This topology only contains connections in forward direction.

Another basic topology is given for the Recurrent Neural Networks (RNN), which contains connections in forward direction and also backwards between layers, having feedback.

The topology used and described in this thesis is a FFNN.

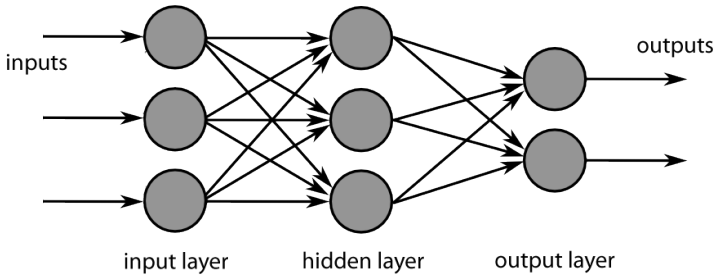


Figure 2.12: Artificial Neural Network feed-forward example with 3 inputs, 1 hidden layer with 3 neurons and 2 outputs.

Putting together the concept of the artificial neuron (Figure 2.11) and neural network (Figure 2.12), each connection is associated with a weight, corresponding to the synapses.
There are also several methods for training ANN, but most of them are divided between supervised learning and unsupervised learning. Supervised learning provides an input with a target output. This means that the weights need to converge to some values which fit the target output when the input goes through the network. Unsupervised methods do not provide a target output for a given input.

The most common method is back-propagation (used for supervised learning). This algorithm uses a propagation-adaption cycle in two phases. Once the input is processed by the network, the output signal is compared with the target output and the error is calculated and propagated backwards layer per layer for the output of each artificial neuron. Neurons in the hidden layer only get a fraction of the total

error, based on an approximation of the relative contribution that each neuron has provided to the original output. This process is repeated until all neurons in every layer receive the error signal that describes the relative contribution to the total error.

The importance of this process remains in, when training the network, neurons in the hidden layer organise themselves learning to recognise different characteristics of the total input's space. After training, arbitrary patterns are given to the input. These patterns go into the network and the neurons in the hidden layers will provide an active output if the new input contains a similar pattern to the characteristic that each neuron has learnt to recognise during its training [15].

Then, the training procedure follows the next steps:

1. Initialisation of the weights with small random values
2. Generate a random permutation of the input-output pairs
3. Compute the output of the network and its error
4. Compute $\Delta w_{i,j}$ for each connection starting from the output layer to the input, following equation 2.5
5. Update the weights $w_{i,j}$ adding its $\Delta w_{i,j}$
6. Repeat from step 2 until the errors drop to the desired level.

Over-fitting must be avoided when training neural networks. Over-fitting means that the behaviour of the ANN is shaped very much around the specific training data set, and therefore cannot equally well manage with processing of other data. What happens here is that the network is over-trained for that data set. Then it is not able to predict the results for new data.

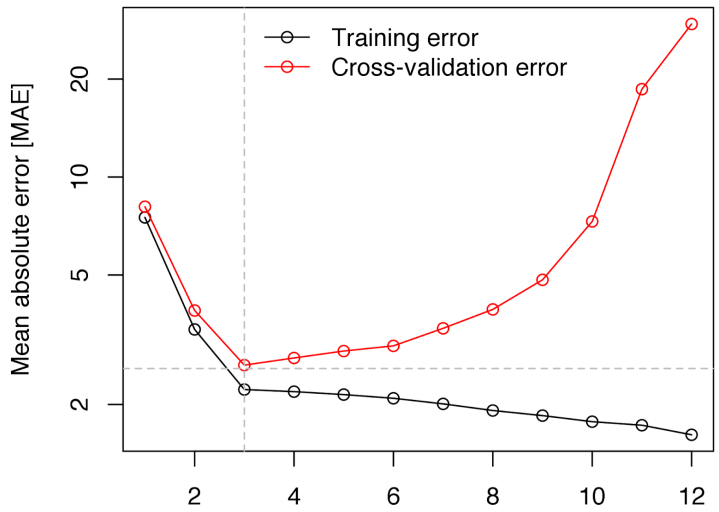


Figure 2.13: Over-fitting example during training. The vertical dotted grey line represents the moment where an early stopping has to be done in the training since from this point, the error for new data is going to increase while the network is over-trained for the training set.

A very important step in order to train the network without over-fitting is that the used data set has to be split between a training set and a validation set. Usually, the training set is a big percentage of the set, while the validation is much smaller. Usually, the proportion is 70-30 %.

Also, the data set has to be sufficiently large to provide a consistent training, but these numbers are much more difficult to predict. It is always a better option to have as larger as possible data sets.

During training, the algorithm needs to check the error level over the total data set. This has to be done for both data sets. Usually, the error for both data sets start decreasing, but the problem starts when the error for the validation set starts to increase while the error of the training data set is still decreasing. This means over-fitting. In that moment of the training, the network is starting to be over-trained for the training set and it is not better for general data anymore. This can be seen in Figure 2.13.

ANNs are computational structures where several learning methods can be applied, but in addition, also various stochastic optimisation algorithms (like GAs, for example) can be applied for the training.

2.3.2 Purpose of ANNs

ANNs have a very wide range of applications. They could be applied to function approximation, time series prediction, classification and pattern recognition, filtering, robotics and control. In very general and abstract view, it can be seen as a "black box" that learns some behaviours and from one input generates an output based on that behaviour.

In this thesis, the purpose of using ANNs is to try to imitate the behaviour of the audio limiter. ANNs have an incredible value reading inputs with very different spaces and abstraction levels. As will be explained in the next chapter, some extracted features from the audio signal beyond the usual amplitude or power in time or frequency domain will be taken as an input.
ANNs will be the tool to understand these features from the signal and it will be trained to obtain similar results than a look-ahead limiter without the necessity of delaying the signal, but attenuating it and anticipating future distortions with some kind of prediction (thanks to the implicit information in the extracted features).

Instead of copying the behaviour of a limiter with a supervised learning method, the aim is to train the network with unsupervised learning and create a new tool, more flexible and adaptive than current implementations. Therefore, ANNs could provide a non-linear control dynamics system that adapts to the audio instead of a deterministic system following the same fix patterns.

2.4 Genetic Algorithms

As a subclass of evolutionary algorithms, a GA is an optimisation tool inspired by processes similar to those that occur during biological evolution.
Evolutionary algorithms inherit some terminology from biology, but the simulations behind are considerably simplified in comparison to the real processes involved in the biological evolution.

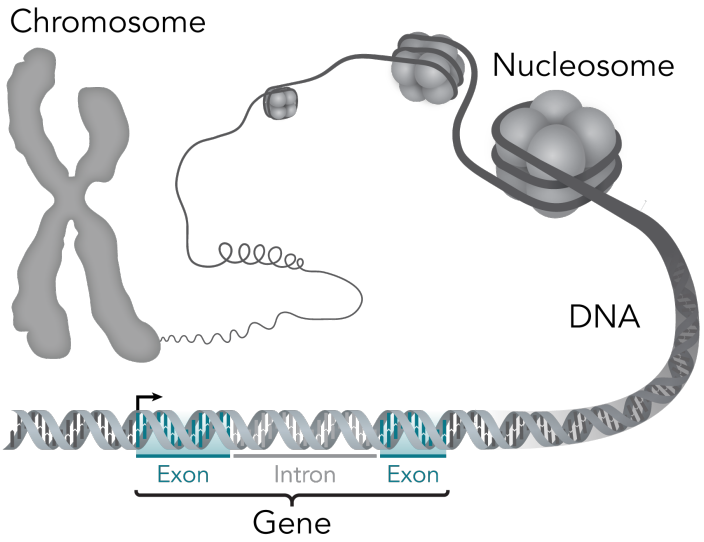


Figure 2.14: The nucleus of the cell contains pairs of chromosomes that store genetic information in the DNA. DNA is segmented in genes.[20]

Therefore, in order to understand how the algorithms work, we first revise some concepts on biological evolution process.

Darwin's theory of evolution provides the concept of progressive and hereditary change in the individuals of a species. The hereditary change needs some tools to transfer the information from the individuals of one generation to the next. Based on the DNA molecule, this information is stored and transferred to next generations. Each individual has its own genome present in each cell of its organism, combining several chromosomes to organise the formation of the DNA. The cells of an individual keep the DNA in a structured manner, this structure is called chromosome.

A human cell has 23 pairs of chromosomes (46 in total), half of them come from the father and the other half from the mother. DNA is made out of two long, twisted strands that contain complementary genetic information and a gene is a segment of DNA [21]. Figure 2.14 illustrates the structure of the cells and chromosomes.

Most organisms evolve by means of two primary processes: natural selection and sexual reproduction. The first determines which members of the population survive and reproduce, and the second ensures mixing and recombination among the genes

of their offspring. When sperm and ova fuse, matching chromosomes line up with one another and then crossover partway along their length, thus swapping genetic material [24]. DNA is passed down from parents to children and it confers a trait to the offspring. This mixing allows creatures to evolve much more rapidly than they would if each offspring simply contained a copy of the genes of a single parent, modified occasionally by mutation. Mutation is obtained through generations due to the adaption to new environments, trends or necessities. In nature, each species needs to adapt to a complicated and changing environment in order to maximise the likelihood of its survival. At the same time that the individuals evolve in different ways, the strongest ones get more probability to be selected for reproduction, generating new individuals that, in principle, should be better to adapt and survive.

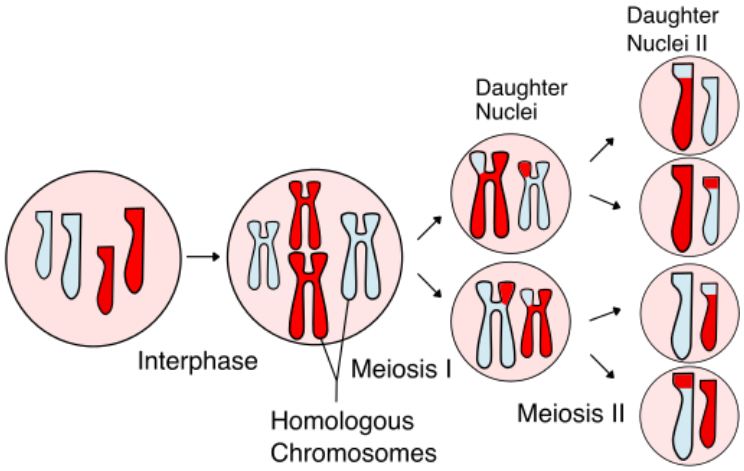


Figure 2.15: Analogy between the biological concept of reproduction and the algorithm. Crossover is the fundamental mechanism of genetic rearrangement for both real organisms and GAs. Chromosomes line up and then swap the portions of their genetic code beyond a crossover point.[25]

As a summary, the most important and general facts that conceive evolution to an initial population are natural selection, reproduction and mutation. Natural selection means that if an organism does not have enough fitness, such as recognising a predator and fleeing, it dies. Reproduction is done by mixing the genes of the selected individuals and mutation is an adaptation acquired through generations.

2.4.1 Implementation

GAs were invented by Holland [24] to mimic some of the processes of natural evolution and selection. An implementation of a GA is based on the previous biological principles. It begins with an initial population that evolves to new individuals after some generations. These new individuals fit better some requirements that optimise the outcome of our system, making it work better and behaving in the way we aim.

Typically, the initial population is random. These structures are evaluated and reproductive opportunities are given with higher probability to the chromosomes

which represent a better solution to the target problem. In general terms, a GA is a population-based model that selects and recombines operators to generate new sample points in a search space [21].

GAs need to specify a search space and identify the heuristic function that evaluates the fitness of each individual. This heuristic function is also called fitness function and the individuals with higher fitness will represent the better solution to the problem. Having higher fitness means to get a higher probability to be selected for reproduction. Not only a correct and efficient implementation of the algorithm will provide a better performance during training (simulating the evolution of the individuals generation after generation), but also it is important to consider the best fitness function to evaluate the individuals and how the data is encoded in the chromosomes in order to be meaningful for the system.

Each chromosome is a set of genes that represent the solution to the problem. The binary alphabet is often used to represent the value of each gene, but it depends on the application. For example, in this thesis, each gene will correspond with one of the weights of the neural network. Therefore, the alphabet will be the real numbers in a predefined range. To evolve classifier rules that solve a particular problem, the initial population is created randomly and evaluated with the fitness function. Once all the individuals (chromosomes) are evaluated, they are ranked according to the fitness obtained.

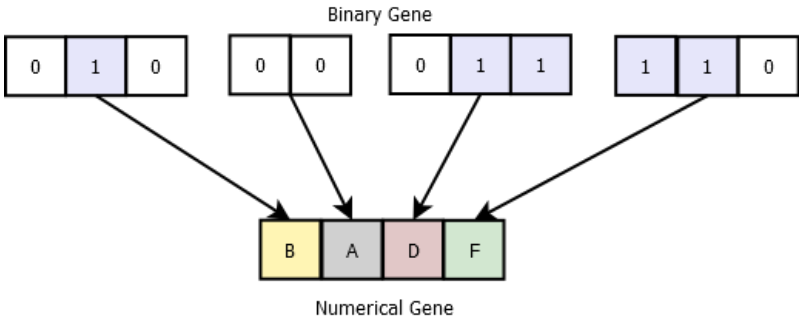


Figure 2.16: Chromosome example for a GA. In this case, a binary stream could be the representation of the real components used in the fitness function. This abstraction is very common and opens the possibility to get much more diverse results since the recombination of the binary stream can produce totally different outcomes for each represented value.

As an easy example, if the purpose of the GA is to approximate a behaviour or a trend described by a n-th order polynomial in some given system, the fitness function will try to find the coefficients that best fit the system. Therefore, this fitness function would be a polynomial with the same order as the number of genes. The genes of the chromosome would be the coefficients of the polynomial and the chromosome with the combination of genes that best fit this behaviour will have highest fitness [22].

This example could be more complicated if, for example, the value of the genes have a different meaning, as can be the case of Linear Genetic Programming [10]. In this case, the function could be a combination of different operations for different variables in a non-linear system. As can be seen in the example of Figure 2.16, chromosomes would be a binary representation of different terms. The first n genes could be a decimal number, the next ones an operator, etc. Some rules about how to combine operators and variables should be implemented, but that is out of the scope of the basic introduction of GAs for this thesis.

After the evaluation of the individuals, chromosomes for the current generation have to be selected for reproduction. If the size of the population is N, the selection procedure obtains two parents chromosomes, based on their fitness values in order to generate two offspring for the new population. This process is carried out until new N individuals are generated for the new generation.

There are different ways to simulate the natural selection, some of them are "roulette wheel" or "tournament selection". Roulette wheel method maps the population onto a roulette wheel, where each individual is represented by a space that proportionally corresponds to its fitness, as can be seen in Figure 2.17. By repeatedly spinning the roulette wheel, the individuals are chosen [21]. Tournament selection method resembles the typical contests between pairs of individuals (and it is the one used for this thesis). As described in Figure 2.17, in this case, k individuals are randomly chosen, two of them "fight" and the one with higher fitness has a predefined higher probability of success. The winner repeats the process with the next one. That makes $k-1$ times in total to have the one that remains for the selection.

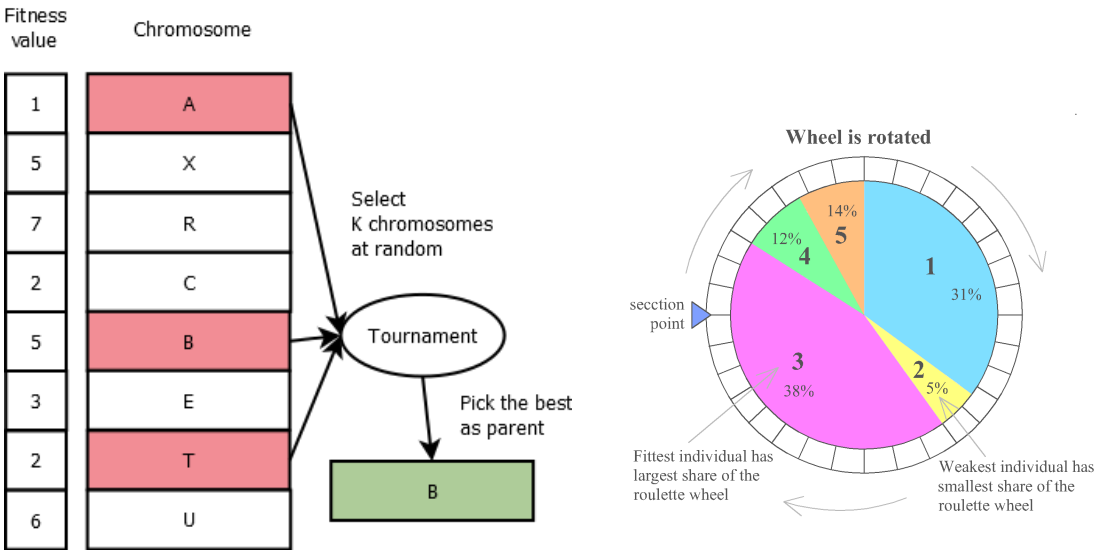


Figure 2.17: Selection methods. On the left, the illustration of the tournament selection method. On the right, the illustration of the roulette wheel selection method.

The GA exploits the "target" regions of the solution space because successive generations of reproduction and crossover produce an increasing number of compatible

genes in those regions. The algorithm favours the fittest chromosomes as parents, and so above-average (which fall in the target regions) will have more offspring in the next generation [24].

Once a pair of chromosomes has been selected, the way to simulate reproduction in the algorithm is called crossover and it is one of the more important steps in the algorithm. This simple procedure combines the genes from two individuals, cutting the chromosomes at a random point and assembling the first part of the first chromosome with the second part with the second and viceversa, as Figure 2.18 shows. If the length of the chromosome has to be the same for all the individuals, the same crossover point is taken and the genes from the parents are interchanged.

There is also a predefined probability to do crossover. It tells if the individuals will reproduce offspring to be replaced or if the parents will pass ("survive") to the next generation.

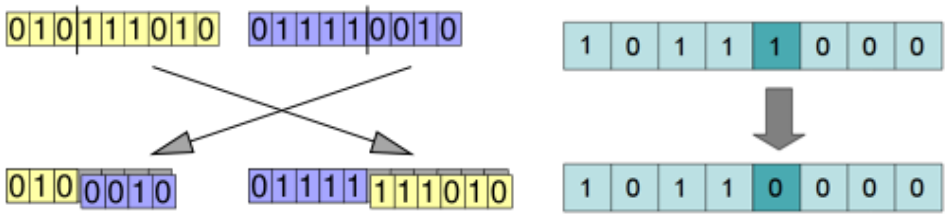


Figure 2.18: On the left, an illustration of reproduction carried out by crossover method. On the right, the adaptation process carried out by mutation method. [26]

After combining the chromosomes of the parents, mutation to each gene of the offspring is applied with a predefined probability. If all the parents have the same values for particular positions, then all future offspring will have the same value at this position.

To sum up the whole algorithm, first of all, a random initial population is created. From this point the next steps are repeated until the end:

1. Evaluate fitness of current population
2. Select chromosomes based on that fitness for reproduction
3. Perform crossover and mutation to give a new improved population

2.4.2 Purpose and Benefits of GAs

GAs encode solutions in data structures and, applying recombination, it tries to converge to the structure that best fit the solution to the problem. GAs are often used for optimisation purposes, but the range of applications is quite broad.

As have been explained in section 2.3, ANNs need coefficients to weigh each input/output of the neurons. In this system, the genes are the coefficients for the network.

As can be seen in the previous section, there are different ways to train the network: Supervised, unsupervised or reinforcement learning. Supervised learning model assumes the availability of a teacher or a solution for the problem and propagates the error to the neurons, while, unsupervised learning model identifies the pattern class information heuristically [23]. In this thesis, reinforcement learning is used and the ANN learns through trial and error interactions with its environment (reward/penalty assignment). Here is where GA and its fitness function definition are important.

Also, for this system is preferred to have a short number of neurons. That makes a better use of GA since very long chromosomes will result in very long training.

One thing that is striking about GA is the richness of how it computes the possible solutions. Usually, what may seem like simple changes is the algorithm sometimes result in surprising kind of totally different behaviours [21]. Therefore, GA is good to avoid the problem of falling in a local minima when converging to a solution. Thanks to mutation and crossover steps (and the probabilities for each process), the solution could jump out of a local minima and stay closer to the global minima solution [27].

The selection procedure should not be fully deterministic, then it should not always favour the fittest individuals. Such procedure would easily lead to stagnation and an individual that happens to be better than the others, but still far from the global optimum may come to dominate the population. In some cases, a less fit individual may contain a sequence of genes that will generate a highly fit individual when combined with genetic material from another individual [10].

3
Methods

This chapter gives a description of the model created for the CFR system based on ANN, the different approaches, strategies and the implementation using the theory presented in the previous chapter.

3.1 Gravity Model

The "Gravity Model" is an important part of this work. This model describes the input signal's samples as particles with their own velocity, acceleration and gravity factor. This model has been created for this thesis in order to extrapolate the meaning of the input, providing new features extracted from itself. For sound processing applications (and signal processing in general), the concepts of amplitude or power in time and frequency domains are always the relevant features and space of the signal (being the space of the signal a two dimensional representation of the amplitude or power versus time or frequency).

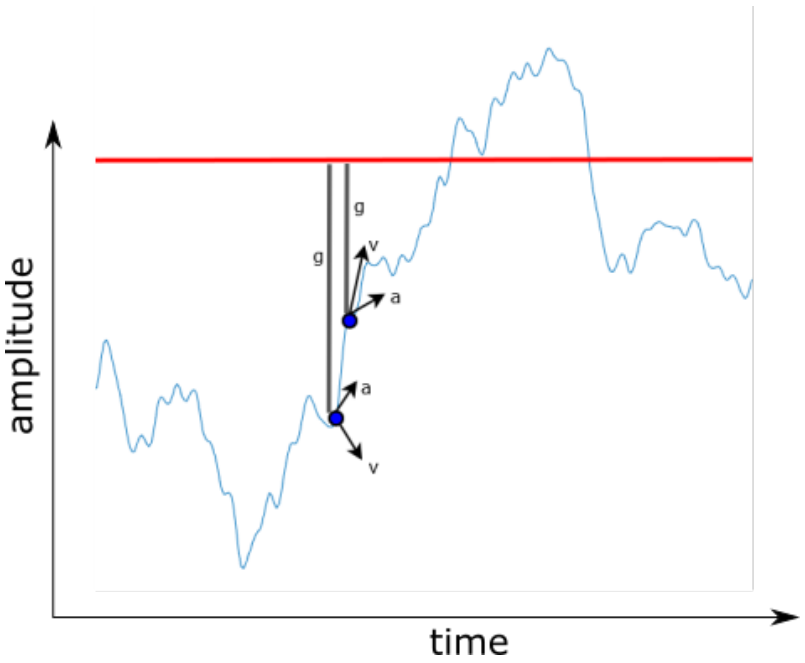


Figure 3.1: Gravity Model: 2 dimensional vector representation. Each sample in the digital audio signal is seen as a particle with velocity, acceleration and distance to the threshold (gravity). The threshold value is defined in red colour.

As an example, in this case, input samples could be compared to particles or bubbles floating in water. These particles go to the surface with a force f that in a simplified view could be similar to the Newton's Law of Universal Gravitation [28]:

$$f = G\frac{M}{r^2} \tag{3.1}$$

where G is the gravitational constant and M the mass of the earth, therefore we can neglect these values as a constants and take into account only that the force f is inversely proportional to the square of the distance r. This principle is used in a similar way as taking the surface as the threshold level in our limiter and r as the distance to the threshold th. For simplicity, the distance to the threshold will be called "gravity" or just g and it will be positive when the amplitude of the sample is smaller than the threshold, or negative when it goes above the threshold. The value of g is computed for each sample and used as an input for the ANN as

$$g[n] = \text{th} - |x[n]| \tag{3.2}$$

As can be seen in Figure 3.1, g is represented as the distance to the threshold. Another two characteristics of each sample are taken into account: velocity and acceleration. These characteristics are extracted in relation to the previous sample. Following the laws of kinematics, making $\Delta t = 1$ for simplicity in discrete domain and choosing n as the index of the current sample, v as velocity, a as acceleration and x as amplitude, one gets:

$$v = \frac{\Delta x}{\Delta t} \;\Rightarrow\; v[n] = x[n] - x[n-1] \tag{3.3}$$

$$a = \frac{\Delta v}{\Delta t} \;\Rightarrow\; a[n] = v[n] - v[n-1] \tag{3.4}$$

The network, during training, will need to understand how these parameters interact in order to get the best attenuation for the current sample. Figure 3.2 shows the topology of the neural networks with its inputs and output. It has the three inputs namely gravity, velocity and acceleration. The output, which is the attenuation (or gain) that multiplies with the input sample. The output will be always equal or smaller than one. The number of neurons in the hidden layer has to be found. Different trainings with a different number of neurons will be done and there will be different results for different configurations as will be explained in section 4.2.1. Not always more number of neurons implies a better performance.

Thanks to the abstraction of the model, these new characteristics can provide a wider space (from 2 dimensions: power/amplitude in time/frequency domain; to 4: velocity, gravity, acceleration and time) than current limiters. The network will use this benefit to map a solution for our purposes. The ANN will have more information which will be used to try to "predict" the trajectory of the samples and the probability to clip of the coming samples or not.

This is a research thesis where the aim is to know how this model could fit for

signal processing since, to the best of our knowledge, there is no previous information about this kind of model. Those characteristics are expected to provide more information than just the comparison of the samples with the threshold and map the corresponding attenuation to apply, as is implemented in current systems. Such systems often overshoot above the threshold and generate distortion since the limiter does not have time to prevent quick transients in the signal. However, this new approach could provide the information necessary to prevent it and generate less distortion.

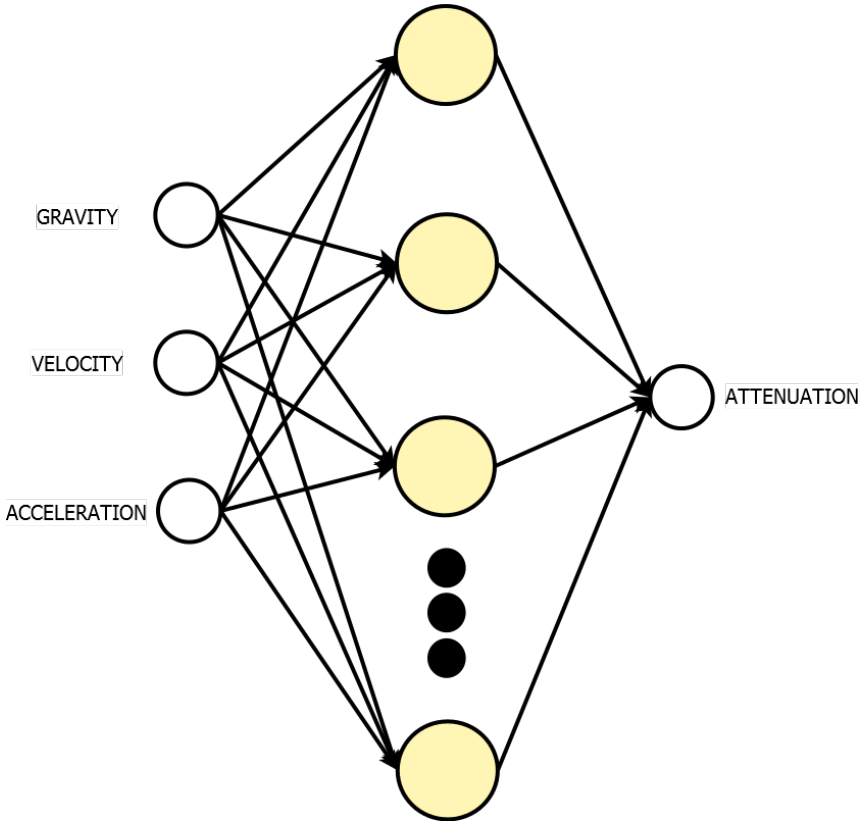


Figure 3.2: Artificial Neural Network general application used for the gravity model.

3.2 Activation Function

The purpose of the activation function is to introduce non-linear behaviour into the network. Non-linear means that the output cannot be obtained from a linear combination of the inputs. This function has to be monotonic and differential for better convergence.

The activation function used in this thesis for the hidden layer neurons is the sigmoid function described as:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.5}$$

The function is applied as it is for the hidden layer neurons, but for the output of the network, it could be mapped in two different ways. Two methods have been used, the first one uses the sigmoid function described in equation 3.5 and the output is in the range (0,1). On the other hand, the best range is (th,1) for the attenuation, therefore the output should be in that range. This method assumes that the network will find the best output values and it will adjust the range itself.

The second method assumes the network will need some help to adjust this range and it will modify this range with the next constraint:

$$f_{th}(x) = f(x)(1 - \text{th}) + \text{th} = \frac{1}{1 + e^{-x}}(1 - \text{th}) + \text{th} = \frac{1 + \text{th}\, e^{-x}}{1 + e^{-x}} \tag{3.6}$$

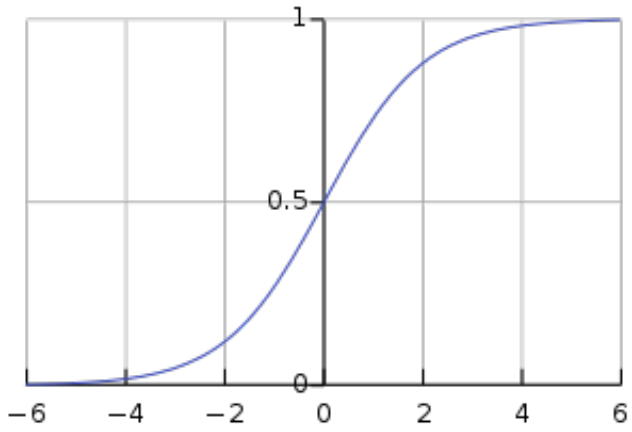


Figure 3.3: Sigmoid function. The output of the function is in the range $(0, 1)$ on the vertical axis while the input range is $(-\infty, \infty)$ in the horizontal axis. As can be seen, it has an exponential transition from negative input values to positives, being the output smaller than 0.5 for the negative inputs and bigger for the positive ones.

Other functions to test for the output neuron are the ReLu and the softplus functions [17]. These functions can be seen in Figure 3.4, and they should provide a more linear behaviour than the sigmoid version. These function can be also limited to have an output range between the threshold and 1, as the modification in equation 3.6. Another modification could be the slope of the functions, instead of behaving as the identity function, it can be modified to a linear function.

$$\text{ReLu} : f(x) = \begin{cases} 0, & \text{for } x \leq 0 \\ x, & \text{for } 0 \leq x \end{cases} \qquad \text{SoftPlus} : f(x) = \ln(1 + e^x) \tag{3.7}$$

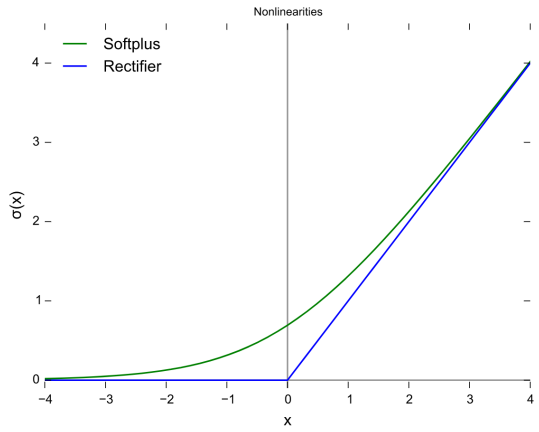


Figure 3.4: ReLu and softplus function. The output of the functions is in the range $(0, \infty)$, while the input range is $(-\infty, \infty)$. As can be seen, ReLu makes 0 all the negative inputs and linear (identity) transfer function for the positives, while softplus has an exponential transition from negative input values to positives. Softplus function fits ReLu with a soft transition.

3.3 Data sources

The database consists of 6 songs from different types of music: hard rock, metalcore, pop, dance, funk and dubstep. It contains instrumental and voice parts with different dynamic profiles, quality and frequency content.

At the beginning of the training, the training and validation sets are created following the next procedure:

1. One song from the six is selected randomly
2. From that song, a chunk of audio is selected randomly
3. The audio chunk is checked in order to know if there is at least one sample above the threshold. If not, come back to step 1.
4. Repeat until having the desired quantity of chunks.

The length of the audio chunk and the number of chunks in each data set in predefined for each training. Due to time constraints, sometimes short audio chunks and not too big data sets are better in order to not make very large trainings. This is important when we only need to get closer to some approach and see if it makes sense. Once we feel we are in the right direction these numbers increase in order to get better results with longer trainings.

It could vary from 20 to 100 audio chunks for the training set and half of it for the validation set. The length of the chunk could vary between 0.3 and 1 second.

3.4 Training

As explained in section 2.4, GAs are applied in order to train the network. Each chromosome corresponds to a set of weights in the ANN. The system was trained with a different number of neurons and different parameters for the fitness function. The results be shown in the next chapter.

Table 3.1 presents the parameters involved in the training of the ANN in order to match the behaviour of the desired limiter. These parameters were combined with different configurations in order to optimise the best performance of the system.

In relation with the GA, the population size is the total number of chromosomes during training. Generations per training are the number of generations (or a number of evolution steps) taken between each evaluation with the validation set (in order to check over-fitting and learning), while the number of trainings, is the total number of evaluations. Therefore, the total numbers of generations are equal to (generations per training)×(number of trainings). The method used for the selection step was the "tournament selection" explained in section 2.4.1. Tournament size is the number of individuals taken from the population, these individuals "fight" between them to be selected with a tournament parameter that gives the probability for the individuals with higher fitness to win. In the reproduction step, there is a crossover probability that dictates the probability to get offspring from the parents or if the parents pass directly to the next generation. Also, mutation probability defines the probability to mutate one of the genes of the chromosome. A recommended value from the prior art [10] is 1/(number of genes), and this has been the value used, inverse proportional to the number of genes in the chromosomes, which means that at least one gene of every chromosome should mutate.

parameter	min. value	max.value
population size	50	120
tournament size	1	4
tournament parameter	0.7	0.9
crossover probability	0.1	0.85
mutation probability	1/nGenes	1/nGenes
shaping penalty	1	500
clip shaping penalty	1	300
clip penalty	1	700
weights range	-40	40
threshold	0.5	0.9
neurons in hidden layer	2	16

Table 3.1: Relevant parameters and the range of values evaluated for the training of the neural network. This range of values is taken from prior art and try and error realisations.

The values related to the ANN, are the *weight range* that determine the range of values that the weights of the neural network can take, and the number of *neurons in hidden layer*.

Also, several *threshold* values were trained in order to know how flexible the network was for other threshold values different than the trained one. There were two different approaches, one was to train the network with a fix threshold value and later test how flexible the limiter was limiting the signal for different thresholds. The second approach was based on evaluating the individuals, when training, with a random threshold (a different one) each time.

3.4.1 Fitness Evaluation

The three penalties, shown in Table 3.1, are part of the fitness evaluation, and if the Gravity Model is one of the cores of the thesis, these penalties are the key-points of the training. These penalties are the parameters that teach the network how its behaviour has to be.

Since this training is based on non-supervised learning (there is no specific predefined correct output for each input), the fitness function evaluates each individual with some rules that add some penalties per sample when the behaviour does not fit the desired one. As can be seen in Figure 3.5, there are three possible outcomes when the input and the output sample are compared.

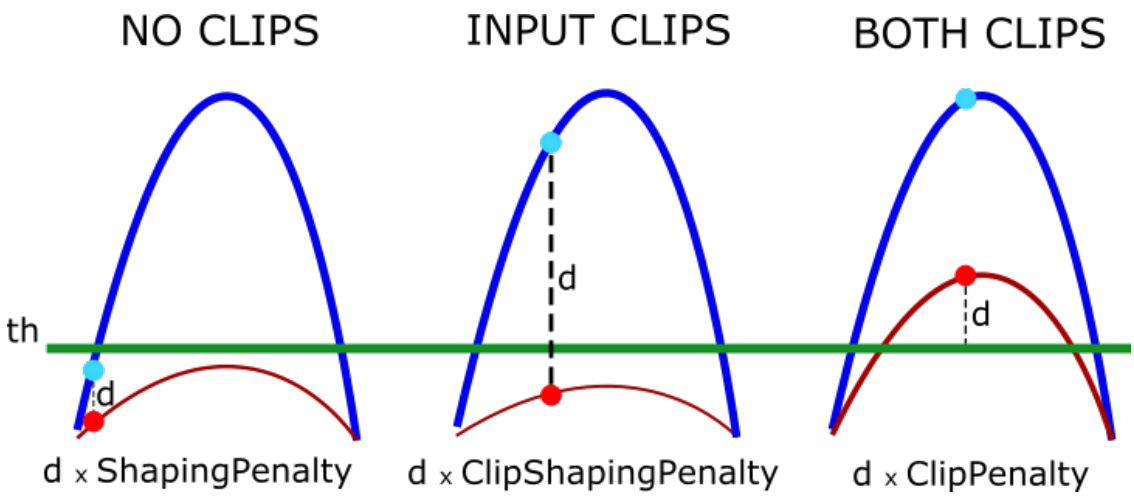


Figure 3.5: Penalties applied for the different cases that can occur when evaluating the fitness of the individuals. Blue dots are the input samples, red dots the output samples (the attenuated ones) and d is the distance of the dotted line. In the first case none of the signals clip (on the left). The second case (in the middle) the input clips, but the output does not clip and the last case (on the right) both clip.

In the first case, both samples are below the threshold. Hence, the output will be always equal or an attenuated version of the input. It will never have more amplitude since it is multiplied by the attenuation factor that is equal to or smaller than

1. In this case, the penalty applied will be (shapingPenalty $\times$ d), the distance d times the "shaping penalty" (previously defined). The behaviour that the network needs to learn is that if the input sample is below the threshold, the output has to be as close as possible to the input keep the shape of the signal.

In the second case, the input clips but the output does not. This is a good behaviour since the system is avoiding clipping, but it also needs to try to keep the shape of the signal. In order to re-shape the input signal below the threshold with highest possible fidelity, (clipShapingPenalty$\times d$) is applied to the input signal. This penalty cannot be extremely high, if not the system will start clipping since it tries to minimise the distance between the clipped input and the output.
The third case, both signals clip. This outcome is the least desired since the first aim of the limiter is to never go above the threshold. Therefore, (clipPenalty $\times d$) is applied, being d the distance to the threshold in this case.

As one can notice, if the system works fairly as desired, the first and the second cases will be the ones that more often happen. This is important to consider when choosing the values for the penalties since these penalties will be applied to a lot of samples. In the first case, the distance can be 0 and have a perfect match, then no penalty is applied, but the second one will be always applied.

The fitness for an individual k from the population is calculated by the inverse of the sum of the penalties p applied to each sample k:

$$\text{fitness}_k = \frac{1}{\sum_{n=1}^{N} p_k[n]} = \frac{1}{\sum_{n=1}^{N} d_k[n]c_k[n]} \tag{3.8}$$

where d is the distance represented in Figure 3.5 for each case and c the penalty applied in each case.

3.4.1.1 Delta Mode

One variation created for the two first cases is the "Delta mode". The Delta mode compares the difference between the distance of the current and previous input sample and the distance of the current and previous output sample.

This representation of the distances for the Delta mode can be seen in Figure 3.6 and the equation for each sample k of the new d becomes:

$$d_k[n] = |d_{k_{in}} - d_{k_{out}}| = |\ (x_{k_{in}}[n] - x_{k_{in}}[n-1]) - (x_{k_{out}}[n] - x_{k_{out}}[n-1])\ | \tag{3.9}$$

This method allows a fairer penalty since the system aims to re-shape the same signal below the threshold, which means both signals should have similar direction or slope between consecutive samples. With the previous method, in the first case the system will try to have the same samples in both signals all the way until the threshold once the input clips, the output will try to remain close to the threshold. This phenomena distorts the shape of the signal. However, the Delta mode tries to keep the same transitions between samples for both signals, no matter if the input

is above the threshold and the output below.

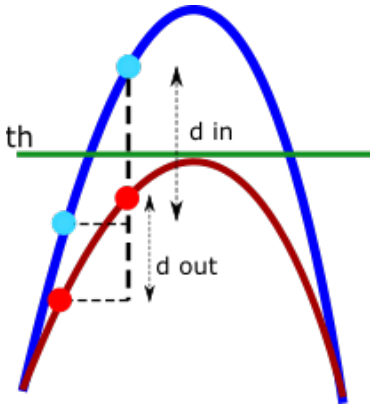


Figure 3.6: Penalty applied in "Delta mode". This penalty takes the difference between the distance of the input samples and the output samples (difference of the slopes). Blue dots are the input samples, red dots the output samples (the attenuated ones).

Another important fact is that the network should attenuate only the batches of samples where some of them clip. That means that in order to keep the shape of the signal, some samples that are not clipping will be attenuated around the samples that are attenuated because they are clipping, as can be seen in Figure 2.7. But the batches where there is no clipping, there should not be attenuation. This is regulated in the current limiters with the attack and release time.

3.4.1.2 Tightness Method

Another variation created for the shapingPenalty case is the Tightness Method. It can be applied with the Delta Mode or without it. This method applies less penalty for the samples closer to the threshold. Then, the system will try to adapt its behaviour in the way that the samples close to 0 amplitude will be "tightened" to the original signal, while the ones closer to the threshold will have more freedom to move. This is due to the total application of the penalty for the samples close to 0 and a percentage of the penalty for the samples close to the threshold (samples above the threshold are not taken into account for this case, that takes part within the clipShapingPenalty). The relation with the percentage of the penalty used (tightness) Γ, will be derived from:

$$\Gamma = 1 - \tau \left(\frac{x}{\text{th}} \right)^4 \tag{3.10}$$

where x is the input level of the signal, th is the threshold level, and τ the tightness factor. Therefore, the penalty applied will be in proportion with Γ ($p_k[n] = \Gamma p_k[n]$). This function was obtained after some brainstorming until finding one behaviour similar to Figure 3.8. The exponent selected (power of 4) was seen as close enough

to the desired behaviour, but other exponents will not change too much the outcome since the network will adapt to those changes.

The range of τ is (0,1) and smaller values mean higher tightness. Its effect can be seen in Figure 3.7

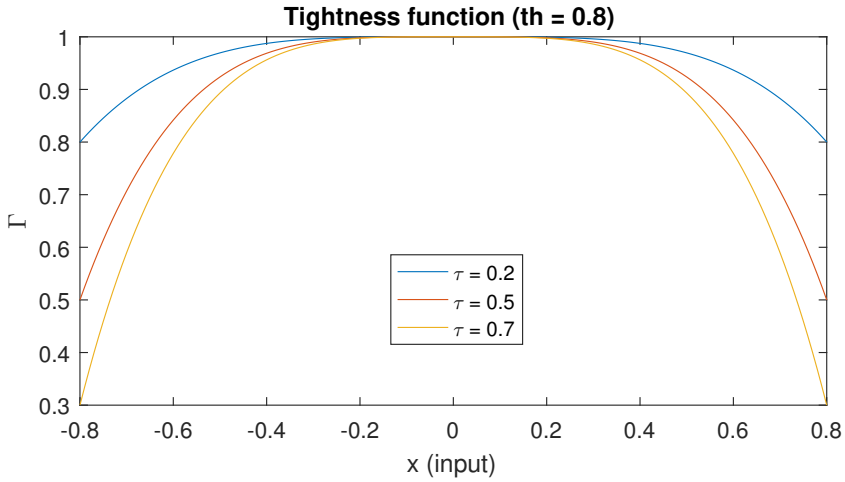


Figure 3.7: Tightness function for a threshold level of 0.8. Since this method is only applied for the case of the shapingPenalty, the input signal is in the range (-th,th). When the input is close to 0, Γ is 1 and the total percentage of the penalty will be applied. For different τ values, tightness changes. Higher τ means more freedom closer to the threshold since a smaller percentage of the penalty will be applied.

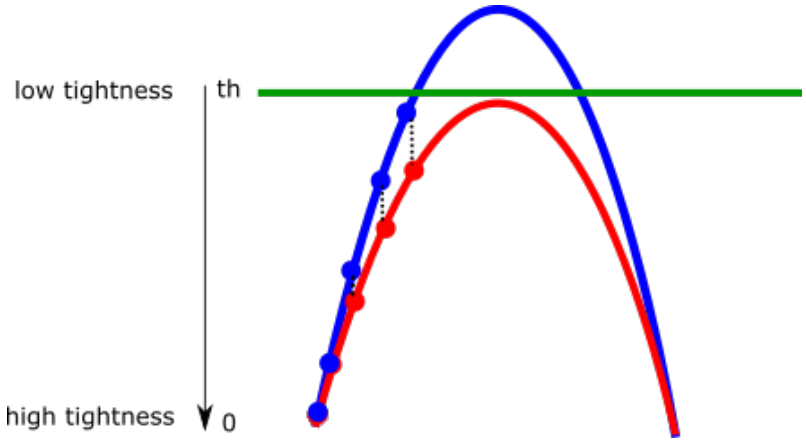


Figure 3.8: Tightness increases for input values closer to 0. The input signal is represented in blue and the output in red in this ideal scenario. The output is attenuated just below the threshold and it keeps the shape as close as possible to the input signal. As a result, the samples closer to 0 have similar values between the input and the output (they remain tight). The ones closer to the threshold are more separated (as the black dotted line indicates) and need some freedom (low tightness) in order to stay below the threshold and maintain the input shape.

Figure 3.8 shows how the ideal case fits with the Tightness Model and setting different τ values, the behaviour of a look-ahead limiter with different release time could be approximated. As can be seen in Figure 3.9, with longer release time the output is less tight to the input values, even for values close to 0. When decreasing the release time, input and output start to be tight for small amplitude value. When release time is almost 0, input and output are almost the same all the amplitudes until the threshold where some low level of freedom is granted.

In this way, the value of τ maps inversely to the release time of the system. For small values of τ tightness is higher and the release time is shorter. Also the opposite, for a bigger τ value, longer release time.

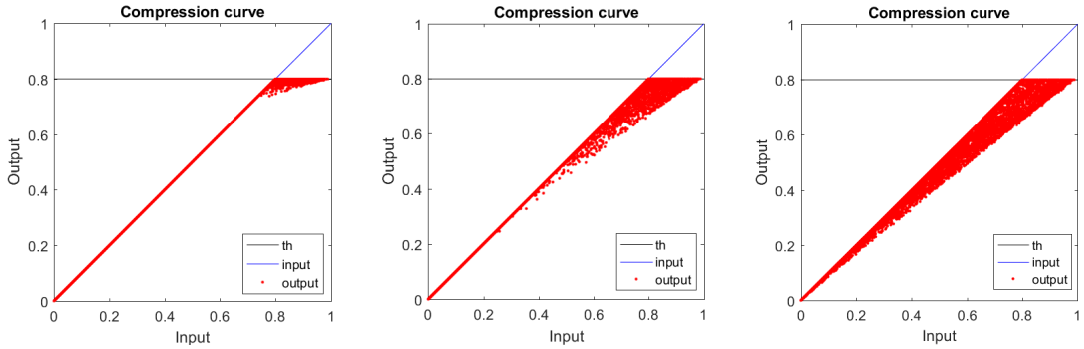


Figure 3.9: Input-output samples relation. From left to right, the same limiter with release time 0.1, 1 and 10 ms. As can be seen, when the release time increases the responses is much disperse and farther from the ideal case due to the reasons explained in Figure 2.5.

3.4.1.3 Fair Penalty Accumulation

There are two different methods for the evaluation of the individuals. The simple one is to accumulate for each sample its penalty, but after some training, it was thought if that was fair enough for the system. There are three different penalties or scenarios when evaluating one individual. If for some random data sets there are much more occurrences of one scenario than the others, the system will be trained for that scenario, but probably not for the others.

In order to avoid this, a variation was implemented: there will be three counters, one for each scenario in order to count the occurrences of each one. Each penalty contribution is accumulated individually for each scenario and weighted by the number of occurrences. Hence, once normalised/averaged individually each scenario, each one will contribute in the same way to the total penalty.

All of this methods and variations will be tested and combined between them in order to know how to reach the desired behaviour of the system. The results will be shown in the next chapter.

An attempt to use prediction based methods resulted in inordinate complexity, and hence, it was not pursued further. The description of the prediction based approach and the results thus obtained can be found in A.

4
Results

This chapter gives a description of the results obtained for the training methods explained in the previous chapter. It is important to note that these system models are not deterministic and the results could vary depending on many factors. Therefore, due to the stochasticity, the same training input can converge to different neural networks. This is a problem in affirming which is the best neural network or which combination of parameters to use. These results will guide to what can be derived from different combinations of parameters and methods.

4.1 Challenges

This section explains potential problems that the system shall be able to prevent. The most important are distortion and clipping. Some of the first implementations resulted in these problems. It is important to analyse how such issues occur.

4.1.1 Clipping

The obvious challenge for a limiter is to limit the input signal. The output has to be generated below the threshold level. As can be seen in Figure 4.1, one example of the attenuation generated by the system is not enough to avoid clipping.

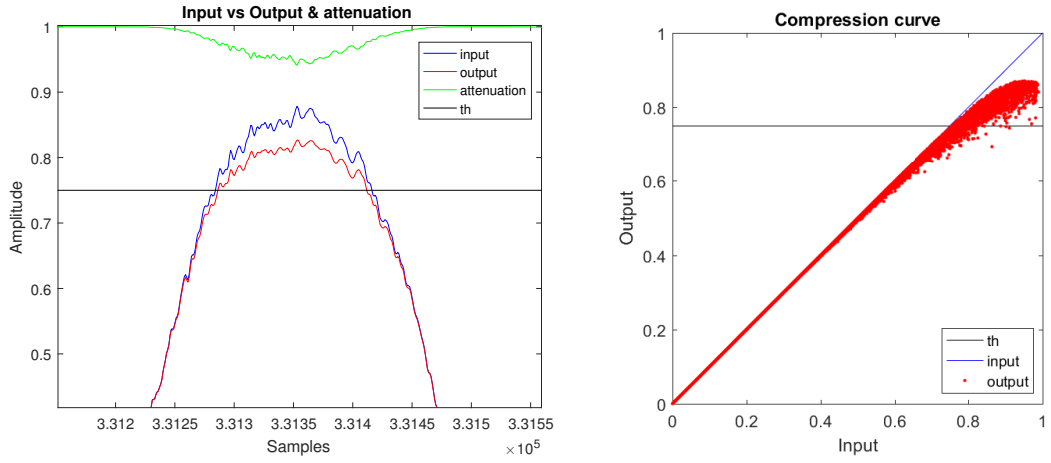


Figure 4.1: Example of a system that does not generate enough attenuation. On the left, input and output signal and the attenuation level for each sample. On the right, the input-output relation.

4.1.2 Distortion

Distortion could be caused out of different reasons. One reason could be discontinuities in the signal as in Figure 4.2. In this case, the attenuation is applied instantly when the signal is above the threshold and the same attenuation is applied to the samples above it. The input-output relation can be seen in Figure 4.2. The behaviour of the systems is linear with ratio 1:1 until the threshold and then it jumps to the attenuated state applying the same gain reduction factor (in this case, it scales the input signal above the threshold multiplying by the threshold level).

The output is attenuated below the threshold, but the transition is not linear and it jumps so fast to a higher attenuation level when the input signals go above the threshold. This causes audible distortions because of the discontinuities. If the magnitude of the input to the sigmoid activation function is too high (positive or negative) the outcome will be mostly two states: negative saturation or positive saturation. Only a few values will be out of the saturation states. In this case, the output attenuation range is between 1 and the threshold and it jumps very fast from the linear behaviour to the maximum attenuation (without smooth transitions).

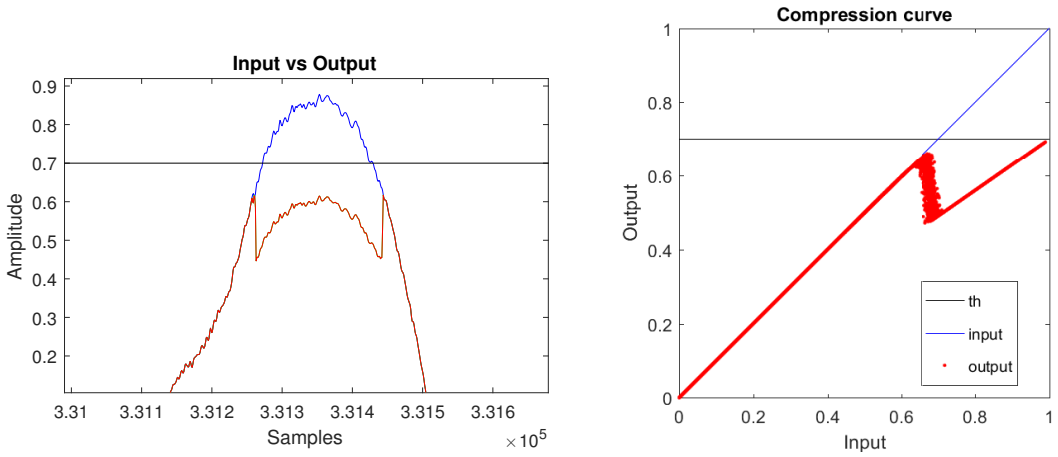


Figure 4.2: Distortion caused because of the non-smooth transition between the non-attenuated part and the attenuated one. On the left, the black line is the threshold, the blue line is the input and in red the output. On the right, if the weight range is too big, it is easier for the ANN to saturate in the output.

Other reasons could be nonlinear behaviours creating artefacts in the output signal, new shapes or inverting some parts as seen later in Figure 4.4.

4.2 Methods and Parameters

This section explains why some parameters and methods are chosen and what is the behaviour of the system found due to those changes.

4.2.1 Number Of Neurons

4.2.1.1 Initial Decisions

The number of neurons in the hidden layer used for the model is an important parameter to determine. If the number of neurons increases, the complexity of the system increases linearly and the complexity of the training exponentially. Also, increasing the number neurons in the hidden layer requires longer chromosomes, and for longer chromosomes, the population size should be bigger. Then, we aim to find the minimum number of neurons that make the system to work properly for the desired task. Due to time constraints of the thesis we avoided using longer training.

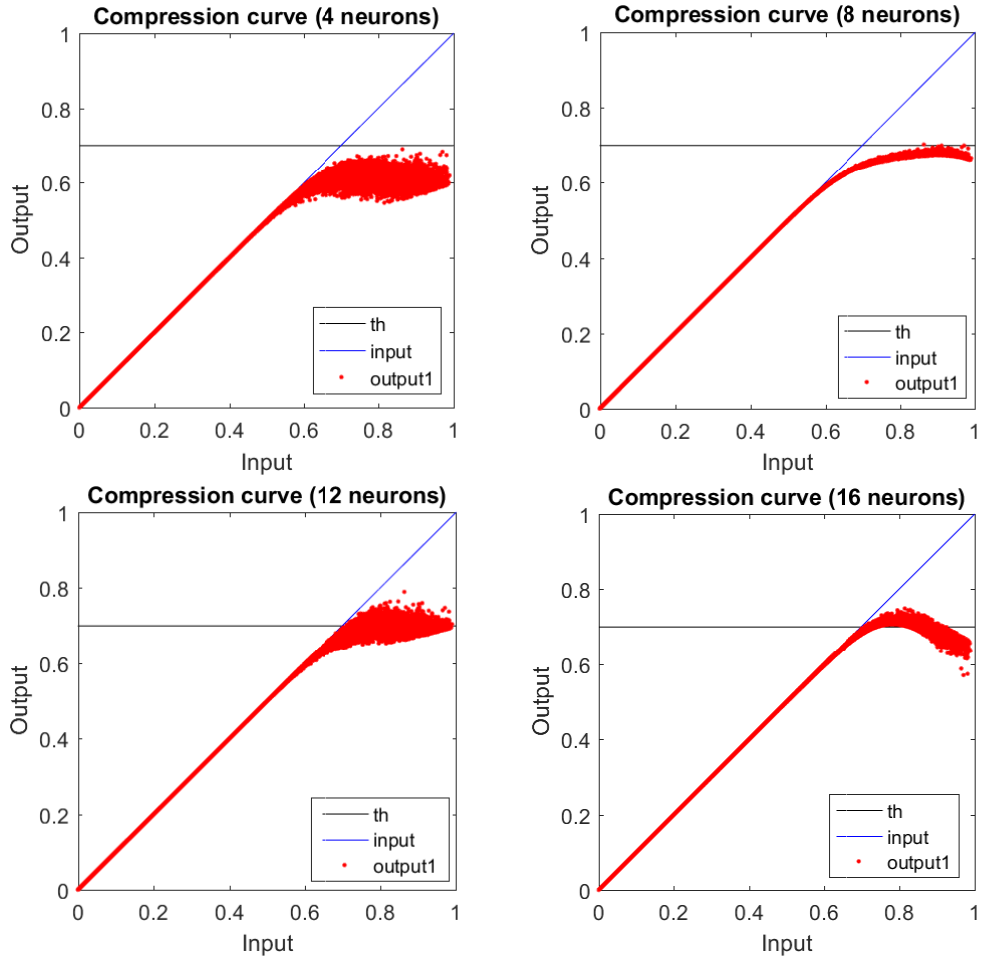


Figure 4.3: Input-Output compression relation for the different number of neurons. The vertical axis represents the output amplitude obtained from the input amplitude of one sample because of the attenuation applied in the limiter. The figure shows the behaviour of 4 different systems trained with different number of neurons, but with the same combination of methods and parameters.

In order to know how many neurons are enough for the model, some initial comparison of different penalties combinations and the simplest methods were obtained. As can be seen in Appendix B, different combinations were tested. One can see that 4 different networks (with 4, 8, 12 and 16 neurons in the hidden layer) trained

with the same combination of penalties results in similar behaviour for all of them. There are no figures in the appendix for networks with 2,3,5 and 6 neurons, but the simulations were done and the conclusion was the same.

The conclusion was that if the number of neurons is bigger than 4, there is no significant difference between the different outcomes. Therefore, in order to avoid long training times, 4 neurons was chosen as the best number. Also, as can be seen in Figure 4.3 (as an example), the case with 4 neurons seem to have less distortion in the output and also it is always below the threshold. The case with 4 neurons is similar to the case in Figure 2.5, like the case of a limiter with short release time. With 8 neurons is similar but without release time, it could be the case of the ideal limiter with a soft-knee. The case with 12 neurons clips and with 16 neurons also distorts. Distortion is obtained when the input-output relation goes up and down. In this case, the relation is 1:1 up until the threshold, it goes above and then the attenuation is too high for high amplitude inputs and the output goes down. This behaviour causes non-smooth attenuation transitions as in Figure 4.2, where the transition between the not attenuated and the attenuated part is nonlinear.

To sum up, as an initial conclusion, a large number of neurons in the hidden layer do not necessarily imply increased performance. But, once the best combination of methods and parameters is found, it should be trained and tested one more time with a different number of neurons in order to verify the best performance of the solution.

4.2.1.2 Final Choice

After studying the rest of the methods and parameters, it has been found that the behaviour of the system increasing the number of neurons is similar to the initial guess. Therefore, the hypothesis of using 4 neurons in the hidden layer is still valid after testing several numbers of neurons with the best combinations of parameters and methods found.

4.2.2 Penalty Accumulation

During training, it was found that in order to have a better control over the penalties the Fair Penalty Accumulation Method (see section 3.4.1.3) is better. If not, the training depends on the data set and the number times that each case occurs. Also, convergence is worse since what is working for one audio data set is not working for another because the system is more trained for one case than other. Hence, the system could be better keeping the shape but does not limit at all, or another case when it is focused on the signal limitation but not on keeping the shape. Without the Fair Penalty Accumulation method, it becomes very hard to adjust the penalties relations.

4.2.3 Activation Function

The activation function used is the sigmoid function for the hidden layers as explained in section 3.2 and Figure 3.3. But for the output neuron, the modified version of the different functions (applying the same procedure as for the sigmoid function in equation 3.6) was tested in order to avoid extra attenuation. This modification helps the network limiting the output of the ANN. The input to the activation function at output neuron of the ANN could saturate easily positive or negatively. If it does negatively and there is no limitation, the network will kill the signal for that samples since the attenuation (negative gain) is equal to 0. Therefore, is much easier to converge to the desired results with this modification.

When training the system without this modification, some crest inversion effects were found often. Figure 4.4 shows this phenomena. The attenuation is too high for the peaks in comparison with the rest of the envelope and the crest is inverted.

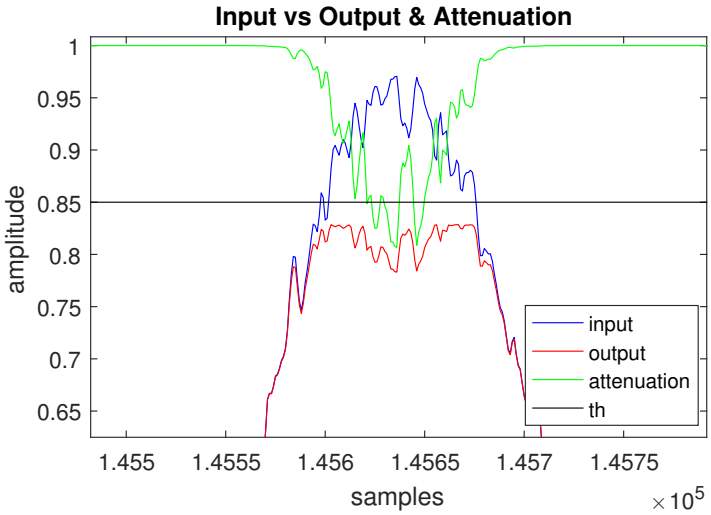


Figure 4.4: When the system output can generate an attenuation level (negative gain) below the threshold, sometimes it can converge to cases where the crests of the signal get inverted. This can be avoided if the range of the output system is between 1 and the threshold.

Other activation function like the "modified linear" (similar to identity, but adding some slope and modified with some limits, see Figure 4.5) and an expanded version in the horizontal axis of the sigmoid was tried out, but they did not provide a better approach for the model. However, the "modified ReLu" (increasing the slope of the normal ReLu function and limiting its maximum and minimum output, as with the other functions) worked better than the sigmoid when applying attenuation for the clipping input, as can be seen in Figure 4.6. The behaviour is more linear in this case for this activation function, while for the sigmoid the attenuation fluctuates.

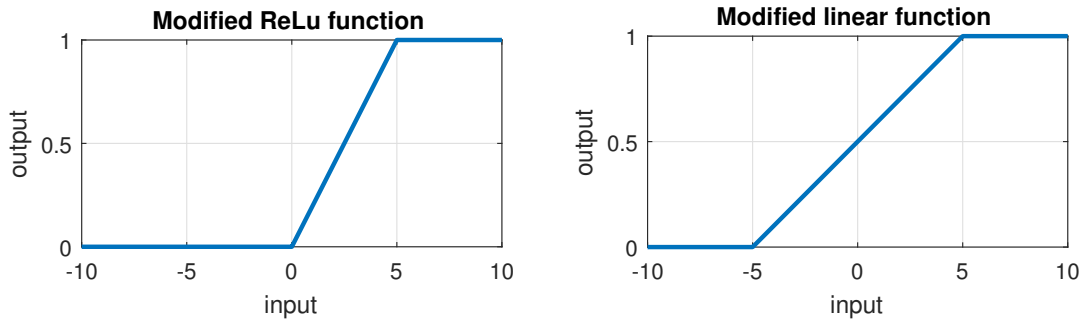


Figure 4.5: Modified ReLu and Linear Function. In these examples, the limits go till 5, but different limits (resulting in different slopes) were tried out.

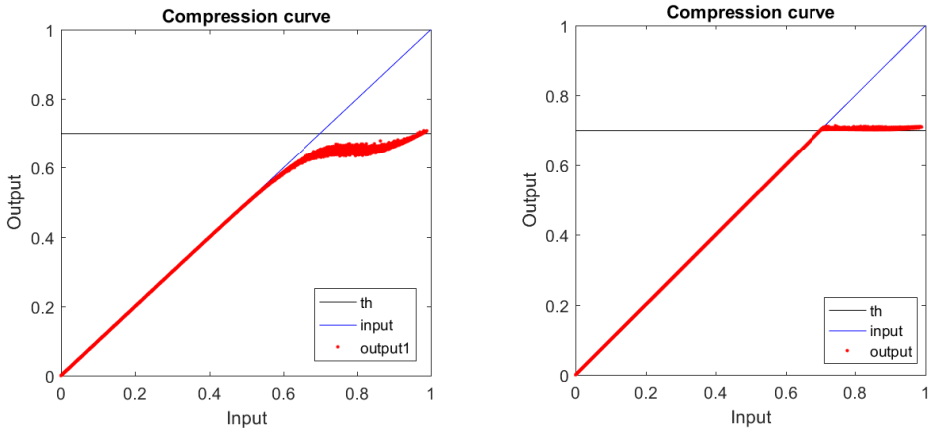


Figure 4.6: On the left, the result from an ANN with a sigmoid function in the output neuron. On the right, the same result with a ReLu function. The transition between the linear behaviour and when applying attenuation is smoother for the sigmoid. But the attenuation is more linearly applied to the ReLu function than the sigmoid.

The SoftPlus function (see equation 3.7) was tested and some modification to the function were made in order to get different behaviours in the transitions between the linear state (when the output signal is equal to the input signal) and the attenuated state (the parts when the samples of the output signal are an attenuated version of the input).

As can be seen in Figure 4.7, the three cases of the "Modified SoftPlus" are variations of where the soft transition is made (also adding the same modification as the previous functions, regarding limitation and slope). In the SoftPlus Single function, the soft transition is like the normal SoftPlus function at amplitude 0. But the transition to the limitation is "hard". The case of the SoftPlus Inverse is the opposite of the single, where the soft transition is applied to the limitation and the hard one at amplitude 0. The SoftPlus Double has both transitions soft.

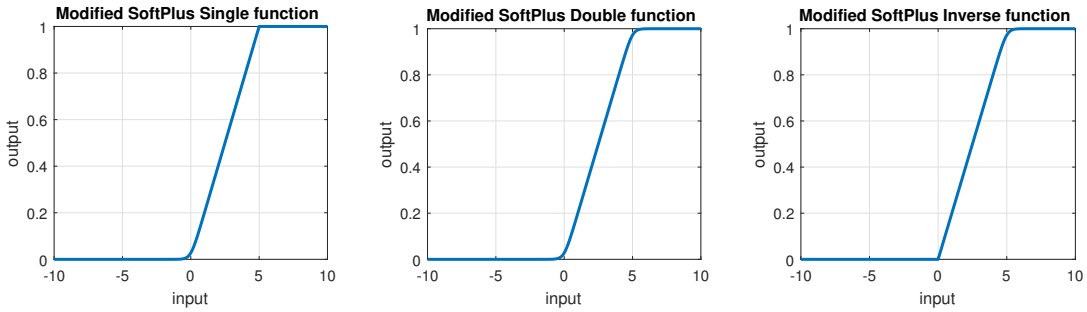


Figure 4.7: Modified SoftPlus Function. In this example, the limits go till 5, but different limits (resulting in different slopes) were tried out. There are the Single SoftPlus where only the corner at 0 has a soft transition (the original version from the theory, but limited on the top), the SoftPlus Double where both corners have soft transitions, and the SoftPlus Inverse variation where only the corner at amplitude 1 has a soft transition.

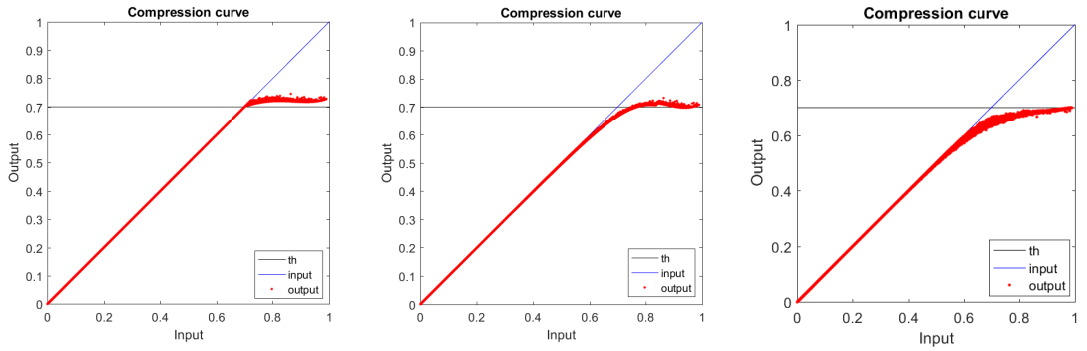


Figure 4.8: On the left, the result from an ANN with a SoftPlus Single function in the output neuron. In the middle, the same result with a SoftPlus Double function. On the right, with a SoftPlus Inverse function. As can be seen, the different transitions from the linear behaviour to the attenuation vary in relation with the transitions explained for each function in Figure 4.7.

Choosing different activation functions results in different behaviours in the attenuation state. Figure 4.8 shows the different behaviours and the relation between the transitions in the activation function and the transition between states for a limited audio signal. If the transition in the activation function is “soft”, at amplitude 0 some fluctuation can be seen in the attenuation state in the input-output relation. This fluctuation could generate crest inversion, but it can help the network to converge without abrupt changes in the attenuation state. On the other hand, if the activation function has a soft transition in the limitation (at amplitude 1), the transition between the linear state and the attenuation state is soft.

4.2.4 Tightness and Delta Modes

Delta Mode adds a meaningful information when training. It says if the comparison between input and output shape has to be measured from the amplitude values of

each signal or the slope between the current sample and the previous.

This method can be applied in two different scenarios: in both of them the output sample does not clip, but the input one can clip or not. As can be seen in Figure 4.9, when the Delta Mode is applied only for the case when the input clips, the shape in the output signal is not kept in relation to the input one. In the other case, applying always the delta mode for both cases, the shape is kept. This training was made with the same values and parameters and just modifying where for which cases the Delta Mode has to be applied. When Delta Mode is not applied, the distance between the input and the output sample is used as an error.

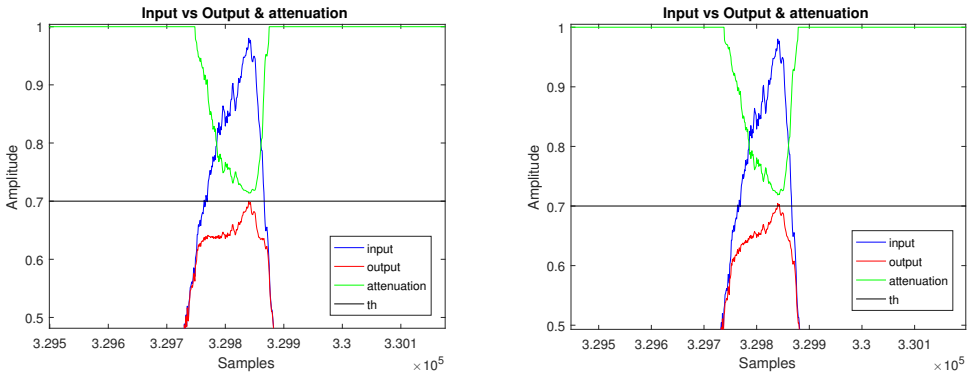


Figure 4.9: Delta mode. As explained in section 3.4.1, on the left Delta Mode applied only to the case when the input clips and the output do not clip (clipShapingPenalty case). On the right, Delta Mode applied for both cases: the previous case and also when input and output do not clip (shapingPenalty and clipShapingPenalty cases). Blue signal is the input, red the output and threshold are the black lines. The green signal is the attenuation applied to each sample. On the right, the output keeps the shape of the input but attenuated, while on the left the shape is only kept when the input signal clips.

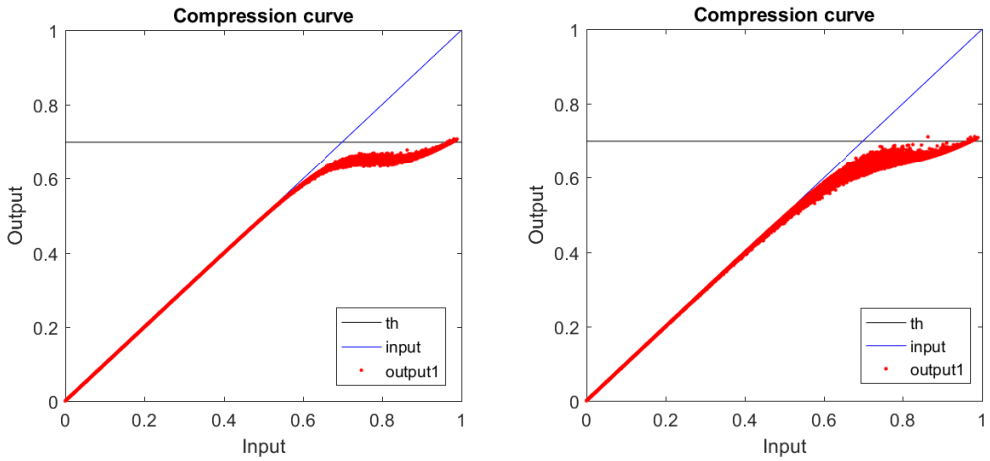


Figure 4.10: Tightness Mode. On the left, a result without tightness method and on the right, applying it with a tightness factor $\tau = 0.7$. $\tau = 0$ means disabling tightness method and $\tau = 1$ high freedom.

As explained in section 3.4.1.2, the tightness method provides more flexibility in the transition from the linear state to the attenuation state. This method tries to avoid the distortion cause for the discontinuity between these states (already explained with Figure 4.2 before).

As can be seen in Figure 4.10, when applying the tightness method the transition is much smoother. Because the constraint of the penalty to keep the exact shape of the signal below the threshold is weaker when gets close to the threshold.

4.2.5 Penalties

The values of the penalties for the fitness evaluation have varied several times since each method applies different rules when applying the penalties. In general, the important fact is the relation between the penalties more than the value of each one separately.

The shapingPenalty tries to tighten the output signal to the input signal's shape. The result of increasing this penalty is shown in Figure 4.11. The transition between the linear state and the attenuation state is centred around 0.5 and 0.6 in the first case. When increasing, the transition occurs above 0.6 and the values of the output remain closer to the values in the input for a larger range. This forces the transition to be closer to the threshold.

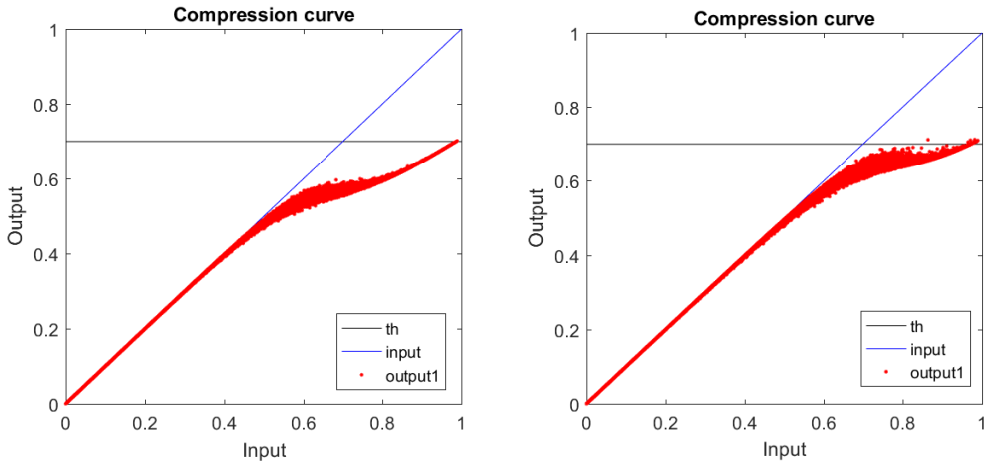


Figure 4.11: ShapingPenalty. With the same parameters and penalties, when the value of the shapingPenalty is increased, the behaviour change from the figure on the left to the figure on the right. Increasing the shapingPenalty means that, when the input is not clipping, the system tries harder to have in the output the same values as the input.

Figure 4.12 shows the result of increasing the clipPenalty and how the system reacts to this change. As can be seen, when the constraint imposed by the penalty is higher, the system that was focused on keeping the shape of the signal, starts applying attenuation above the threshold.

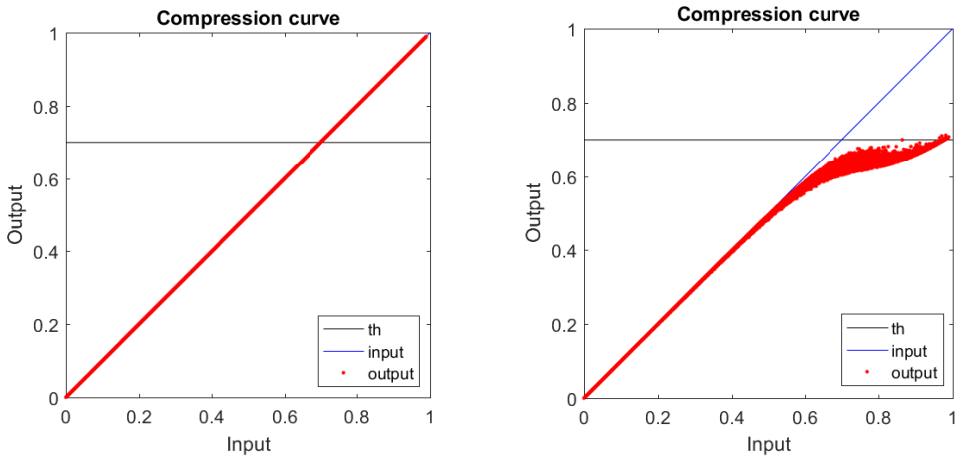


Figure 4.12: ClipPenalty. With the same parameters and penalties, when the value of the clipPenalty is increased, the behaviour changes from the figure on the left to the figure on the right. Increasing the clipPenalty means that, when the input is clipping, the systems tries harder to do not clip in the output. It is very easy for the system to jump from a behaviour that just tries to fit the shape of the input signal or start limiting the signal as a priority. This penalty could be very sensitive.

For the case of the clipShapingPenalty, as can be seen in Figure 4.13, exists a nonlinear behaviour that can cause distortion in the signal for discontinuities. Increasing the clipShapingPenalty, the system converges to a solution where the clipping samples (in the input) are attenuated but trying to keep the shape as close as possible. The result is a more linear behaviour in that range.

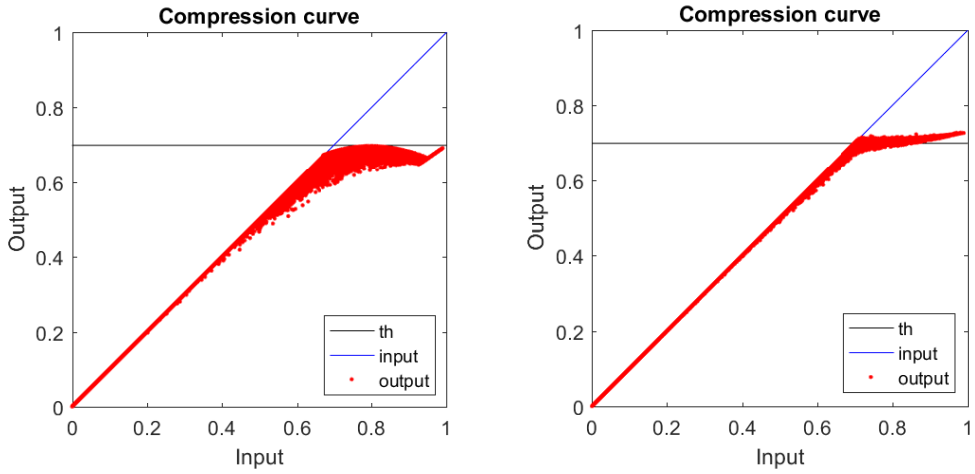


Figure 4.13: ClipShapingPenalty. With the same parameters and penalties, when the value of the clipShapingPenalty is increased, the behaviour change from the figure on the left to the figure on the right. Increasing the clipShapingPenalty means that, when the input is clipping and the output is not, the systems tries harder to have in the output the same shape as the input. On the left, the behaviour creates some distortion and crest inversion (as explained in Figure 4.4), On the right, the attenuation is more linear for different input amplitudes.

It is important to note that the relevant fact is the relation between the penalties and not the values of each penalty individually.

4.2.6 Parameters Complexity

There is a huge number of possible combinations of parameters and methods, therefore there is a high complexity in order to find best general behaviour of the system.

parameter	Complexity	Proposal
Neurons in hidden layer	Tested integers from 2 to 16	4
Tournament size	Tested integer from 1 to 4	3
Tournament parameter	Tested real numbers from 0.7 to 0.9	0.8
Crossover probability	Tested real numbers from 0.1 to 0.85	0.6
Weights range	Tested integers from -40 to 40	5
Delta Mode	It can be not used, or combined with the clipShaping and Shaping Penalties	Use it for both cases.
Tightness	Tested real numbers from 0 to 1	~ 0.3
Fair Penalty Accum.	It can be use or not	Use it.
Output Limitation	It can be use or not	Use it.
Shaping penalty	Tested from 1 to 200. It depends of the desired behaviour.	$\sim$ 200 (Using Delta Mode, Fair Penalty Accumulation and Output Limitation)
Clip Shaping penalty	Tested from 1 to 300. It depends of the desired behaviour.	$\sim$ 10 (Using Delta Mode, Fair Penalty Accumulation and Output Limitation)
Shaping penalty	Tested from 1 to 700. It depends of the desired behaviour.	$\sim$ 15 (Using Delta Mode, Fair Penalty Accumulation and Output Limitation)
Activation function	6 different functions: Sigmoid, Linear, ReLu, SoftPlus Single, SoftPlus Double and SoftPlus Inverse. It can have different slopes, tested from 1 (identity) to 5	SoftPlus Inverse with slope 1(identity)

Table 4.1: Parameters complexity and proposal.

Table 4.1 shows all the parameters and methods used. It also shows a proposal for

which method or parameter to use when training the system. These parameters help the network to converge to the desired behaviour, but it is a stochastic model that will vary depending on several factors as the database or the random initialization of the chromosomes.

This method is the tool to find the desired system, but not the solution. Different users could desire different responses from the system and therefore, one will not be better than the other. This is the reason why a proposal is given, but not the best match for a parameter or method.

4.3 Threshold Flexibility

Since the system is trained for one threshold, its best response is when we run it for that threshold. On the other hand, it has been found that there is some flexibility where the system still can work good, but the best solution for a real implementation would be to have different already trained systems with different thresholds.

Taking into account this flexibility, for example, one system could be trained for a threshold at 0.7 and run with thresholds from 0.6 to 0.7. But this is something that needs more research and it is out of the scope of the thesis.

4.4 Extra Blocks

Some extra blocks could be added to the chain after the ANN. Two different blocks were implemented trying to improve the result obtained from the ANN.

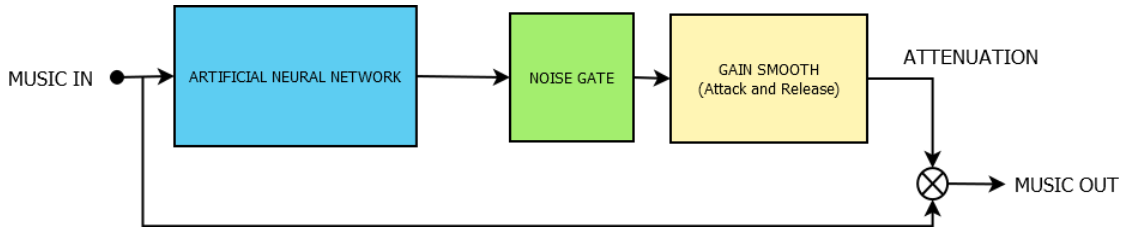


Figure 4.14: Block diagram of the application adding Noise Gate and Gain Smooth blocks.

4.4.1 Noise Gate

This blocks remove small attenuation values that could attenuate some part of the signal that is not going to clip. A noise threshold level has to be set. Figure 4.15 shows an example with a noise threshold equal to 0.96. All the attenuation above the threshold, between 0.96 and 1 will be not considered (this threshold works opposite as the limiter, it removes the part above the threshold). It can be seen that the signal attenuated using the noise gate block (green signal) attenuates the parts that are clipping in the input, but not the ones that are below the threshold. The signal using the attenuation from the ANN (in red) attenuates both parts.

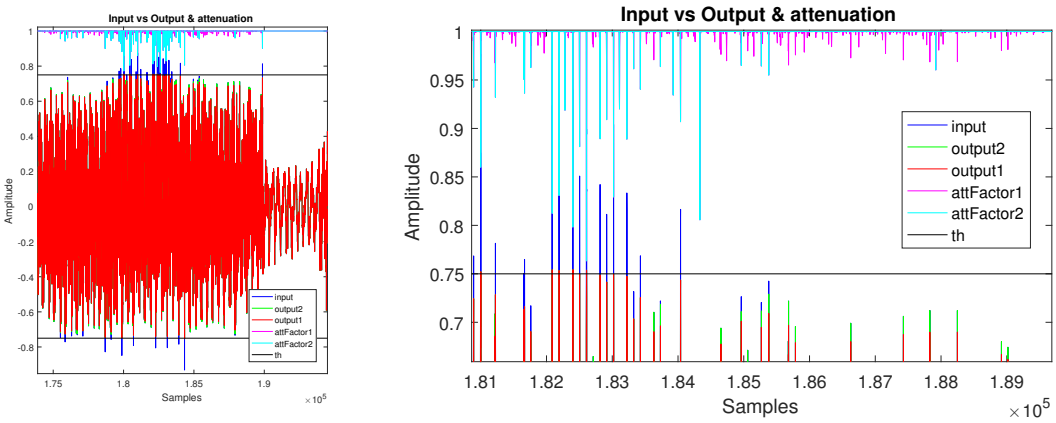


Figure 4.15: Example of Noise Gate for attenuation signal On the left is the signal and on the right a zoom of the same signal for a more detailed view. Purple shows the attenuation obtained from the ANN, light blue after noise gate (removing attenuation between 0.96 and 1), the red signal is the attenuated directly from the ANN and the green one with the attenuation filtered with the noise gate.

On the other hand, one cannot abuse of this block using a very low threshold (considering a lot of noise to remove). This will lead to nonlinearities when attenuating signals above the limiter's threshold, as can be seen in Figure 4.16. It can be seen in different states in the new output (in green), breaking a breach in the original (in red). This will cause discontinuities in the output signal as was explained in Figure 4.2.

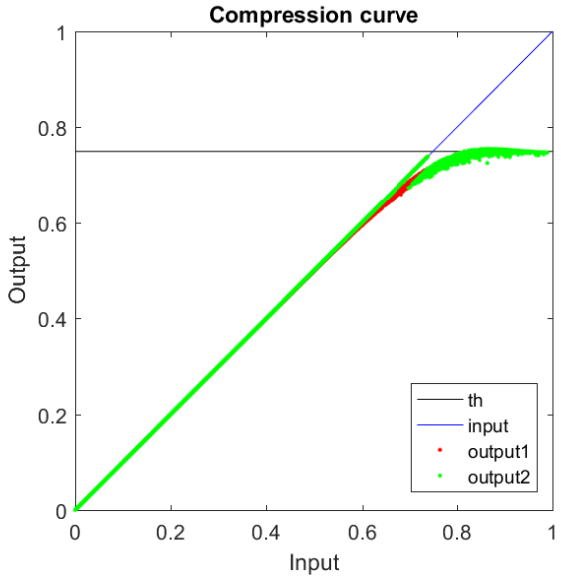


Figure 4.16: Input-output relation after Noise Gate block. Red is the output without using the noise gate, and green using it.

4.4.2 Gain Smooth: Attack and Release Time

As it was explained in section 2.1, the same block for gain smoothing than normal limiter is applied for the same purpose. Figure 4.17 shows an example of how the attenuation signal is filtered and the transitions in the signal become slower. The outcome is a signal with a shape closer to the input and fewer nonlinearities due to fast changes in the attenuation.

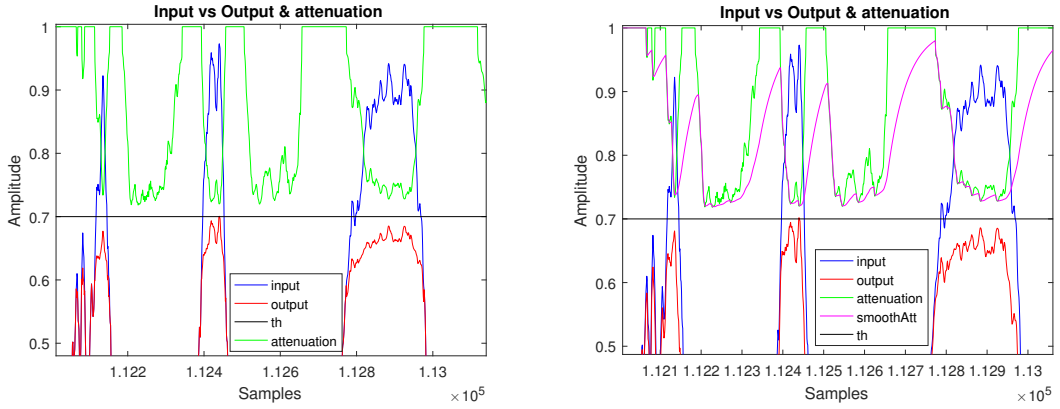


Figure 4.17: Attenuation with Gain Smooth block. On the left, a signal attenuated without applying smoothing to the attenuation signal. On the right, applying smoothing. Green is the attenuation from the ANN and purple after the Gain Smooth block.

4.5 Comparison

In this section, different audio limiter implementations are tested and compared:

- Hard knee limiter: All the samples below the threshold are the same, above are equal to the threshold. Explained in Figure 2.1 when ratio is $\infty : 1$.
- Soft knee limiter: Same as the hard knee limiter but with a soft transition to the attenuated state. Explained in Figure 2.2.
- Look-ahead Limiter: Limiter with some delay in order to anticipate the attenuation, hence keeping a better shape of the signal. Explained in Figure 2.6.
- ANN Limiter: One realisation of the system described in this thesis. The response of the limiter can be seen in Figure 4.18.

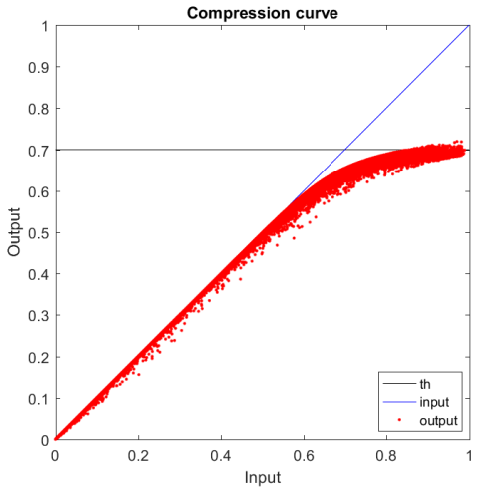


Figure 4.18: ANN Limiter input-output relation. This response is measured without any attack or release time over the input. It is the raw ANN limiter relation. As can be seen, there are some nonlinearities in the behaviour, acting in different ways for samples with the same amplitude.

As has been explained before, there could be very different responses and this was chosen because it does not create too much distortion in comparison with another result. On the other hand, it clips sometimes but only slightly above the threshold. The nonlinear behaviour was taken into account when choosing it in order to measure the responses of the nonlinearities in the outcome.

4.5.1 Clipping

For this comparison music signals with different PAPR and normalized in the range (-1,1) were used. For each signal the percentage of clipping was derived following the next equations:

$$\text{PAPR}_{dB} = 20\log_{10}\left(\frac{max_{1\leq i\leq n}|x_i|}{x_{RMS}}\right) = 20\log_{10}\left(\frac{max_{1\leq i\leq n}|x_i|)}{\sqrt{\frac{1}{N}\sum_1^N x_i^2}}\right) \tag{4.1}$$

$$\text{clipping percentage} = \frac{\sum_{i=1}^N \mathbf{1}\{x_i > th\}}{N} \tag{4.2}$$

Figure 4.19 shows the relations between the different systems. As can be seen, without attack and release time the current systems work as ideal and there are no samples above the threshold. While for the ANN some of them go above. The significant fact is that for signals with less PAPR and a high percentage of clipping, the ANN works better (signals with lower PAPR tend to clip more often since the distance between the peaks and the average of the signal is smaller. Therefore, it is likely to reach the peak and go above the threshold). This behaviour is not common for linear systems where the output of the limiter follows the same trend as the input, clipping more for smaller PAPR signals.

On the other hand, applying 0.1 ms attack time and 35 ms of release, the ANN

limiter works better than the hard limiter, but worse than the soft limiter. The look-ahead limiter due to its look-ahead can predict always when the signal is going to clip and the output never clips.

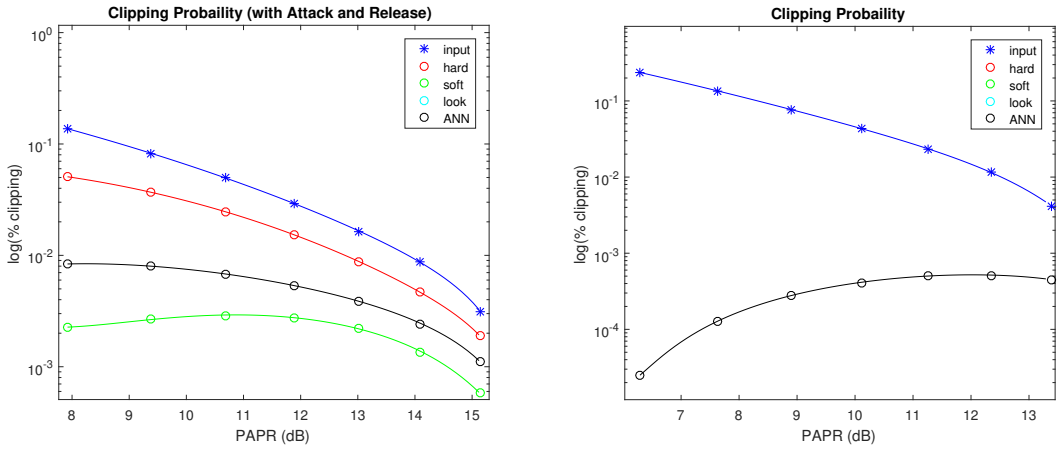


Figure 4.19: PAPR and clipping percentage comparison. On the left, the realisations with attack and release time measured for the different systems. On the right, without attack and release. Attack time = 0.1 ms and Release time = 35

4.5.2 Harmonic Distortion

The Spurious-Free Dynamic Range (SFDR) is the ratio of the fundamental signal to the strongest spurious signal in the output. This measurement could tell how much distortion the system adds to the signal with limiting it.

Different tones were generated in order to test the different response of the systems for different frequencies. Some of the simulations are shown in Appendix C. The result is summarised in Table 4.2. The SFDR level shown is the average for different frequencies in the audible spectrum.

Analysing the effect of the attack and release is another important factor since systems with attack and release time have softer transition in the attenuation signal. This makes the system generate less harmonic distortion when attenuating, but at the same time the system is not fast enough to recover and it modifies the dynamic range of the signal too much. In this case, the Root Mean Square Error (RMSE) of the signal increases. Then, another simulation with attack and release time was carried out, some graphs are shown in Appendix D and the results are also summarised in Table 4.2.

	SFDR (dB) (no attack and release)	SFDR (dB) (attack and release)
Hard Limiter	50,8	93
Soft Limiter	49,6	95
LookAhead Limiter	92	97
ANN Limiter	57,6	96
	RMSE (no attack and release)	RMSE (attack and release)
Hard Limiter	0.052	0.076
Soft Limiter	0.074	0.120
LookAhead Limiter	0.082	0.084
ANN Limiter	0.058	0.088

Table 4.2: Comparison of the distortion generated by each system. Above the results for the harmonic distortion and below the RMSE. These results were obtained simulating thousands of signal and averaging the error. Some examples of this simulation are shown in Appendixes C and D.

When the SFDR is bigger, the distortion is smaller. In such case, the difference between the power of the original signal and the power of the harmonics generated by the system is higher. This means, with attack and release the systems work better generating less distortion than not using it. Actually, with attack and release, the distortion can be neglected. For the look-ahead limiter is the same in both cases since the look-ahead allows to predict the signal and keep the original shape. If the other three systems are compared, ANN generates the lowest harmonic distortion. Hard limiter and soft limiter have similar higher distortions levels.

Regarding the dynamic range and Root Mean Square (RMS) power, the release time could affect considerably the effect of the attenuation. Longer release time could keep the attenuation level for so long after it is not needed. This effect can be seen in Table 4.2. RMSE tells the deviation of the differences between the input x and the output y. This number was obtained following the next equation:

$$\text{RMSE} = \sqrt{\frac{\sum(x_i - y_i)^2}{N}} \quad \text{with} \quad i = 0...N \tag{4.3}$$

The hard limiter without attack and release tells the smallest possible error since it makes equal to the threshold all the samples above it. The soft limiter makes the same with the soft transition, therefore it has more error. More error means less RMS (average) power in the output signal since it is more attenuated. The look-ahead limiter has the highest error since it is attenuating more time due to the prediction ability. It starts attenuating before it reaches the threshold. In this case, the delay is 35 samples look-ahead. The ANN limiter without attack and release has the closest value to the smallest possible error.

Adding the attack and release times make the results for the four systems more

similar. Due to the time reaction, the systems starts to attenuate more than necessary and the error increases. The important fact is how much they change in relation to the first cases without attack and release. It can be seen that the look ahead only increases in 0.002, while the hard limiter 0.024, soft limiter 0.046 and the ANN limiter 0.030 % . This means that hard, soft and ANN limiter especially are more sensitive to the release time reaction and will apply more attenuation after needed.

Putting together both results, the harmonic distortion and the error, the ANN limiter works better than hard and soft limiters regarding distortion and also it has less error than the soft limiter. It also has less RMSE than the look-ahead limiter what means more RMS output power. The results with attack and release are very similar. It is important to note that the look-ahead system adds 35 samples latency to the system due to the look-ahead, which is by itself an undesirable effect that the ANN limiter does not cause.

5
Conclusion

After the results presented above, a summary is due and some conclusions can be drawn.

The thesis work culminates in an implementation of an original non-deterministic signal limiting algorithm, which is benchmarked against other widespread deterministic methods. The originality resides mostly in the choice of a neural network-based method, which by itself opens new possible research paths.

A number of variations of the same algorithm can be conceived, and a number of different application of the same kind of algorithms, besides "limiters" for the purpose defined in the introduction, can be considered.

No method is the absolute best in all regards. Different algorithms have shown strengths and weaknesses: this is about latency, distortion, clipping performance, i.e. parameters affecting the listener's perception, but also about implementation complexity and computational cost.

So, how to chose the best one out of many options? As usual when it is about audio signal processing, complying with pre-determined requirements is not necessarily enough: psychoacoustics has a key role, how the listener perceives the sound quality is still the most important overall benchmark parameter. Therefore, the optimal algorithm is picked after a listening test, which inevitably adds a degree of subjectivity to the final evaluation.

No matter what, the opinion I share with the experienced developers of audio systems I had the opportunity to work with, is that the method herein exposed operates in a satisfactory manner, and introduces enough advantages to be considered interesting for future product design.

6
Future Work

There has been a limitation in time and resources, longer trainings and more parameters and methods combinations could be done in order to get more accurate results and understanding of the system.

Some procedures could be carried out in order to see how the ANN works for the system. For example, it could be interesting to see where are the active neurons for each sample and which is the complexity of the of the activation patterns for different signals.

Some extra algorithms for the GA like dynamic probability mutation. This probability could vary based on the diversity of the individuals and stimulate having new individuals. The new individuals could lead to better results and less probability to get stuck in a local minima.

Another training methods and approach could be tested for the ANN, like backpropagation or Recursive Neural Networks.

This result includes only one system implementation per threshold. This means that it is required different systems obtained for different trainings if the user wants to modify the threshold of the ANN limiter. Some research in this area could improve the system and make more flexible for different thresholds.

A deeper study of the non-linear behaviour of the system and what it adds. And also, how to take more advantage of the new dimension space that the ANN can handle, instead of only time and frequency in the current systems. For example, the extra Noise Gate block proposed in this thesis could be the first step to make a differentiation between two different states, the linear and the attenuated.

The next step for the result obtained could be to test it in a real-time system, as a Digital Signal Processor.

Bibliography

[1] Udo Zolzer. (2002) DAFX: Digital Audio Effects. John Wiley and Sons, Ltd.

[2] Sophocles J. Orfanidis. (1995) Introduction to Signal Processing. Prentice Hall.

[3] J.P. Mackenzie. (1995) Chaotic Predictive Modelling of Sound. ICMC Proceedings.

[4] Abdelhakim Dahimene, Mohamed Noureddine and Aarab Azrar. (2007) A Simple Algorithm for the Restoration of Clipped Speech Signal. Informatica 32 (2008) 183–188 .

[5] Limiter ratios figure. Creative Commons Licence. `https://en.wikipedia.org/wiki/Dynamic_range_compression#/media/File:Compression_ratio.svg`

[6] Soft-knee figure. Creative Commons Licence. `https://commons.wikimedia.org/wiki/File:Compression_knee.svg`

[7] Hiten N. Mistry (2003) Implementation of a Peak Windowing Algorithm for Crest Factor Reduction in WCDMA. Simon Fraser University Master Thesis.

[8] Vinay Reddy N., Arthy Suganthi Kani J., Ajay Kumar D., (2015) Peak Cancellation Crest Factor Reduction Technique for OFDM signals. IMPACT: International Journal of Research in Engineering Technology

[9] Neuron figure. Creative Commons Licence. `https://askabiologist.asu.edu/neuron-anatomy`

[10] Mattias Wahde (2008) Biologically inspired optimization methods.An introduction

[11] Synapse figure. Creative Commons Licence. `https://commons.wikimedia.org/wiki/File:SynapseSchematic_lines.svg`

[12] Dhawale A, Bhalla US. (2008) The network and the synapse: 100 years after Cajal. HFSP Journal.

[13] D. O. Hebb. (1949) The Organization of Behavior.Wiley.

[14] W. McCulloch andW. Pitts (1943) Alogical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 7:115–133.

[15] Simon Haykin (2001) Neural Networks, A Comprehensive Foundation. Prentice Hall.

[16] David Kriesel (2005) A Brief Introduction to Neural Networks.

[17] P. Sibi, S. Allwyn Jones, P. Siddarth (2013) Analysis of Different Activation Functions Using Back Propagation Neural Networks. SASTRA University, Kumbakonam, India. Journal of Theoretical and Applied Information Technology

[18] Perceptron figure. Creative Commons Licence. `https://commons.wikimedia.org/wiki/File:Rosenblattperceptron.png`

[19] ANN figure. Creative Commons Licence. https://commons.wikimedia.org/wiki/File:Neural_Network.gif
[20] DNA figure. Creative Commons Licence. https://upload.wikimedia.org/wikipedia/commons/e/e1/Chromosome-DNA-gene.png
[21] Darrell Whitley. (1994) A Genetic Algorithm Tutorial. Statistics and Computing (1994) 4, 65-85
[22] Michael Vose (1999). The Simple Genetic Algorithm: Foundations and Theory. Cambridge, MA: MIT Press.
[23] R. Sathya, Annamma Abraham (2013) Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification. (IJARAI) International Journal of Advanced Research in Artificial Intelligence, Vol. 2, No. 2.
[24] John H. Holland (2005) Genetic Algorithms. Computer programs that "evolve" in ways that resemble natural selection can solve complex problems even their creators do not fully understand.
[25] GA reproduction figure 1. Creative Commons Licence. https://en.wikipedia.org/wiki/Meiosis#/media/File:Meiosis_Overview_new.svg
[26] GA reproduction figure 2. Creative Commons Licence. https://commons.wikimedia.org/wiki/File:Computational.science.Genetic.algorithm.Crossover.Cut.and.Splice.svg
[27] D Nagesh Kumar (2010) Optimization Methods: Optimization using Calculus-Stationary Points. IISc Bangalore. Lecture Notes
[28] Paul M. Parker (2002) Newton's Law of Universal Gravitation. Dept. of Physics, Mich. State Univ
[29] J.P.Mackenzie (1995) Chaotic Predictive Modelling of Sound. School of Electronic and Manufacturing Systems Engineering. University of Westminster.
[30] Eugen Diaconescu (2008) The use of NARX Neural Networks to predict Chaotic Time Series. Communications and Computer Science Faculty. University of Pitesti.

A

Prediction

During the research and implementation of the model some of the results were not enough to limit the samples on time for quick transients, therefore an extra approach was considered. Some advance was done in non-linear limitation, but since the system does not have look-ahead, an extra network could try to predict future samples.

A simple Recurrent Neural Network was trained in parallel in order to predict next samples, this network is shown in Figure A.1. It was trained with different combinations of neurons, activation functions (linear, tangential, logistic...)[17] and different types of data.

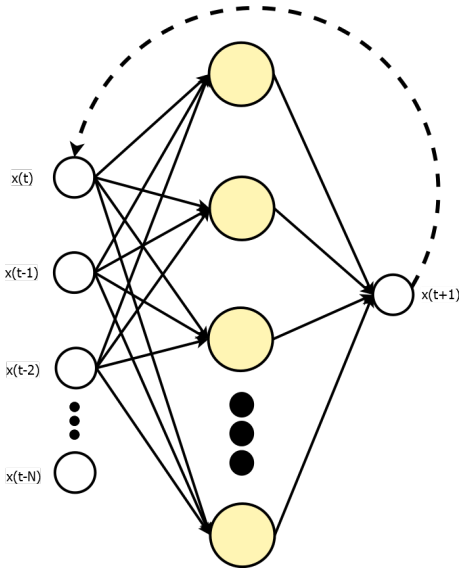


Figure A.1: Artificial Neural Network general application used for prediction.

There is some prior art based on chaotic series prediction and non-linear dynamics [29] [30], and the results for a simple signal like simple tones could be more or less acceptable, but for complex sounds did not work at all. However, in this thesis, it has been tried to predict some samples ahead and the outcome resulted in the conclusion that this approach was out of the scope of the thesis. It is needed a totally different approach if the prior art is followed and just applying the aforementioned topology did not fulfil the requirements.

Some results are shown in Figure A.2. With one sample look-ahead prediction

the values are stable but a high percentage of the samples seems to be delayed and therefore the amplitude of the signal that has to trigger the threshold in order to know when it is going to clip is wrong. If one has a look of the examples for 2 or 5 samples look-ahead the prediction becomes more and more unstable.

One of the constraints in this system to make it works as the number of previous samples needed as an input to the network. As the number of the inputs increases the number of computations and training time increases exponentially, this makes the system less efficient when implementing it in a real product. Because of this reason, after some considerable amount of time, this approach was not considered anymore.

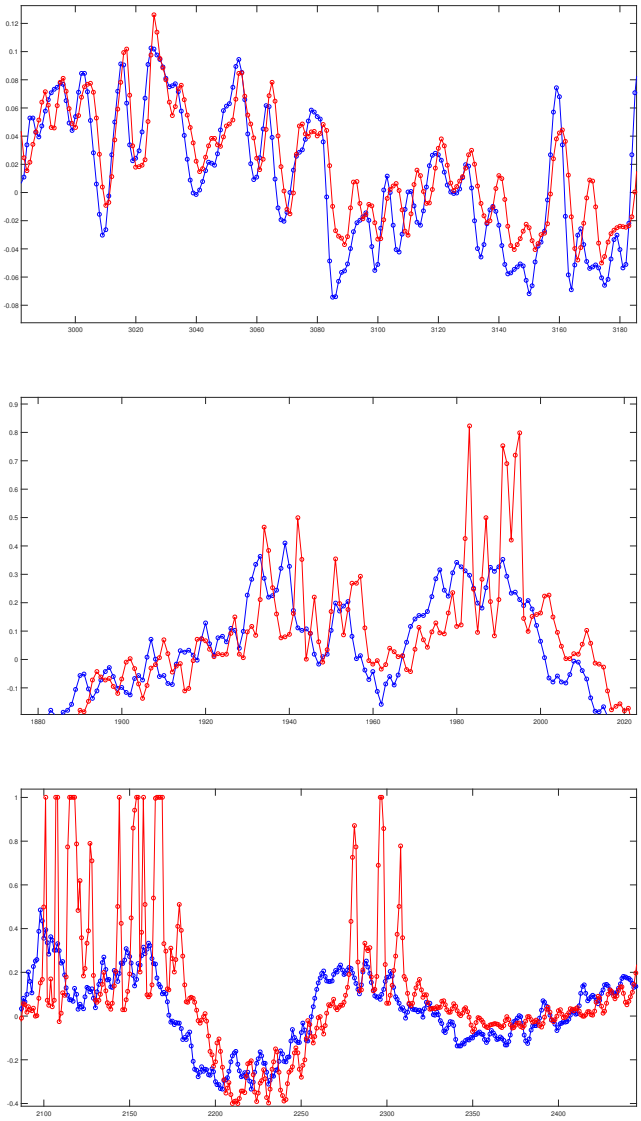


Figure A.2: Results of prediction for 1 sample look-ahead (up), 2 samples look-ahead(middle) and 5 samples look-ahead (down). The input signal is in blue and the prediction in red.

B

Number Of Neurons Comparison

In and out compression/limitation relation for different combinations of penalties and number of neurons in the hidden layer (blue is input, red output and black is the threshold):

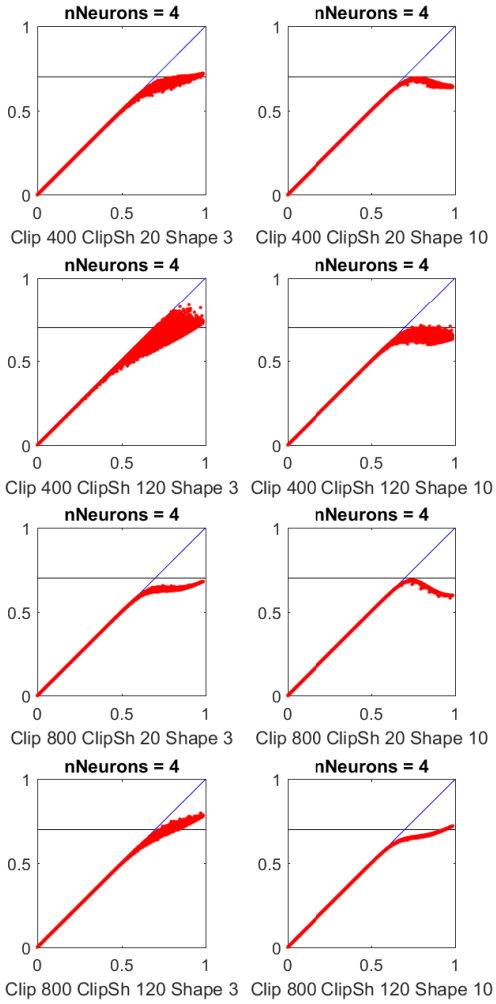


Figure B.1: Neural network with 4 neurons in the hidden layer and different penalty combinations.

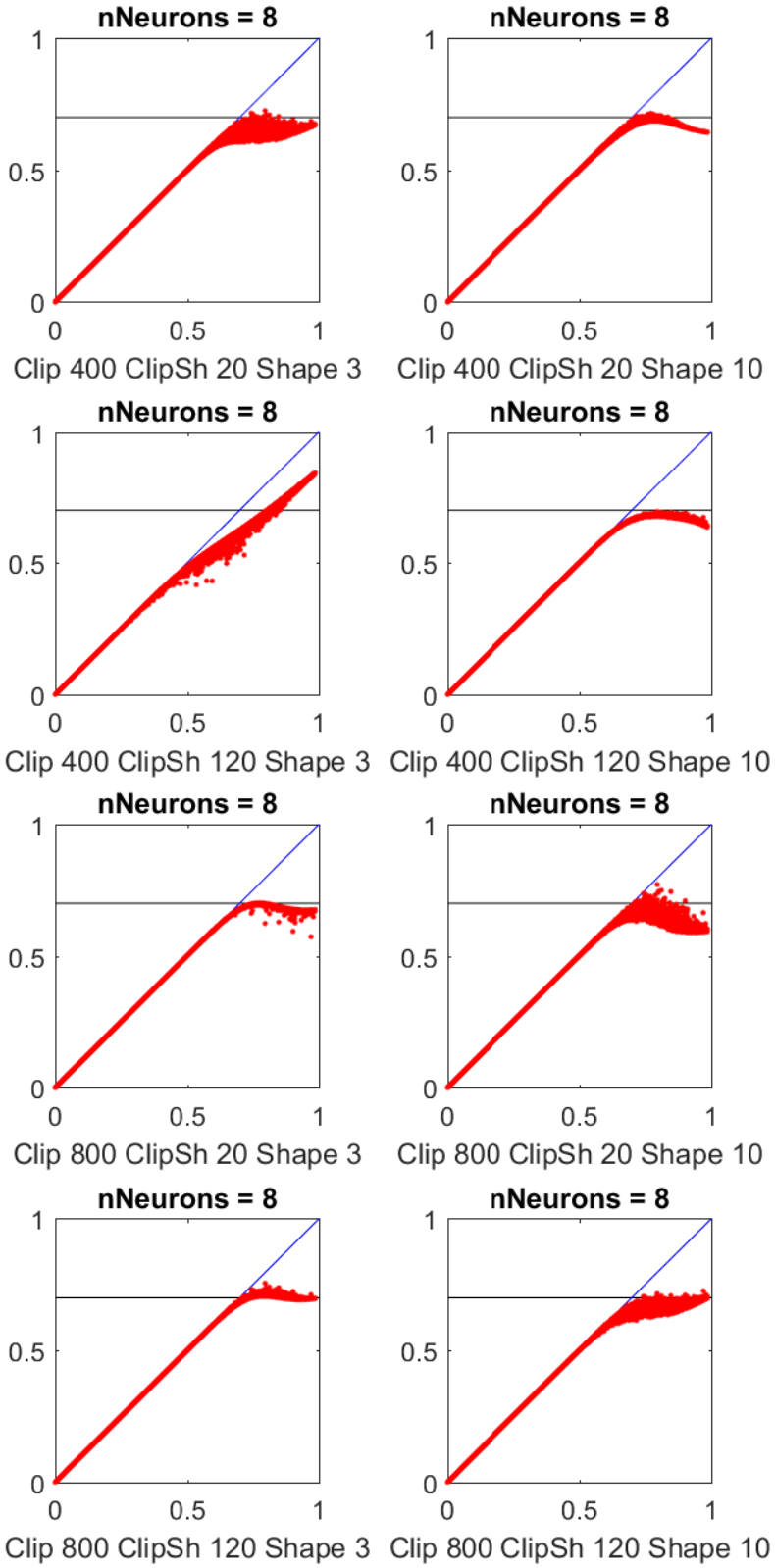


Figure B.2: Neural network with 8 neurons in the hidden layer and different penalty combinations.

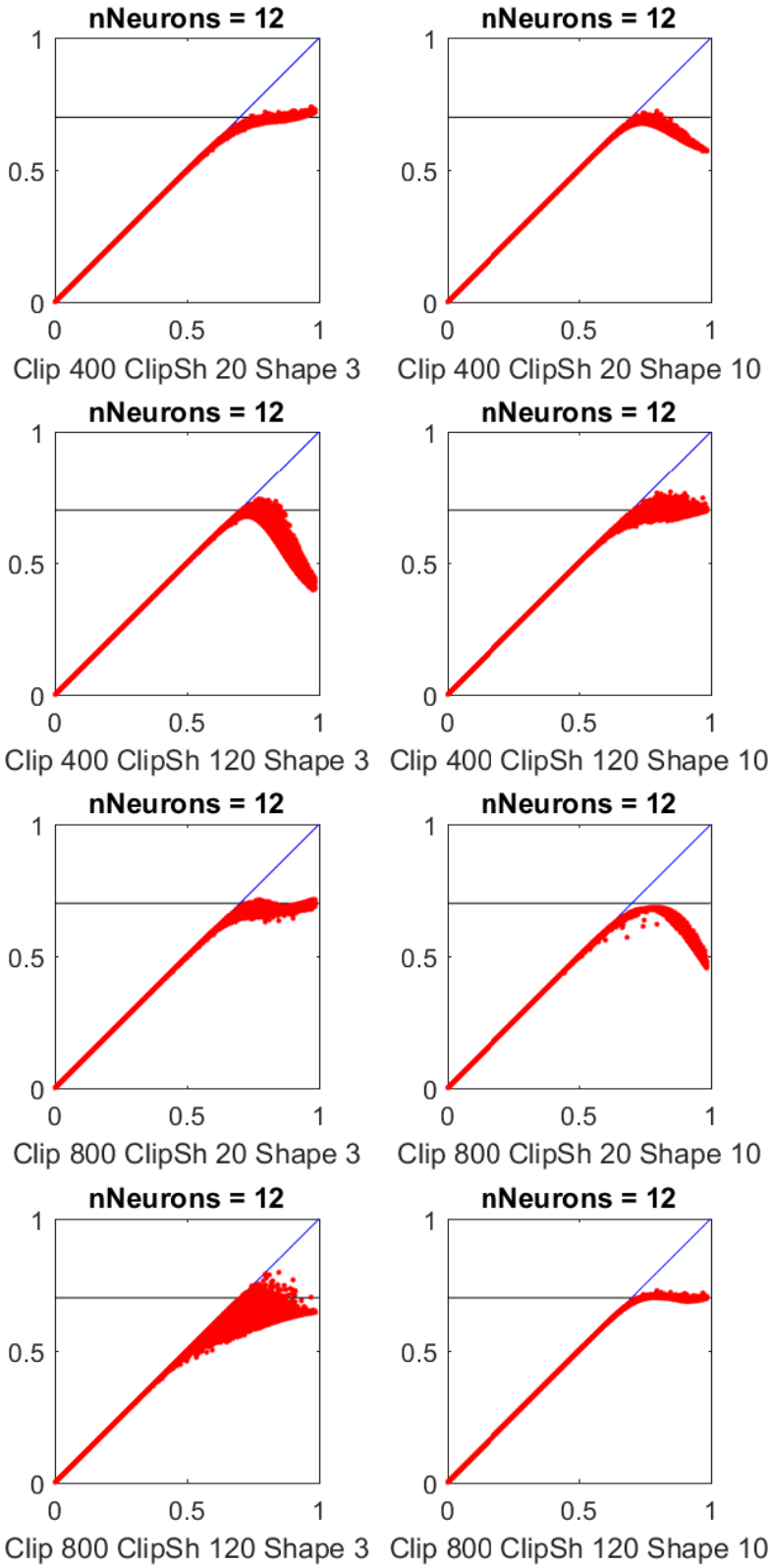


Figure B.3: Neural network with 12 neurons in the hidden layer and different penalty combinations.

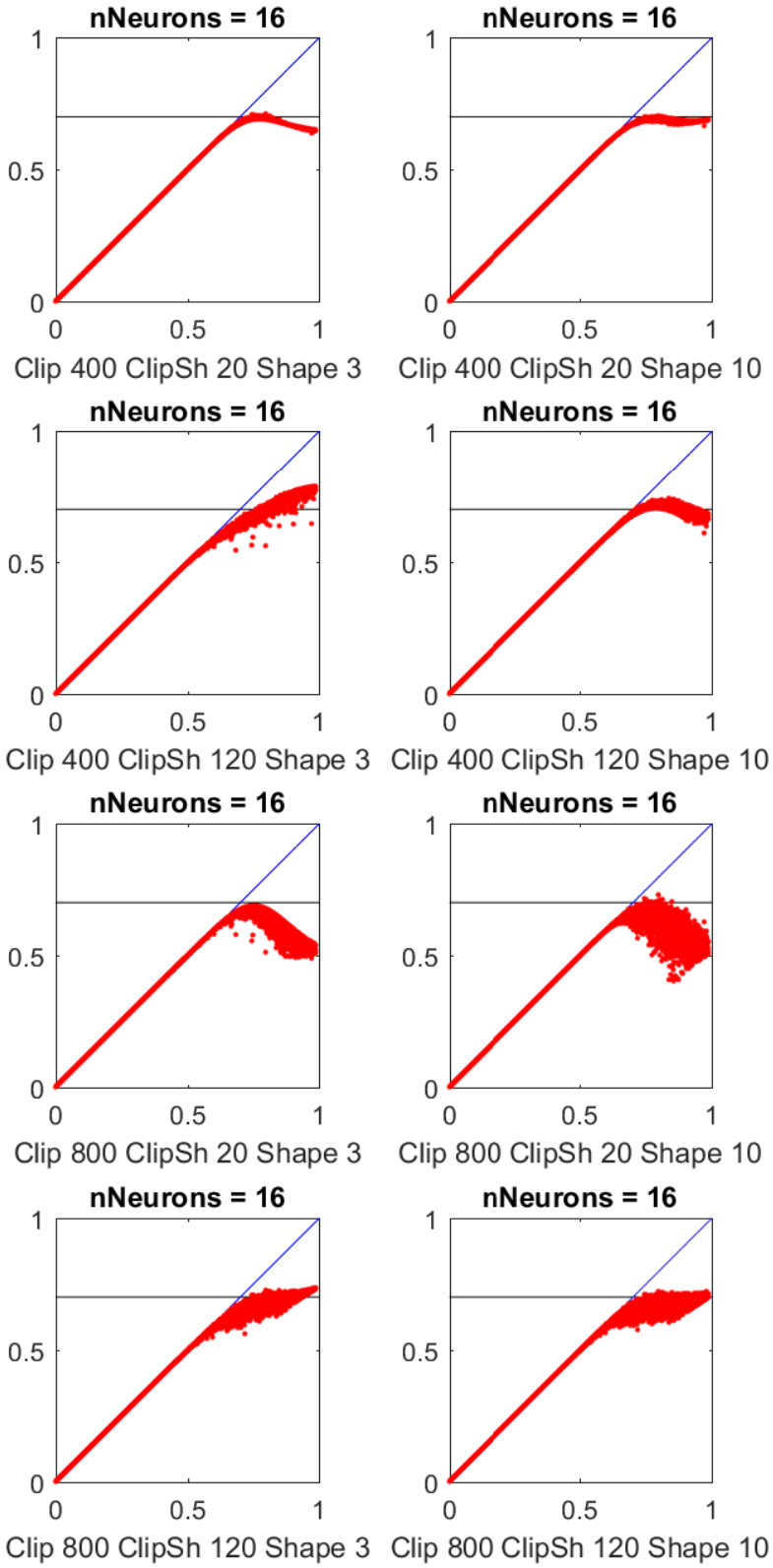


Figure B.4: Neural network with 16 neurons in the hidden layer and different penalty combinations.

C

Harmonic Distortion

Spurious-Free Dynamic Range simulations:

Time and frequency domain input and output signal for different systems without applying attack and release time:

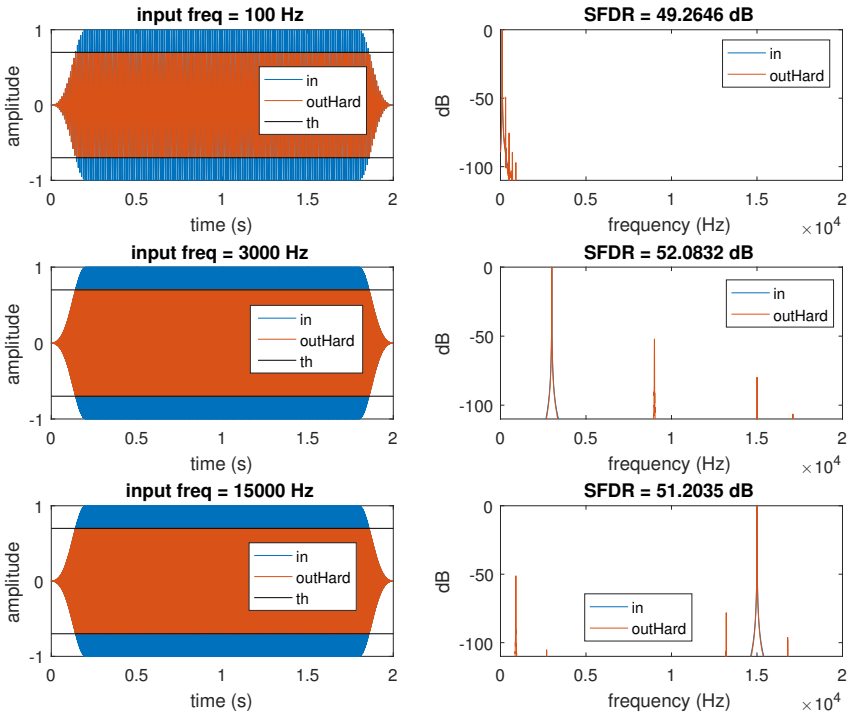


Figure C.1: Hard limiter response for a different frequencies burst signal and the harmonic distortion generated for each tone. Attack and release time is not applied.

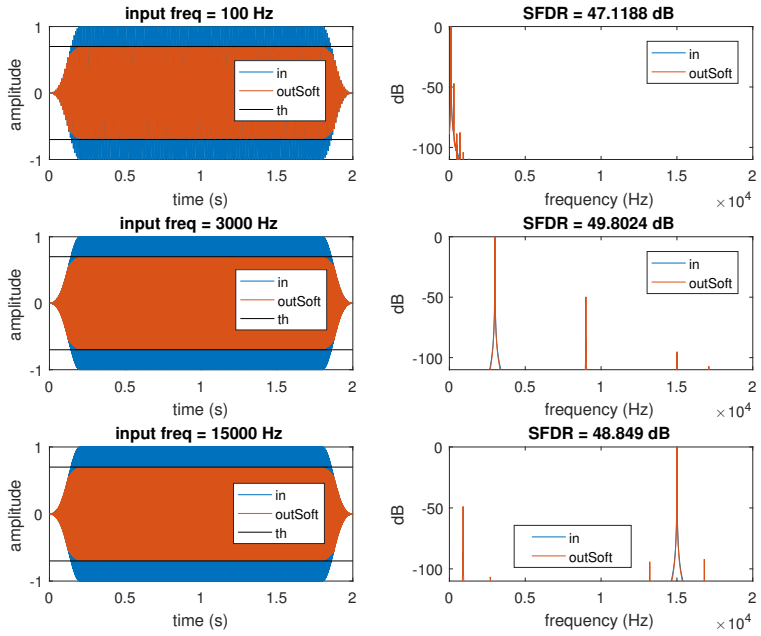


Figure C.2: Soft limiter response for a different frequencies burst signal and the harmonic distortion generated for each tone. Attack and release time is not applied.

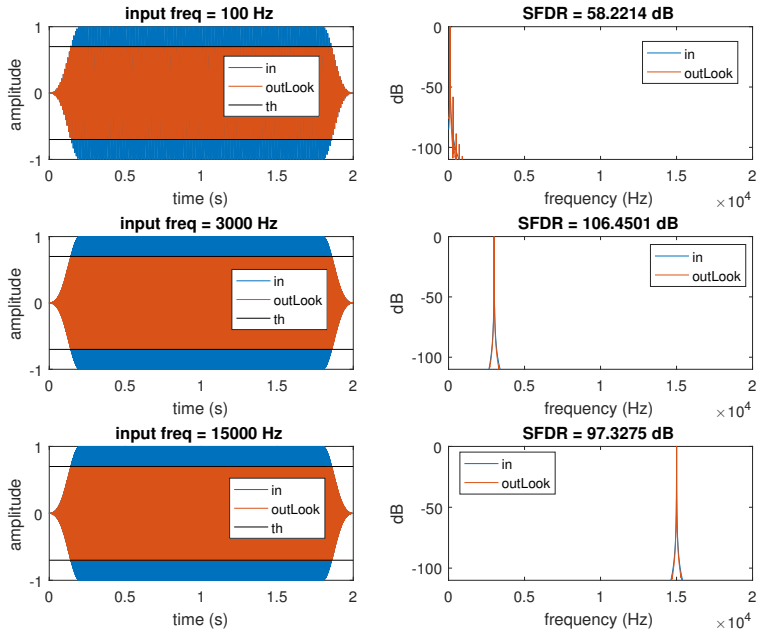


Figure C.3: Look-ahead limiter response for a different frequencies burst signal and the harmonic distortion generated for each tone. Attack and release time is not applied.

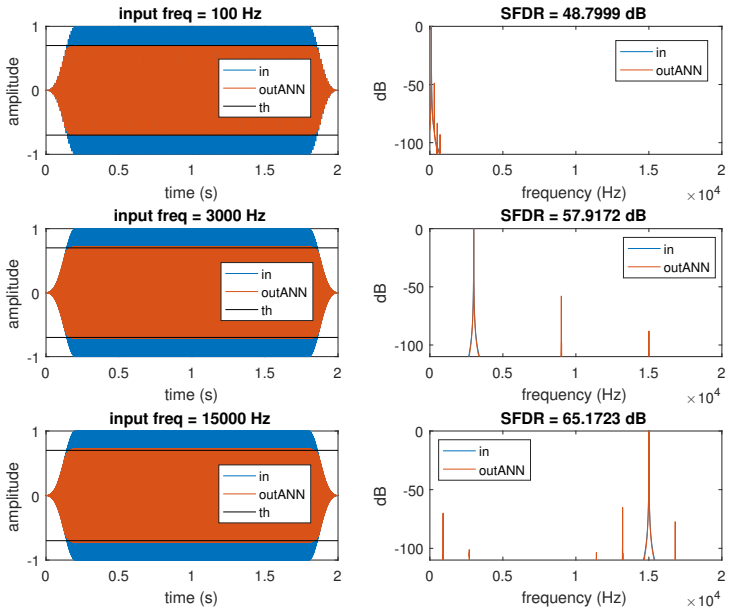


Figure C.4: ANN limiter response for a different frequencies burst signal and the harmonic distortion generated for each tone. Attack and release time is not applied.

Applying attack and release time:

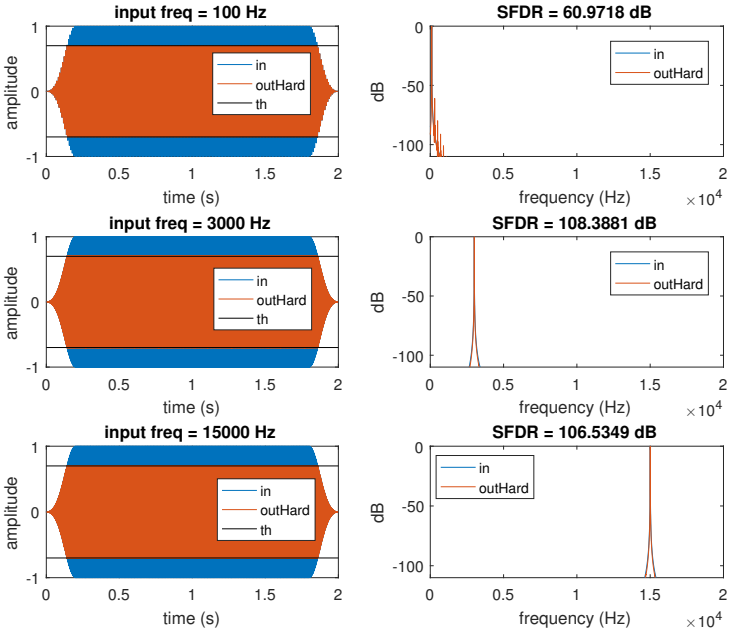


Figure C.5: Hard limiter response for a different frequencies burst signal and the harmonic distortion generated for each tone. Attack and release time is applied.

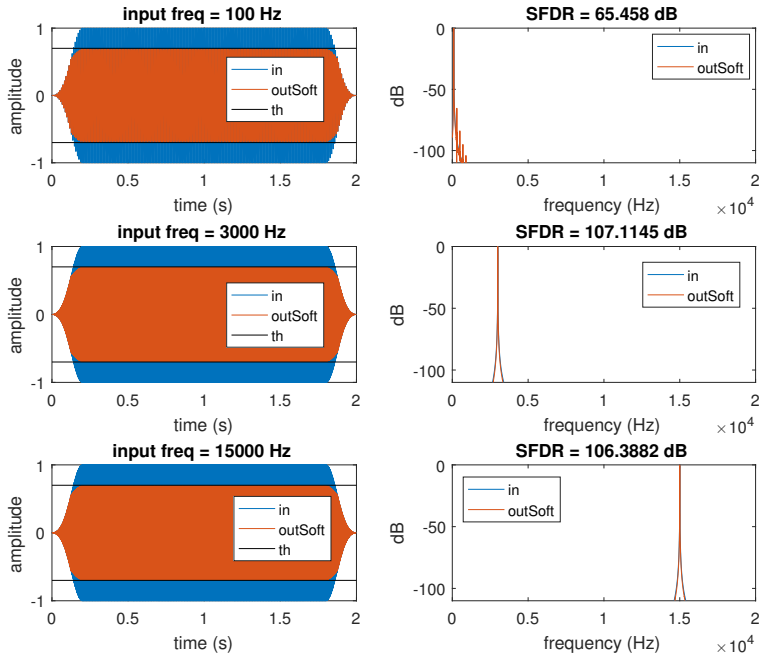


Figure C.6: Soft limiter response for a different frequencies burst signal and the harmonic distortion generated for each tone. Attack and release time is applied.

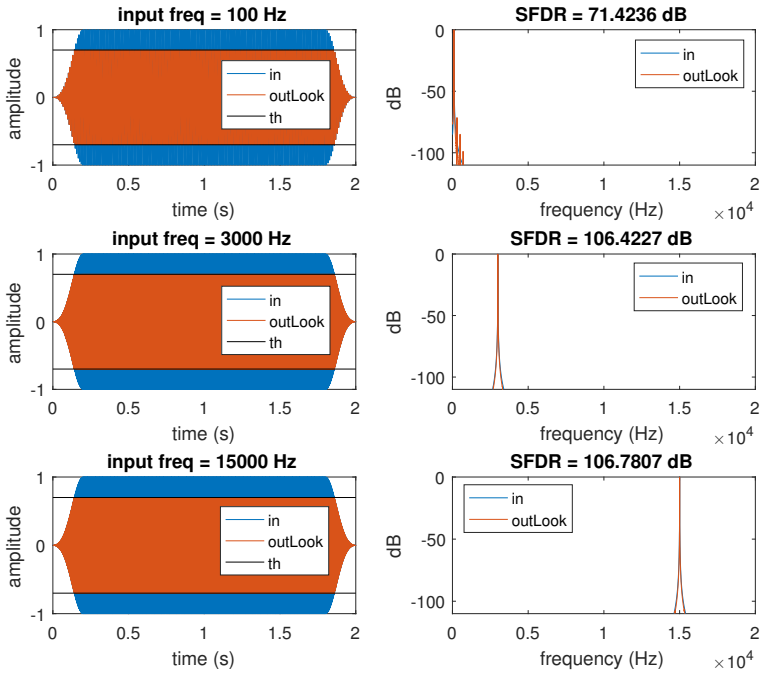


Figure C.7: Look-ahead limiter response for a different frequencies burst signal and the harmonic distortion generated for each tone. Attack and release time is applied.

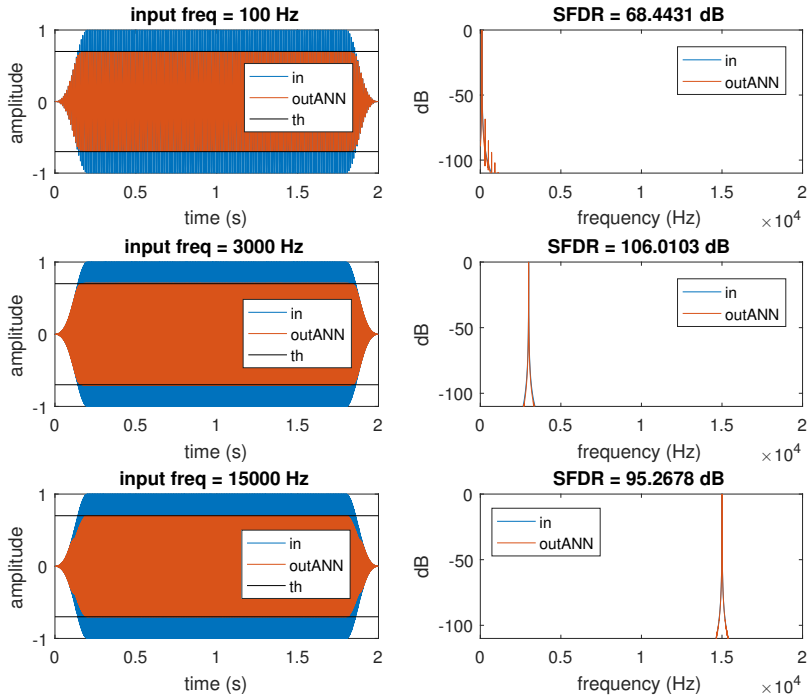


Figure C.8: ANN limiter response for a different frequencies burst signal and the harmonic distortion generated for each tone. Attack and release time is applied.

D

Dynamics and RMS Saving

Simulation carried in order to show how the release time change the dynamics of the signal.

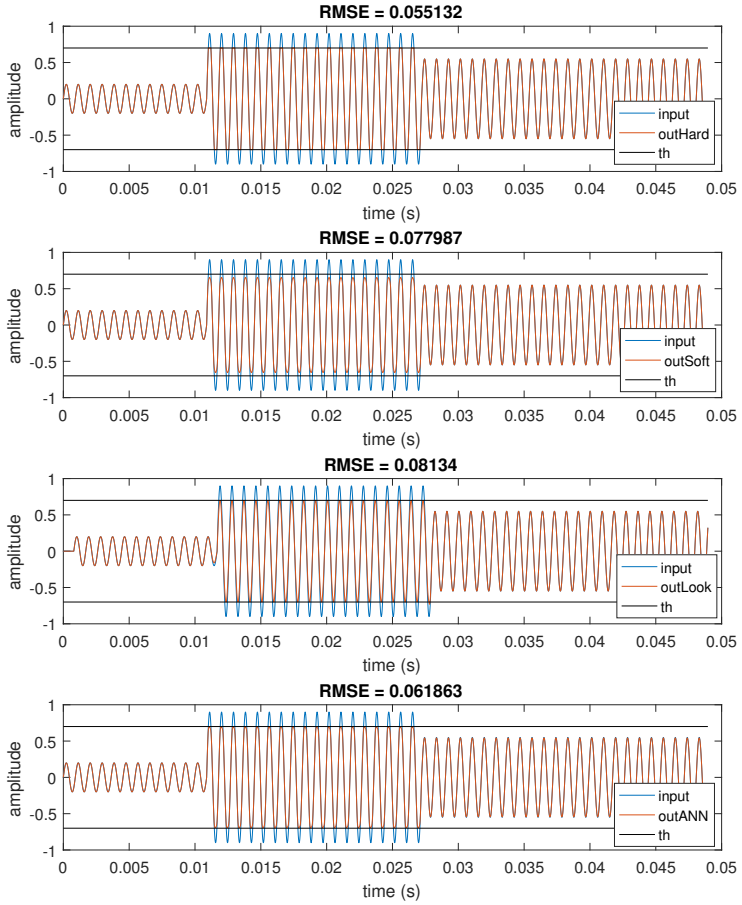


Figure D.1: Time domain input and output signal for different systems without applying attack and release time. From the first (up) to the last (down): Hard limiter, Soft limiter, Look-ahead limiter and ANN limiter

Time domain input and output signal for different systems applying attack and

release time:

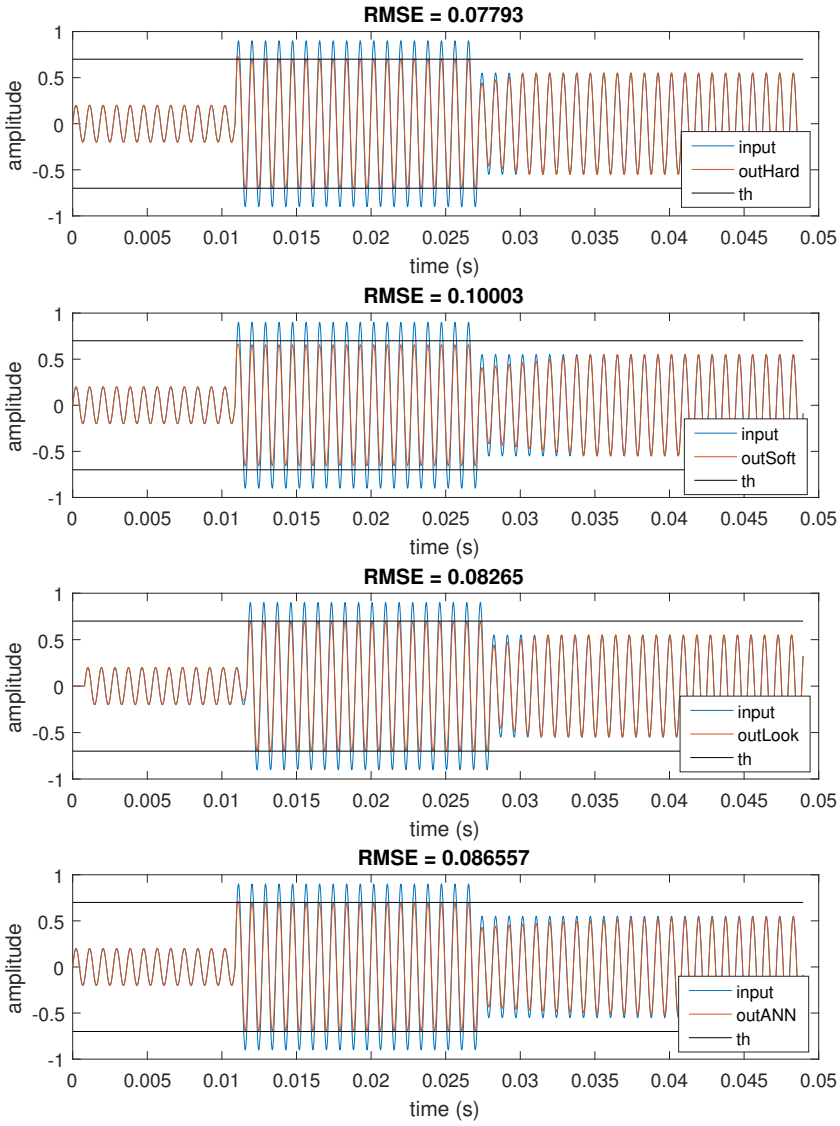


Figure D.2: Time domain input and output signal for different systems applying attack and release time. From the first (up) to the last (down): Hard limiter, Soft limiter, Look-ahead limiter and ANN limiter