POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

AI Powered Pick-up

Tesi di Laurea Magistrale in
Music and Acoustic Engineering

Author: **Emanuele Voltolini**

Student ID: 944835
Advisor: Prof. Fabio Antonacci
Co-advisors: Sebastian Gonzalez
Academic Year: 2021-22

Abstract

With the advent of digital technologies in the music industry, the modeling of analog equipment such as guitar amplifiers, distortion pedals and other effects has assumed a central role. As a matter of fact, this makes expensive analog technologies more affordable, as well as guaranteeing a sound quality comparable with the original one.

In the last years, the use of deep neural network was heavily introduced in sound modeling as a valid alternative to more classic DSP (Digital Signal Processing). Specifically, neural networks such as RNN (Recurrent Neural Network) and WaveNet have been adopted to reach the best results. Furthermore, these architectures are mainly employed in modeling analog devices related to electric guitar sounds.

This thesis work is aimed at black-box modeling acoustic guitar pick-up - microphone sound using a RNN neural network with an LSTM (Long-Short Term Memory) unit. For this purpose, we create a training dataset composed by pairs of microphone and pick-up acoustic guitar recordings. Furthermore, we studied a loss function that could fit our task. Finally, we evaluate the results in terms of ESR (Error to Signal Ratio) and give also a perceptual evaluation from the author perspective.

We conclude that the model has shown its ability of following the trend of the target microphone signal in time domain, given as input the pick-up one. However, the model is not able to capture properly the high frequencies components of the spectrum, which are attenuated for frequencies greater than 3 kHz. In order to overcome this issue, we proposed a solution based on the sum of two different models output. This new audio present more energy in the upper frequencies components.

Within said, the final output is not good enough to have the same tone of the target one. We hope this thesis could be a starting point in this research area, which would bring new tools in music field.

Keywords: deep learning, recurrent neural network, black-box sound modeling, acoustic guitar

Abstract in lingua italiana

Con l'avvento del digitale nell'industria musicale, la modellazione virtuale di strumenti analogici come amplificatori, pedali di distorsione e altra effettistica ha assunto un ruolo centrale. Questo ha permesso di rendere le costose attrezzature analogiche più facilmente accessibili, garantendo una qualità sonora comparabile con l'originale.

Negli ultimi anni, nella modellazione virtuale si è sempre più affermato l'utilizzo del deep learning come valida alternativa ai classici metodi DSP (Digital Signal Processing). Nello specifico i risultati migliori sono stati ottenuti da reti neurali quali RNN (Recurrent Neural Network) e Wavenet. La modellazione in questo campo si concentra prevalentemente su suoni legati alla chitarra elettrica.

Il lavoro di questa tesi punta a modellizare con un approccio scatola-nera la relazione tra il suono registrato dal pick-up di una chitarra acustica e quello di un microfono professionale, utilizzando una RNN con un'unità LSTM (Long-Short Term Memory). Abbiamo quindi creato un dataset composto da coppie di audio ottenute registrando simultaneamente la chitarra acustica dal pick-up e dal microfono. Inoltre è stata studiata una funzione di perdita che fosse consona al nostro scopo. Infine, abbiamo valutato i risultati ottenuti in termini di ESR (Error to Signal Ratio), fornendone una valutazione percettiva personale.

Abbiamo concluso che il modello proposto è in grado di seguire l'andamento del segnale del microfono (target) nel dominio temporale usando come input il segnale del pick-up. Dal punto di vista spettrale la rete neurale proposta non è in grado di catturare le componenti ad alta frequenza del segnale, che risultano attenuate per frequenze maggiori di 3 kHz. Abbiamo quindi proposto una soluzione basata sulla somma di due audio ottenuti da due modelli differenti, in questo modo l'output finale presenta più energia nelle alte frequenze.

Con questo detto, il timbro ottenuto nell'audio finale non può essere considerato indistinguibile da quello del microfono. Ci auspichiamo questa tesi possa essere un primo passo in questa nuova area di ricerca, portando innovazione in ambito musicale.

Parole chiave: deep learning, rete neurale ricorrente, modellazione scatola-nera del suono, chitarra acustica

Contents

Introduction

Nowadays digital techniques in music industry and production are becoming more and more popular alongside the classic analog ones or in some cases directly replacing them. One of the most innovative research area in this sense is represented by the analog audio effect modeling, which exploits DSP (Digital Signal Processing) or modern deep learning techniques to create digital models of analog amplifiers, pedals and other effects. VST (Virtual Studio Technology) plugins are an example of direct implementation of these studies and they consist of software emulations of various sound recording equipment and instruments. In addition VST plugins make use of DSP in order to simulate specific equipment. Furthermore, their minimum required computational power permits to run them on a common laptop.

In analog audio effect modeling, one of the main non deep learning based methodologies relies on WDFs (Wave Digital Filters), which are a particular kind of digital filters based on physical modeling principles. WDFs are specially well-suited for modeling block-based networks such as electrical circuits due to their modular characteristic, as a matter of fact some implementations are the real-time simulation of tube amplifier stage [1] and distortion pedal modeling [2]. An alternative non deep learning based approach is built on block-oriented Wiener-Hammerstein model, which is a parametric model adaptable to many distortion effects [3]. Both methods present difficulty in handling multiple nonlinearities and they are often demanding from a computational point of view.

In order to overcome these issues, deep learning based models are introduced in analog audio effect modeling. Deep learning is part of a broader family of machine learning methods based on artificial networks, whose aim consist in learning the relation between what is given as an input and the target data through a large amount of data. As a consequence this technique allows to learn the nonlinear relationship of analog audio effects, such as distortion pedal and guitar amplifiers. As an example of this methodology, in Wright's work [4] two neural network models which emulate two different guitar amplifiers are presented. The paper shows how good results can be achievable with these new models, explaining also the possibility in terms of real-time applications. Another interesting research in this field is brought by Steinmetz and Reiss [5], which carries on

Wright's work applying a new model based on TCNs (Tempoal Convolutional Networks) on a more complex audio effects (dynamic range compressor). It is shown how this new architecture is more efficient from a computational effort making the model particularly suitable for real-time implementations.

A new research area close to analog audio effect modeling is distortion removal, which is introduced by Imort and Fabbro [6]. Their aim consists in removing distortion and clipping applied to guitar tracks for music production. The innovation is represented by the implementation of different models with respect to traditional ones (such as Recurren Neural Network and WaveNet). The best performing architecture is the Demucs which is an autoencoder with a BLSTM (Bidirectional Long Short Term Memory) used in music source separation. Obtained results achieve a great step forward in the state of the art, leading to a new approach in this field of study.

All these researches lead to new digital implementations of analog devices, opening new possibilities for musicians and more in general in the music studio production. Even if some of the models can not reach the sound quality of the physical analog devices, the research makes great strides forward bringing more affordable and more easily usable tools.

In this thesis work we apply deep learning to acoustic guitar pickup - microphone black-box sound modeling. Since this is a new field of research, we selected one of the most used deep neural network in black-box sound modeling, a RNN (Recurrent Neural Network) with an LSTM (Long Short Term Memory) unit [4]. In order to test this model, we had to record a new dataset, which is used in the loss evaluation and in the training process.

The dataset was made using a single acoustic guitar, which was recorded simultaneously from its piezo-pickup and a professional SM57 microphone placed in front of the instrument. In this way, we obtained a pair of signal for each recording: the first correspond to the input of the network and the other recorded with the microphone to the target. We tried to collect the majority of guitar playing style to have the greatest possible variability.

Before feeding the network with the recorded signals, we studied the loss function implemented by A. Wright [4] to see if it could fit our task. In order to reach our aim, we built a fake signal which tried to emulate what the network should do with the original one, performing a comparison of the loss function values between the "fake signal" and the raw one.

Once the model was tested, we trained it feeding the network with the recorded data. Since one single training can even last a day, a large part of this thesis work consisted of

tuning all network parameters to reach the best possible output results.

We studied the best performing model in terms of ESR (Error to Signal Ratio) both in time and frequency domain. Furthermore, we presented a comparison between different models based on the ESR values taking into account also the author perceptual evaluation.

Our thesis work can be considered as an addition to black-box sound modeling application in the current state of the art, which is mainly focused on modeling guitar amplifier and effects pedals [1, 4, 5, 7, 8]. Our research has sufficient characteristics to open new possibilities in the acoustic guitar field, bringing new techniques in studio recordings and live music. For example, a possible application could be the creation of a VST plugin which exploits the studied model.

This thesis is organized as follows. In Chapter 1 we introduce a general description of deep learning and neural networks. Moreover, we also show the theory behind the most used model in the audio field. Furthermore, since no previous work on our thesis task has been made, we describe most advanced studies in similar fields for completeness. In particular a comparison between non deep learning model and neural network based ones is made. In Chapter 2 we present the used neural network model, the data acquisition procedure and the applied pre-processing. Moreover, a description of the obtained signals and the loss function implemented is given. In Chapter 3 we show the network output results both in time and frequency domain, doing a comparison between different models. Finally, Chapter 4 contains the thesis work conclusions and further suggestions for future developments.

1 | State of the art

In this chapter we are going to introduce the state of the art. Deep learning is part of machine learning family methods based on artificial neural networks with representation learning. Unfortunately no previous research on deep learning applied to acoustic guitar pickup - microphone black-box sound modeling can be found to the best of our knowledge. As a consequence in Section 1.1 we firstly describe the general theoretical background of deep learning, focusing on the neural network architecture used in our thesis work. Furthermore, we present most advanced studies in the similar fields in Section 1.2 and Section 1.3.

1.1. Deep Learning

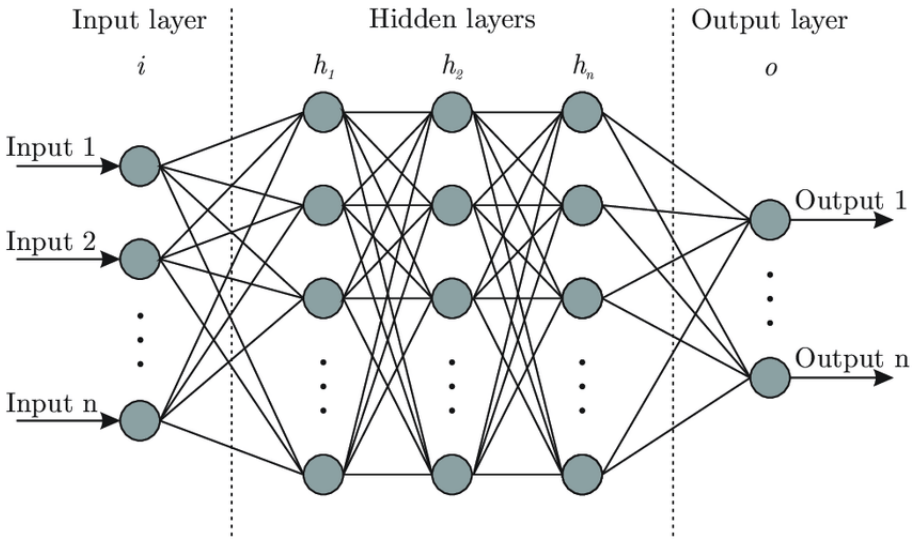


Figure 1.1: From [9] - General illustration of a deep neural network. It is formed by the input layer i, two or more hidden layers h_i and an output layer o.

Machine-learning technology powers many aspects of modern society: from web searches to content filtering on social networks to recommendations on e-commerce websites, and it is increasingly present in consumer products such as cameras and smartphones. Machine-learning systems are used to identify objects in images, transcribe speech into text, match

news items, posts or products with users' interests, and select relevant results of search. Increasingly, these applications make use of a class of techniques called deep learning.

Deep learning is a branch of machine learning which allows computational models to learn representations of data with multiple levels of abstraction [10]. Conventional machine-learning techniques were limited in their ability to process natural data in their raw form. They required careful engineering to design a feature extractor that transformed the raw data into a suitable internal representation or feature vector from which the learning subsystem, often a classifier, could detect or classify patterns in the input. Differently from shallow machine learning, deep learning uses a cascade of layers of nonlinear processing units for feature extraction and transformation. The complete set of these layers is called *deep neural network*. This process allows computers to learn from a hierarchical data representation where higher level features are derived from lower level features ones, where the word *feature* means an individual measurable property or characteristic of a phenomenon.

It has been shown that deep learning is very good at discovering intricate structures in high-dimensional data, in fact some fields of application include: computer vision, speech recognition, natural language processing, machine translation and medical image analysis.

In general a deep neural network (Fig 1.1) is composed by multiple *layers*. The first one is the *input layer*, which receives the data under study themself or with modified dimensions. In addition, the input layer is followed by two or more *hidden layers* which are the core of the network, since the relationship between the input and target data is learnt. This structure of layers is called the architecture. In order to better understand, a brief example is introduced: if we consider an image as the input of the neural network (specifically the image consists of a collection of pixels), then edges can be identified in the first hidden layer, corners and contours in the second and collection of them in a third layer. The *output layer* is the final layer in the neural network where desired predictions are obtained. Its length varies in relation to the task the network is developed for (prediction, classification, etc...).

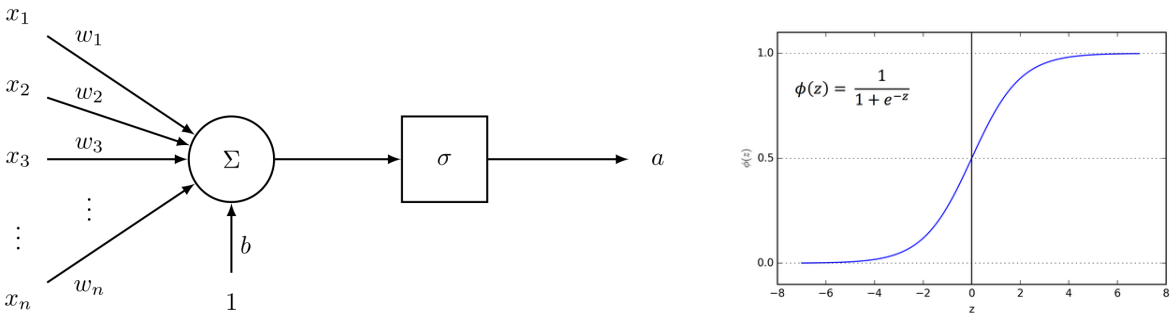
Each layer is composed by a fundamental unit called *neuron*. A neuron[1] (Fig 1.2a) has inputs $x_1, x_2, .., x_n$ that can take continuous values between 0 and 1. For each input the neuron has weights $w_1, w_2, ..., w_n$ and an overall bias b. The output is given by $\sigma(w \cdot x + b)$, where σ is called *sigmoid function*(Fig 1.2b) defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{1.1}$$

[1]with the term neuron we are referring to a sigmoid neuron [11]

The output of a sigmoid neuron with inputs $x_1, x_2, ..., x_n$, weights $w_1, w_2, ..., w_n$ and bias b is given by the following formula:

$$a = \frac{1}{1 + exp(-\sum_j w_j x_j + b)} \tag{1.2}$$



(a) Sigmoid neuron. (b) Sigmoid function.

Figure 1.2: Complete structure of a neuron (a): the inputs x_i are multiplied by the weights w_i and summed with a bias b. A sigmoid function σ (b) is applied to the result to obtain the output a.

Two main different ways of learning are defined in the literature: the supervised [12] and the unsupervised learning [13]. On the one hand, given the input and the target data, the aim of supervised learning consists in predicting the target from new data. Once the target has been labelled, an objective function is introduced so as to label the network output of new input data. This function is called *loss function* and it measures the error between the output and desired target. According to it, weights of the entire network are updated with a process called *backpropagation* [14]. This type of learning is usually adopted for regression problems [15].

On the other hand in *unsupervised learning* a large amount of unlabelled data are given to the network which needs to find a good representation of the input. This technique is often used in data clustering [16] and encoding [17].

Multiple deep neural network architectures are present in the literature (CNNs, RNNs, DBNs, etc...), each one aiming to resolve different tasks. In the next sections we present only the structures which will be treated in our thesis work.

1.1.1. Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are an example of artificial neural network which involves sequential inputs or time series data. These deep learning algorithms are commonly used for ordinal or temporal problems such as speech, language and music. They are distinguished by their "memory" as they take information from prior inputs to influence the current input and output. Figure 1.3 shows on the left the general structure of the model. Specifically, this architecture consists of an input sequence which feeds the neural network one element at a time. The presence of recurrent connections implies hidden units which maintain a state vector containing information about the history of all the past elements of the sequence. Moreover, this process allows the network to learn patterns in which the output is strongly related to prior elements within the sequence.

In order to better understand the internal process, a RNN can be unfolded into an acyclic NN as shown in the right side of Fig 1.3, in order to better understand the internal process. In the reference figure we denote x as the input and o as the output of the network, while t represents the number of steps. The network state at step t h_t can be calculated using the input of the same step x_t and the output of the hidden layer in the previous step h_{t-1} by:

$$h_t = f(U * x_t + W * h_{t-1}) \tag{1.3}$$

where f is normally a nonlinear activation function. h_t can also be viewed as the memory

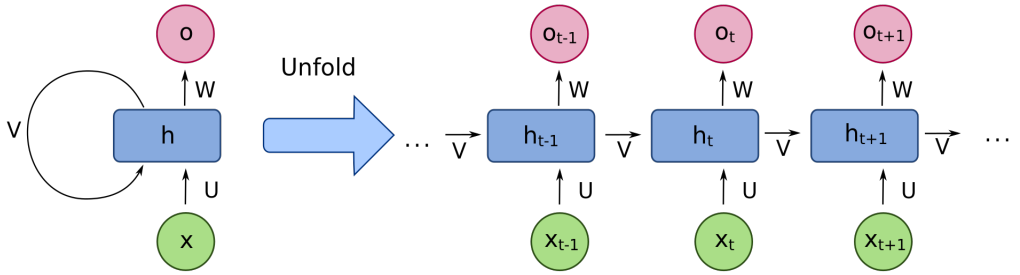


Figure 1.3: From [18] - Illustration of an unfolded recurrent neural network. The parameters V,U and W are shared between all the layers. t refers to the current step of the network. x_t and o_t are respectively the input and the output at step t. h_t in the hidden state of the network at step t.

unit in hidden layers, which can be used to store information from previous steps. The novelty of RNN relies on the share of parameters U,V and W among layers since it greatly reduces the requirement to learn the parameters of the network. The parameters

are updated through the computation of the loss function gradient that is backpropagated trough all the network.

RNN may lose the ability to learn connections among input data when their dependencies are very long. In fact, training this type of network has been shown to be problematic because at each time step the backpropagated gradients either grown or shrunk, so over many time steps they typically explode or vanish [19].

LSTM unit

To overcome the problem, long short-term memory (LSTM) network has been introduced [14]. In this architecture neurons are replaced by blocks (Fig 1.4) with a memory cell (long-term memory c_{t-1}), in addition to the hidden state of the classic RNN. In this way, this cell allows to remember long term dependencies.

Introducing respectively W_q and U_q[2] the matrices of input and recurrent connections' weights, it is possible to write the compact forms of the equations for the forward pass of the LSTM cell:

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{1.4a}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{1.4b}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{1.4c}$$

$$\tilde{c_t} = \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{1.4d}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c_t} \tag{1.4e}$$

$$h_t = o_t \circ \sigma_h(c_t) \tag{1.4f}$$

Where the initial values are $c_0 = 0$ and $h_0 = 0$. The operator $\circ$ denotes the element-wise product and the subscript t indexes the time step. In addition, σ_g is the sigmoid function while σ_c and σ_h are hyperbolic tangent function, while f_t refers to the forget gate which controls what information should be forgotten. As a consequence, the sigmoid function ranges between 0 and 1 and it sets which values in the cell state should be discarded (multiplied by 0), remembered (multiplied by 1), or partially remembered (multiplied by some value between 0 and 1). Furthermore, i_t is the input gate, whose role is to identify important elements which need to be added to the cell state c_t. o_t and $\tilde{c_t}$ are respectively the output gate's and the cell input activation functions.

[2] q can either be: the input gate i, the output gate o, the forget gate f or the memory cell c, depending on the activation being calculated.

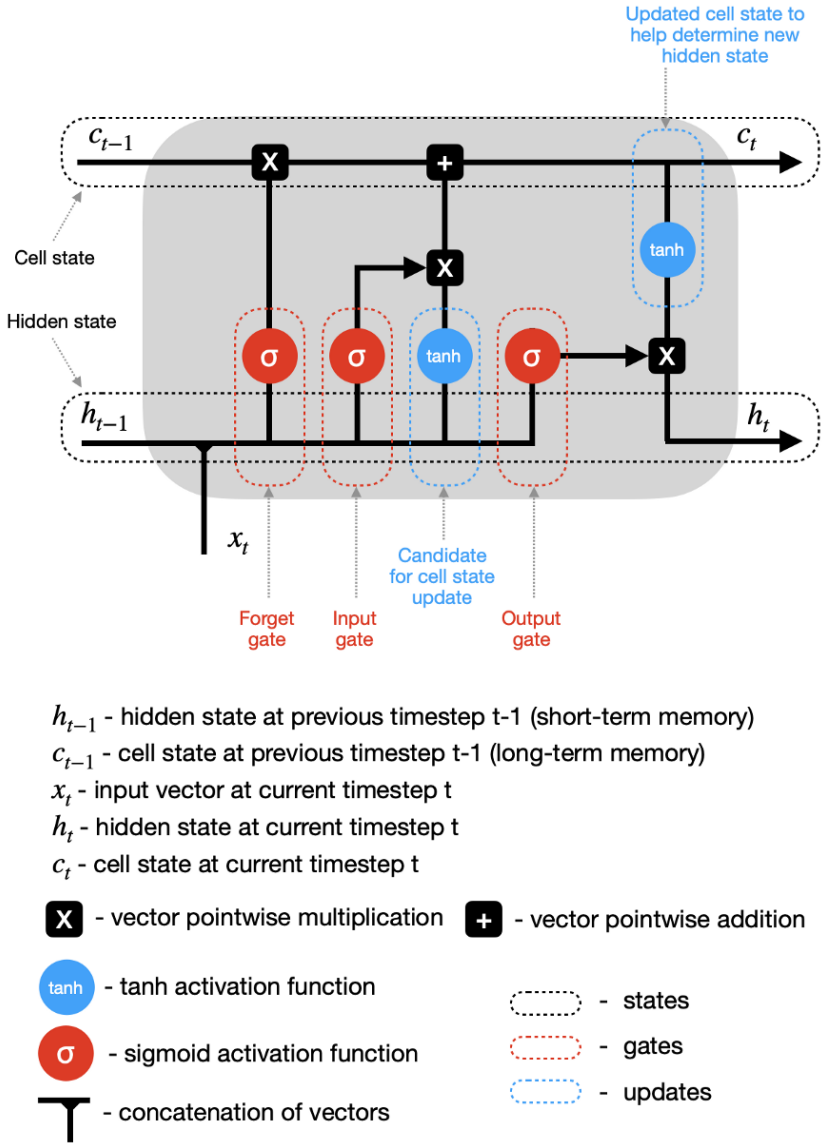


Figure 1.4: From [20] - Illustration of a LSTM recurrent unit.

1.1.2. Auto-encoders (AEs)

An auto-encoder is a specific type of a neural network, which is mainly designed to encode the input into a compressed and meaningful representation, and then it decodes the input back in such a way that the reconstructed input is as similar as possible to the original one [21].

Figure 1.5 provides an example of the auto-encoder model. Autoencoders are first introduced in [14], their purpose is to learn in an unsupervised manner an informative representation of the data, which can be used for various implications such as clustering. In particular, the main problem formally defined in [22], is to find functions $A : \mathbb{R}^n \rightarrow \mathbb{R}^p$

(encoder) and $B : \mathbb{R}^p \rightarrow \mathbb{R}^n$ (decoder) which satisfy:

$$\arg\min_{A,B} E[\Delta(x, B \circ A(x))] \tag{1.5}$$

where E is the expectation over the distribution of x, and Δ is the reconstruction loss function, which measures the distance between the output of the decoder and the input. The latter is usually set to be the ℓ_2-norm.

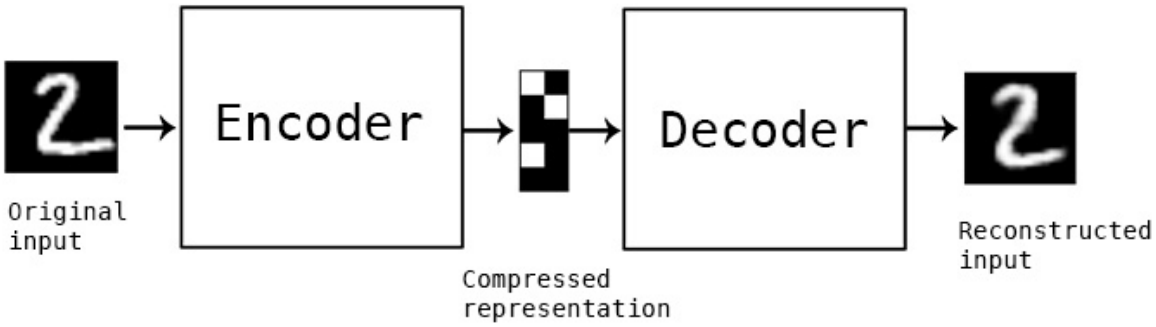


Figure 1.5: From [23] - An auto-encoder example. The input image is encoded to a compressed representation and then decoded.

In the most popular form of auto-encoders, A and B are neural networks [24], whose basic structure is formed by fully connected layers. AEs may be trained end-to-end or gradually layer by layer. In the latter case, they are "stacked" together, which leads to a deeper encoder [25]. Some examples of auto-encoders applications can be found in [26].

Convolutional auto-encoders (CAEs)

A convolutional autoencoder extends the basic structure of the simple autoencoder by changing the fully connected layers to convolution layers. CAEs differ from conventional AEs as their weights are shared among all locations in the input, preserving spatial locality. Their structure is described in [27]. For a mono-channel input x, the latent representation of the k-*th* feature map is given by:

$$h^k = \sigma(x * W^k + b^k) \tag{1.6}$$

where the bias b is broadcasted to the whole map, σ is an activation function (like ReLu or sigmoid) and $*$ denotes the 2D convolution. The reconstruction is obtained using:

$$y = \sigma\left(\sum_{k \in H} h^k * \tilde{W}^k + c\right) \tag{1.7}$$

where one bias c per input channel is present. Moreover H identifies the group of latent feature maps, while $\tilde{W}$ identifies the flip operation over both dimensions of the weights. The cost function to minimize is the mean squared error (MSE):

$$E(\theta) = \frac{1}{2n}\sum_{i=1}^{n}(x_i - y_i)^2 \tag{1.8}$$

Than the backpropagation algorithm is applied to compute the gradient of the error function with respect to parameters.

This type of auto-encoder is broadly used in image analysis field. Some implementation of CAEs are described in [28–30].

1.2. Analog audio effects modeling

One of the main research area which resembles the task of our thesis is the modeling of analog audio effects using deep neural networks. In fact, in multiple applications we are feeding an AI algorithm with clean guitar sound as input and the effected guitar sound as a target. In particular, the network models the non-linear behaviour between two sounds, both in the frequency (the spectral content) and time domain.

1.2.1. Non deep learning based models

Before the introduction of deep learning in audio effects modeling, different methodologies were proposed which are still a good alternative to neural networks. One of the main method of virtual analog modeling is based on WDFs (Wave Digital Filters). WDFs are a particular kind of digital filters based on physical modeling principles. They are modular, a peculiar characteristic which yields them especially well-suited for modeling block-based networks such as electrical circuits. Historically WDFs are able to manage just one non-linear electronic component in the system, for example an inductor or a capacitor. Fortunately, further studies overcome this limitation opening the possibility to model more complex circuits with a higher number of non-linearities, such as audio circuits. In particular, possible implementations vary from real-time simulation of tube amplifier

stages [1, 31], to distortion pedal modeling [2]. The primary advantage of the Wave Digital Filters approach is its modularity, namely the ability to construct a circuit model with each circuit component treated completely independently in the digital domain. Additionally, this modularity allows each circuit component to be discretized separately, which can improve the model behavior for certain classes of circuits. On the contrary, the main disadvantage of WDFs consist of their difficulty in handling circuits with complex topologies or multiple nonlinearities. While the recent addition of R-type adaptors to the Wave Digital formalism [7] begin to make these circuits tractable, the WDF models of these circuits type are significantly more computationally complex. A good comparison between this approach and a deep neural network based one is done in [8].

A different approach to digitally recreating an analog device is found in [3]. The architecture is composed by a block-oriented Wiener-Hammerstein model. The basic idea is to have a parametric model which is flexible enough to be adaptable to many distortion effects, but still simple enough to be computationally efficient. The architecture consists of lineartime-invariant (LTI) blocks and a nonlinear block. Blocks are ordered in series where the nonlinear block is lined by two LTI blocks. The LTI blocks are filters, and the nonlinear block is a mapping function, mapping the level of the input signal to an output level, which simulates the nonlinear behavior of the distortion effect. Block-oriented Wiener-Hammerstein models are successfully used in commercial products due to their flexibility and expandability. Although this approach is more computationally demanding than other methods, it also suffers from local minimum convergence instead of global minimum convergence.

1.2.2. Deep learning based models

Trying to overcome the difficulty in modeling multiple nonlinearities, a deep learning based method is introduced by Zhang et al.[32] work. They proposed a long short-term memory (LSTM) model with many layers but a small hidden size in each layer, although the authors reported clearly audible differences between the resulting model and the target device. An improvement on perceptual results is brought by Wright [4]. The paper presents a comparison between two deep learning based methods applied to black-box modeling of two valve amplifiers. The first is a feed forward variation of the WaveNet convolutional autoregressive model [33], while the second is one of the most widespread method in this research area, since it is based on RNN (Recurrent neural network). In particular, this method exploits the LSTM unit cell described in Section 1.1.1, which allows the neural network architecture to have better output predictions. Both methods show an interesting result, however the RNN model performs better than the Wavenet

from a computational load and ESR point of view, make it more suitable for real-time implementations on a consumer-grade desktop computer. Moreover, the paper shows that only three minutes of audio data are necessary for both models' training part. Finally, the authors conduct a perceptual evaluation of the two proposed architecture results based on MUSHRA method [34]. The RNN model shows a better score than WaveNet, but in general they both achieve great results in the listening test.

A different approach for audio effects modeling based on TCNs (Temporal Convolutional Networks), is presented by Steinmetz and Reiss [5]. They do a comparison between a 32 hidden size LSTM (LSTM-32) and TCN architecture applied to dynamic range compressors modeling [35], which turns out challenging due to their time-dependant nonlinearities. Figure 1.6 shows the structure of the TCN model, which consists of residual blocks composed by 1-dimensional convolutions with increasing dilation factors, followed by batch normalization, a conditional affine transformation (FiLM) [36], and a PReLU [37] nonlinearity.

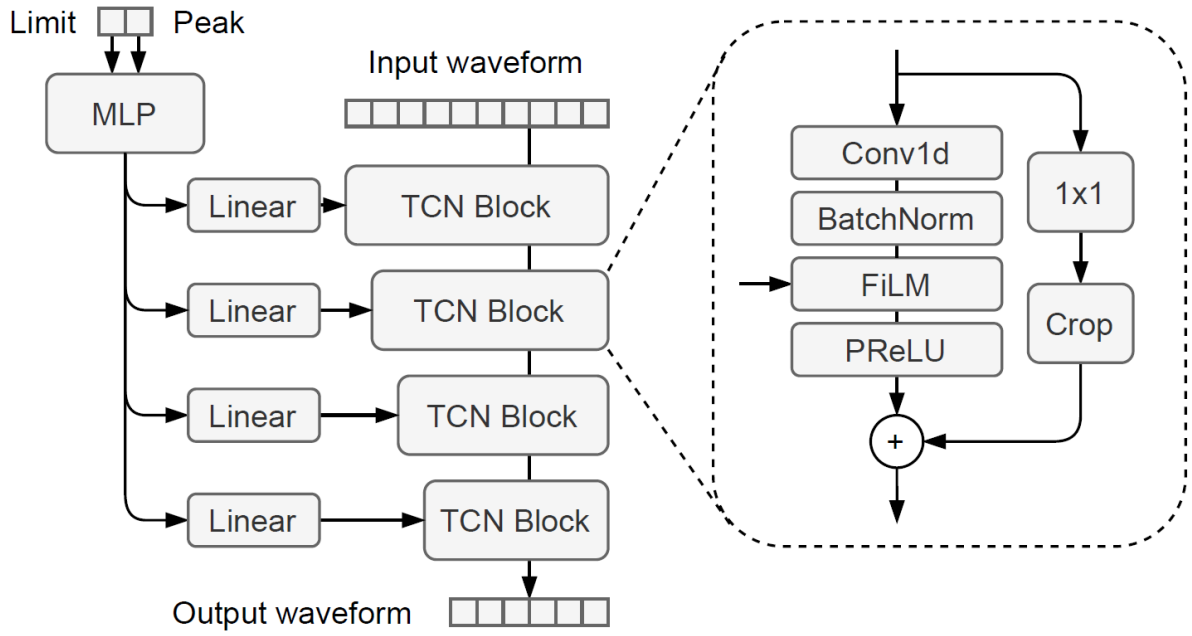


Figure 1.6: From [35] - General TCN architecture featuring a global conditioning module (3 layer MLP) that generates embeddings for each FiLM operation in the TCN processing pipeline as a function of the limit and peak reduction controls. The contents of the TCN block are shown in the dashed block on the right.

The study shows interesting results both in dataset training length and computational efficiency. In fact, it demonstrates that optimal outcome can be achieved also with a training subset of 11 minutes of the total 20 hours of training data audio. Moreover, the

TCN model succeeds in a better performance in terms of real-time implementation than the LSTM-32 one which is not able to reach real-time operation. On the other hand, the real-time factor for the TCN model is proportional to the frame size, moreover larger frame sizes turn out greater real-time factors as a result of parallelization, both on CPU and GPU.

A final comparison based on objective results and listening test is discussed. It is shown that the LSTM-32 model performs better that TCN in terms of the objective metric parameters such as LUFS error and STFT metric. The first metric is calculated as:

$$\ell_{LUFS}(\hat{y}, y) = |G(\hat{y}) - G(y)|$$

where $\hat{y}$ and y are respectively the model prediction output and the target output. Moreover $G(\cdot)$ refers to the ITU-R BS.1770 loudness algorithm [38]. First introduced by Arik et al [39], STFT metric (1.11) is composed of two terms, the spectral convergence ℓ_{SC} (1.9), and spectral log-magnitude ℓ_{SM}, where $\|\cdot\|_F$ is the Frobenius norm, $\|\cdot\|_1$ is the L1 norm, and N is the number of STFT frames.

$$\ell_{SC}(\hat{y}, y) = \frac{\| \, |STFT(y)| - |STFT(\hat{y})| \, \|_F}{\| \, |STFT(y)| \, \|_F} \tag{1.9}$$

$$\ell_{SM}(\hat{y}, y) = \frac{1}{N} \| \, log(|STFT(y)|) - log(|STFT(\hat{y})|) \, \|_1 \tag{1.10}$$

$$\ell_{STFT}(\hat{y}, y) = \ell_{SC}(\hat{y}, y) + \ell_{SM}(\hat{y}, y) \tag{1.11}$$

Although in the listening test performed using the MUSHRA method, both models performed in a good manner, appearing slightly below the reference sound. Moreover, it seems that some listeners struggled to differentiate between the reference and the tested models.

1.3. Distortion removal in music

Another research area close to our thesis work is distortion removal, which was introduced by Johannes Imort and Giorgio Fabbro in [6]. Their aim consist in removing distortion and clipping applied to guitar tracks for music production while presenting a comparative investigation of different deep neural network (DNN) architectures applied to the task.

The discussed models are CRAFx [40],Wave-U-Net [41], Open-Unmix [42] and Demucs [43]. In order to assess the quality of all models, four different metrics are used: the scale-

invariant source-to-distortion ratio (SI-SDR), the perceptual evaluation of audio quality (PEAQ), the Rnonlin metric, and the Frèchet audio distance (FAD). The first one is calculated as:

$$\text{SI-SDR} = 10\log_{10}\left(\frac{||\frac{\hat{s}^T s}{||s||^2}s||^2}{||\frac{\hat{s}^T s}{||s||^2}s - \hat{s}||^2}\right) \tag{1.12}$$

where s denote the clean signal and $\hat{s}$ is its reconstruction. Since the SI-SDR does not take into account the human perception, paper's authors introduced the other three metrics to evaluate the models. The PEAQ measures the amount of degradation between two generic audio signals using a fast Fourier transform (FFT) based on peripheral ear model for the calculus of MOVs (Model Output Values), and ODG (Overall Difference Grade) for the final score. Moreover, the R_{nonlin} (Perceived Quality of Non-linearly Distorted Music and Speech Signals) was developed specifically for non-linear distortions. This metric is based on multiple filters which tries to model the human ones (such as the 1-ERB_N-wide gammatone filters), and on the short-term cross-correlation which measures the difference between the reference and the test signal. Furthermore, R_{nonlin} is defined between 0 (high distortion) and 1 (no distortion). The last used metric is the FAD (Frèchet Audio Distance) which is based on the calculus of the Frèchet distance between two multivariate Gaussian models of the reference and test signal. In order to obtain the embedding statistics for the calculus of the Gaussian, VGGish model [44] is employed. For each sample it uses mel spectrograms as its input.

Once the main four metrics have been briefly described, we can dare to say that the Demucs model is the best in comparison to all the other models independently from the used metric. Demucs architecture was initially designed to be an end-to-end model for music source separation in time domain. It comprises a convolutional encoder, a BLSTM structure and a convolutional decoder and Figure 1.7 shows its complete structure. Each of the encoder's L levels consists of a 1-D convolution, while the initial number of output channels doubles with each level. Each level uses ReLU activations and passes its output to an additional 1-D convolution with kernel size and stride equal to 1 which doubles the output channels. Then, a gated linear unit halves the output channels. After the last level, the output is fed to a 2-layer BLSTM structure in order to capture long-term dependencies. The decoder is defined as the inverse of the encoder and it reduces the output channels with each layer. Skip-connections allow a direct access to the input signal's phase while they facilitate gradient exchange between each encoder and decoder. Papers' authors reduced the learnable parameters from 66 million to 1 million by decreasing the number of layers.

Johannes Imort and Giorgio Fabbro's work introduces Demucs model to a new application

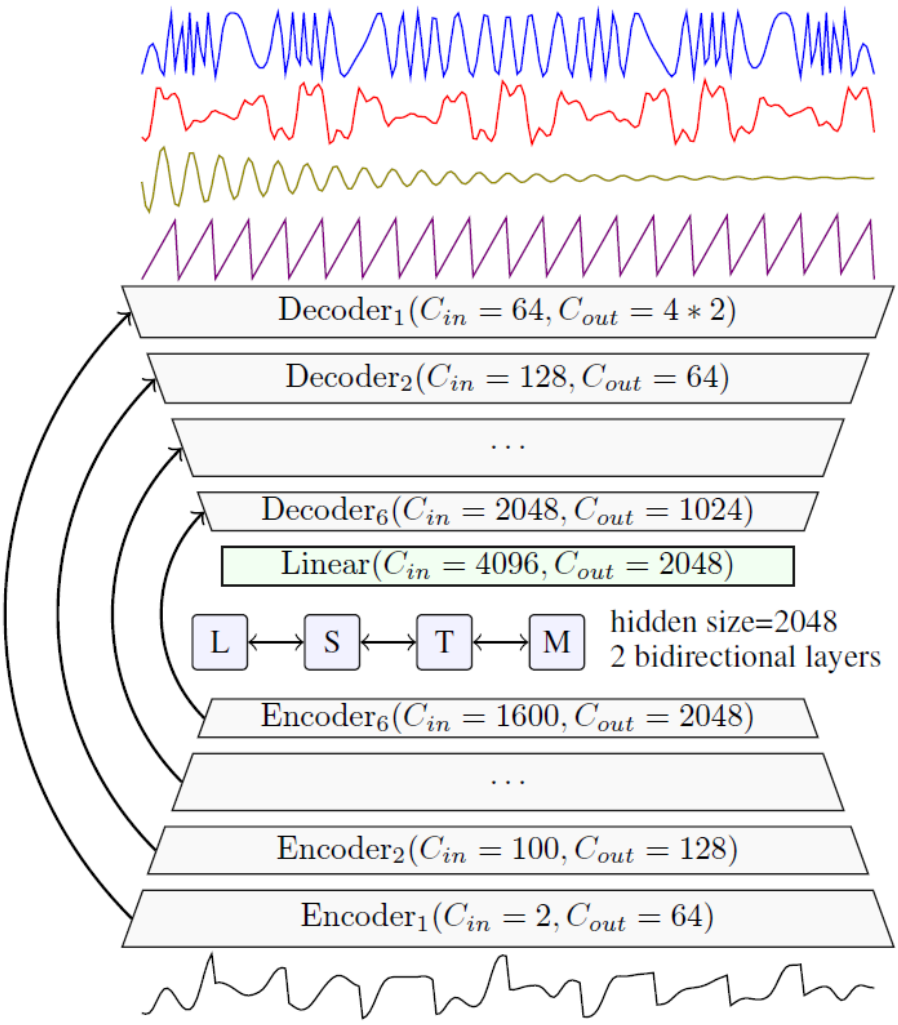


Figure 1.7: From [43] - Demucs architecture with the mixture waveform as input and the four sources estimates as output. Arrows represents U-Net connections.

field, opening new possibilities in music and audio modeling.

2 | Model and methods

In this chapter we present the model and methods for black-box modeling of the nonlinear relation between a piezoelectric pick-up and a cardioid microphone recordings. In Section 2.1, we describe the model of the neural network. We describe the overall data acquisition process and the description of the obtained signals in Section 2.2. Furthermore, an analytical assessment of the model's loss function is explained in Section 2.3. Finally, the pre-processing and training are presented in Section 2.4.

2.1. RNN Model

In order to accomplish our task, a deep learning based approach is adopted. First implemented in [4], the chosen neural network model is the RNN (Recurrent Neural Network). This network receives the piezo-electric pickup recording as input and the cardiod microphone ones as target.

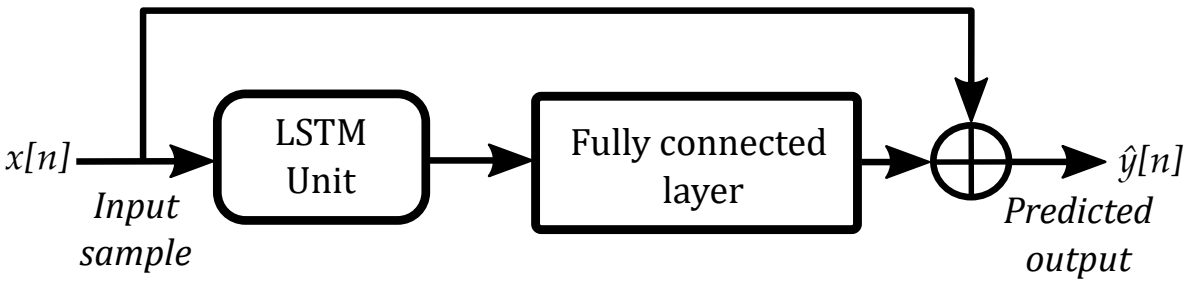


Figure 2.1: RNN model. The input x_n goes first to the LSTM unit and then into a fully connected layer. The output of the latter is summed with the initial x_n to obtain the predicted output $\hat{y}_n$

Fig 2.1 shows the entire architecture. The RNN network is composed by an LSTM (Long-Short Term Memory) unit, followed by a Fully Connected layer. At each time step n a single input sample $x[n]$ is fed into the LSTM unit. The output of the latter goes into

the Fully Connected layer to produce a single output which is summed with the initial input $x[n]$ to obtain the final predicted output $\hat{y}[n]$. Doing so, the network just learns to predict the difference between input and output samples.

The state of the LSTM unit is made of two vectors: the *hidden state* h and the *cell state* c. For each time step n, $x[n]$, $h[n-1]$ and $c[n-1]$ are used to calculate the LSTM's output $h[n]$ and $c[n]$. Equations (1.4) show how each value of the LSTM unit is computed. The hidden layer $h[n]$ is input to the fully connected layer. As the fully connected layer is not followed by an activation function, the output of the fully connected layer is simply an affine transformation of the LSTM hidden state. This output is summed with the input sample, $x[n]$, to produce the RNN's predicted output:

$$\hat{y}[n] = W_{fc}h[n] + b_{fc} + x[n] \tag{2.1}$$

Where W_{fc} and b_{fc} are respectively the fully connected layer's weights matrix and bias vector. The fully connected layer consists of a single neuron that will output a single value at each time step n.

The size of both the hidden and cell states is equal to the LSTM's hyperparameter *hidden size*. Increasing the hidden size generally results in the model being more accurate. However it increases the number of learnable parameters in the network, as well as the processing power required to run it. The PyTorch machine learning library [45] was used to implement the whole RNN model.

2.2. Data acquisition

The diagram of the data acquisition process is shown in Figure 2.2. An acoustic guitar is simultaneously recorded from its piezo-electric pickup and a professional microphone placed in front of it. In order to do that, an audio interface is used. A jack cable connects the guitar pickup to the first channel, while an XLR cable carries the microphone signal to the second channel. The audio interface is connected to a laptop using an USB cable. To record the multi-track we relied on Ableton Live®. This software allow us to record and edit multiple audio tracks at the same time. Finally, all the recorded tracks (pickup and microphone version) have been exported in mono audio files. The instrument has been recorded in a small room with no particular acoustic treatments.

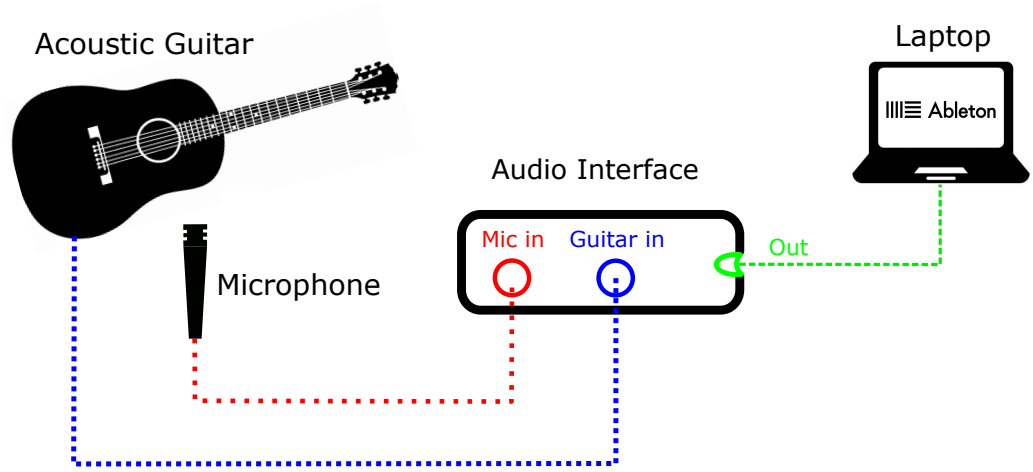


Figure 2.2: Data acquisition process. The signal is recorded simultaneously from the acoustic guitar pick-up and a microphone using an audio interface. Its output is connected via USB to a laptop. Ableton Live® was used to record and export the audio.

2.2.1. Equipment

For the recording process we use the *ovation®-celebrity-cs24-4-g*[46] acoustic guitar. Which has flamed maple top, the body is made in Lyrachord®, the neck is made in NATO wood and the fretboard and bridge are in Ovangkol. Furthermore, we use the low-impedance cardioid dynamic microphone *Shure® SM57*[47]. Which is typically used in contexts like recording studio or live music concert. Finally, we adopt *Roland® duo-capture-ex* as audio interface[48]. It has two input channels and two preamplifiers Roland VS. The signal from the guitar to the audio interface is carried by a *Fender® jack cable California series*, while for the microphone one a Cordial® XRL is used.

2.2.2. Acquisition parameters and signals description

The guitar and microphone signals are acquired at 44.1kHz. We obtain two mono audio tracks for each recording. All the audio lengths are between 1 and 2 minutes.

Figure 2.3 shows the time domain representation of the two obtained signals. The curves exhibit a different trend. The blue one referring to the pick-up acquired signal is richer in high frequencies components than the orange one referring to the SM57. This characteristic is reflected in time domain by the abrupt changes of the the blue curve. We can observe it also in the spectrograms of the two signals (Fig.2.4).

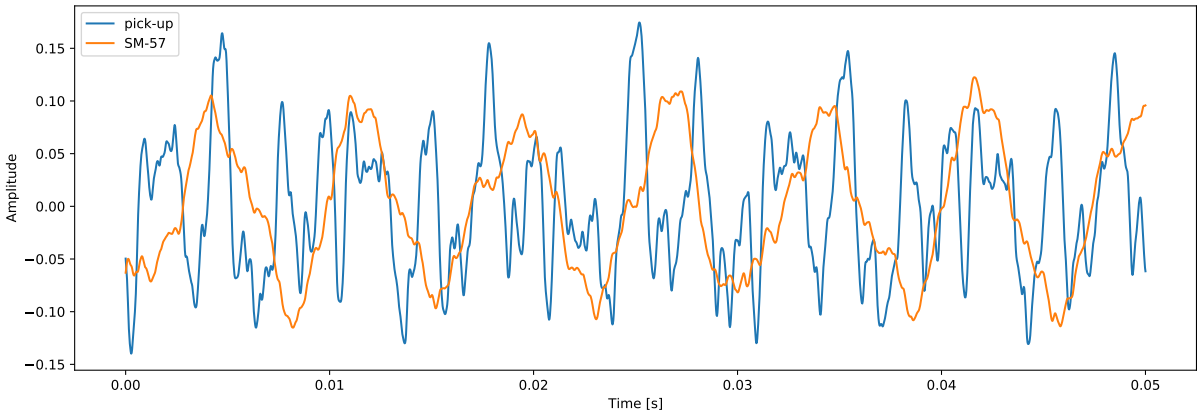
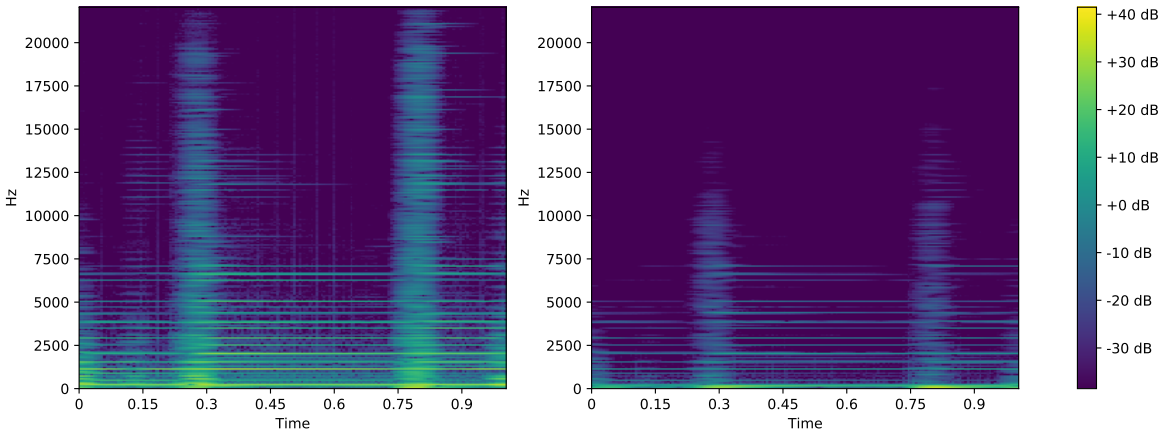
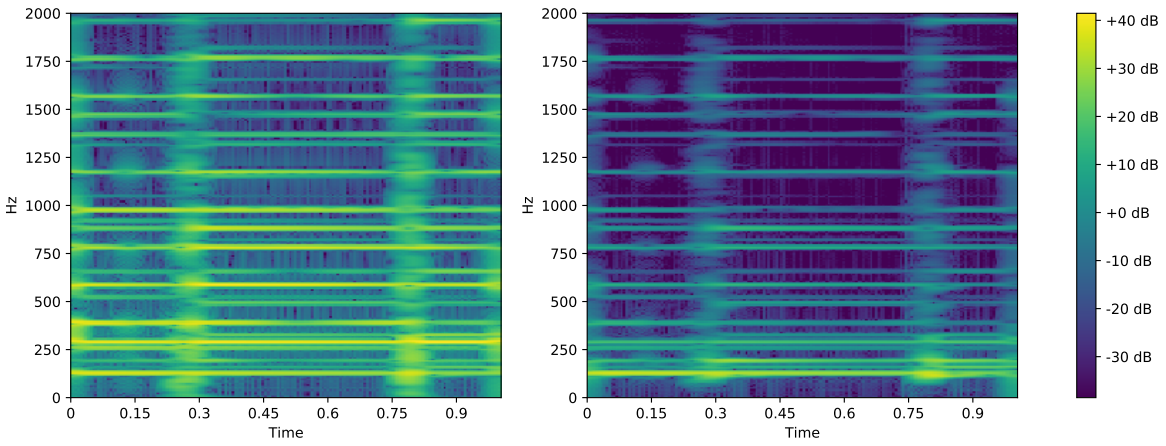


Figure 2.3: Time domain representation of the recorded signals. The image shows 0.05 second of audio. The blue curve refers to the signal recorded by the acoustic guitar's pick-up, while the orange indicates the one obtained using a SM57 microphone.



(a)



(b)

Figure 2.4: Spectrograms of the pick-up and microphone signals. The image refers to 1 second of audio. For both images we have the guitar pick-up signal on the left and microphone signal on the right. (a) refers to the entire audio bandwidth, (b) refers to a low-mids frequency band (0 Hz - 2 kHz).

As a matter of fact, we can see that the energy of the microphone one is concentrated more on frequencies lower than 1 kHz. The pick-up spectrograms instead present a lot of energy also in the mid-high frequencies. These differences can be heard clearly in the two recordings. The microphone audio is characterized by a darker tone with respect to the piezo pick-up one.

2.2.3. Dataset definition

One of the key aspect for a proper learning of the network is the definition of a good dataset [49, 50]. We aim to collect as many styles of acoustic guitar playing as possible. Some example are: strumming, single note, arpeggio and open chords playing. For each one of these styles at least two audios are recorded. The single duration of each piece of music is around 1-2 minutes, for a total duration of 29 minutes and 5 seconds. The Python *Pandas* [51] and *csv* libraries are used for the management of the dataset.

	name	ext	n_segm	length
8	A_blues	input	0	1367100
9	A_blues	input	1	1367100
10	A_blues	target	0	1367100
11	A_blues	target	1	1367100
12	chords	input	0	2469600
13	chords	input	1	2469600
14	chords	target	0	2469600
15	chords	target	1	2469600

Figure 2.5: Example of some elements of the dataframe. Starting from the left, for each audio we have: the number of the audio, the name, the extension, the segment number and the length in samples.

2.3. Loss function

A key element for a proper learning of the network is the loss function. In this section we give the definition of the loss function used in [4]. We analyze it in relation to our problem and we present the final function we use. Furthermore, we describe the method adopted to assess whether it is suitable for our task.

2.3.1. Definition

For a signal of length N the loss function ε is the result of the sum of two contributions:

$$\varepsilon = \varepsilon_{ESR} + \varepsilon_{DC} \tag{2.2}$$

The first component is the error to signal ratio (ESR) with respect to the training data, calculated as:

$$\varepsilon_{ESR} = \frac{\sum_{n=0}^{N-1} |y_p[n] - \hat{y}_p[n]|^2}{\sum_{n=0}^{N-1} |y_p[n]|^2} \tag{2.3}$$

Where $y_p[n]$ and $\hat{y}_p[n]$ are respectively the target signal and the output of the neural network at sample n. For both signals a low-passed A-Weighting filter (Fig. 2.6) has been applied. Its purpose is to emphasise the frequencies in the loss function, based on their perceived loudness. The denominator in the ESR normalises the loss with regards to the target signal energy. As a matter of fact it prevents the loss function to be dominated by the segments of signal with higher energy.

The second additional member ε_{DC} of the equation (2.2) represents the difference in DC offset between the target and neural network output:

$$\varepsilon_{DC} = \frac{|\frac{1}{N}\sum_{n=0}^{N-1}(y[n] - \hat{y}[n])|^2}{\frac{1}{N}\sum_{n=0}^{N-1}|y[n]|^2} \tag{2.4}$$

The target $y[n]$ and the network's output $\hat{y}[n]$ have not been filtered.

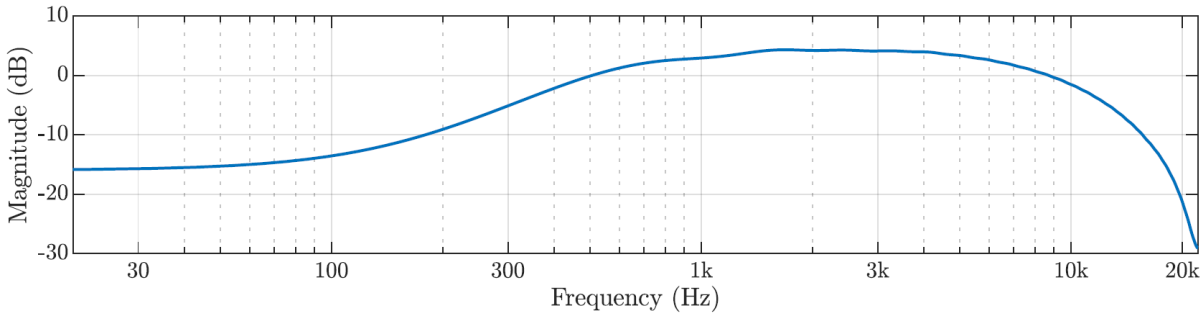


Figure 2.6: From [4] - Frequency response of the low-passed A-weighting pre-emphasis filter.

2.3.2. Evaluation

Firstly, the signal is divided into segments and for each one we calculate the ϵ_{ESR} (2.3) and ϵ_{DC} (2.4) components of the loss function. Figure 2.7 shows their values on the y

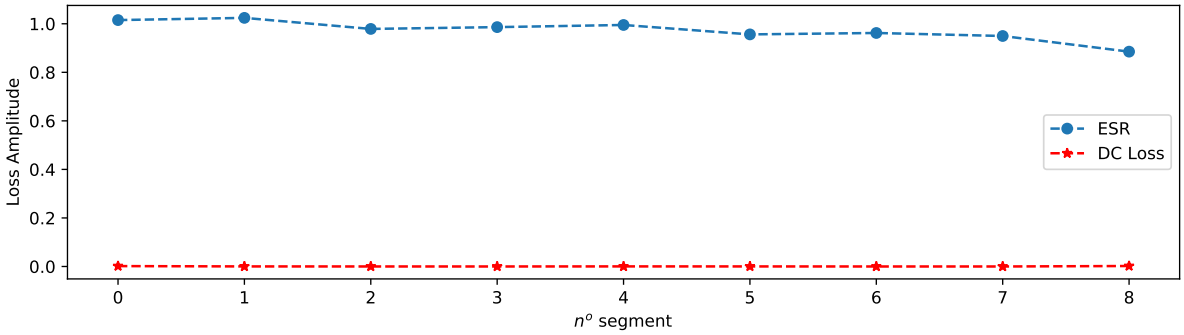


Figure 2.7: Comparison between loss results. Each dot represent a value of the loss function calculated in a specific segment. The red and blue curve refers respectively to the DC and ESR components of the loss.

axis for 9 consecutive segments. Each chunk corresponds to $0.1s$ of audio. We tried to select a group of segments in which a guitar's chord is played. The red curve refers to DC component. Moreover, we can notice that it is always close to 0, on average in the order of 10^{-4}. We can conclude that the ESR factor (the blue one) carries the information we are interested in, therefore we decide to neglect the DC component and use only the ESR for the loss evaluation. The final function used in the network model is:

$$\varepsilon = \varepsilon_{ESR} \tag{2.5}$$

Looking now at the spectrograms of input and target signals, Figure 2.4 shows that the input has more energy in the middle-high frequencies with respect to the target one. In order to get more similar signals, a smoothing algorithm is applied to the pick-up audio in the pre-processing stage. The smoothing consists of a convolution between the signal and a Blackman window. The result of this process is shown in Figure 2.8. In order to see if the smoothing process improves the loss values, we calculate the ESR component for the 10 segments previously defined. As a result, we can observe in Figure 2.10, that the relative orange curve has lower values than the blue one referring to the raw input signal.

Another aim is to understand if the network is able to learn using the ESR loss function. As a consequence we build a fake signal which tries to emulate what the network should

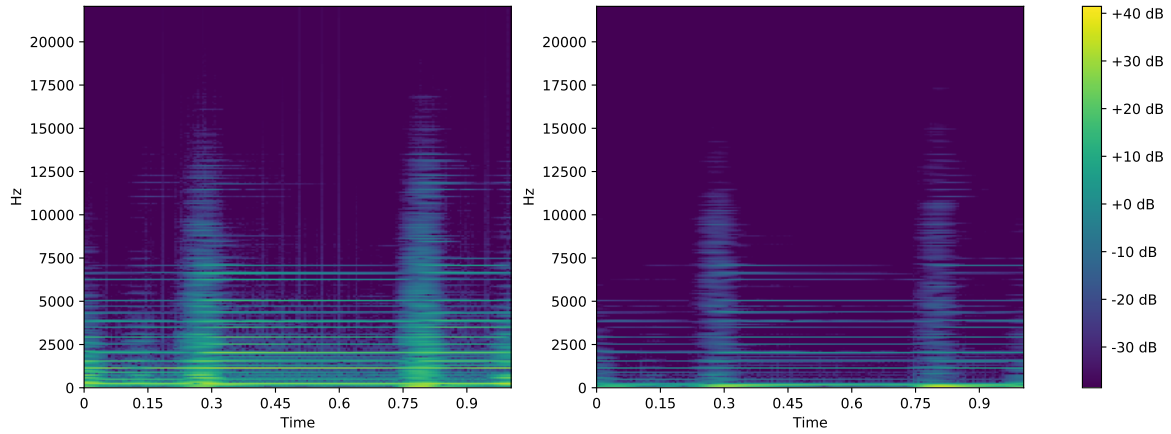


Figure 2.8: Spectrogram of the input signal (on the left) feeded to the network and the target signal (on the right). The input signal has been smoothed to be more similar to the target one. The difference between the recorded signal and the smoothed one could be seen doing the comparison with Fig 2.4a.

do with the original signal. The *fake signal* is obtained as a sum of the smoothed input signal s_{smooth} with 4 sinusoids at different frequencies (fig 2.9): 127-200-350-583 Hz:

$$s = s_{smooth} + \sum_{i=1}^{4} s_i \tag{2.6}$$

We choose the frequencies starting from the target and input signal spectrograms by looking at which frequencies the input lacks/exceeds in terms of energy. The single sinusoid is obtained as:

$$s_i = \sin(2\pi \cdot f_i \cdot t) * rms_n \tag{2.7}$$

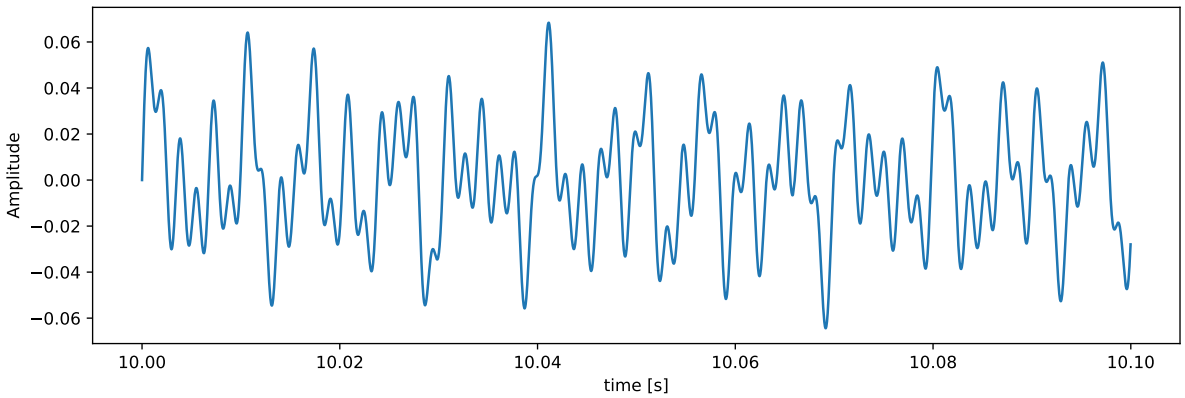


Figure 2.9: Time domain representation of the sinusoidal signal. It is obtained as a sum of multiple sinusoids at low frequencies.

Where f_i is one of the selected frequencies. t correspond to the vectors of all time steps. rms_n is the root mean square values for each segment of audio.

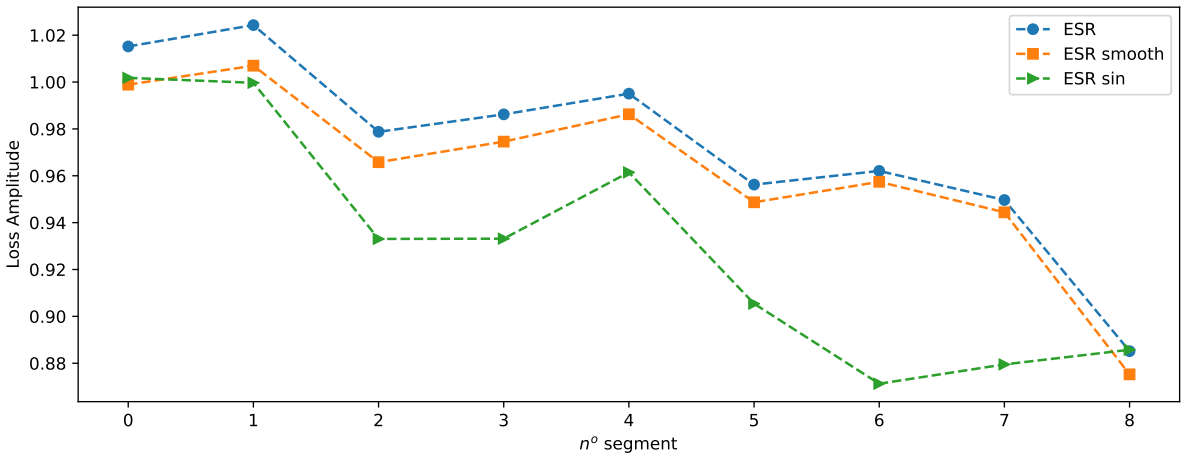


Figure 2.10: ESR loss values for different type of input signals. Each dot represent a value of the loss function calculated in a specific segment. The blue one is the ESR of the raw input signal. The orange one is obtained using a smoothed version of the input signal. The green one uses the latter, at which a sinusoidal signal (Fig 2.9) is summed.

Fig.2.10 shows that the *fake signal* (green curve) has lower values of the ESR loss than the smoothed one (curve in orange). This result suggests that the loss function we choose could perform well on the task we aim to.

2.4. Pre-processing and Training

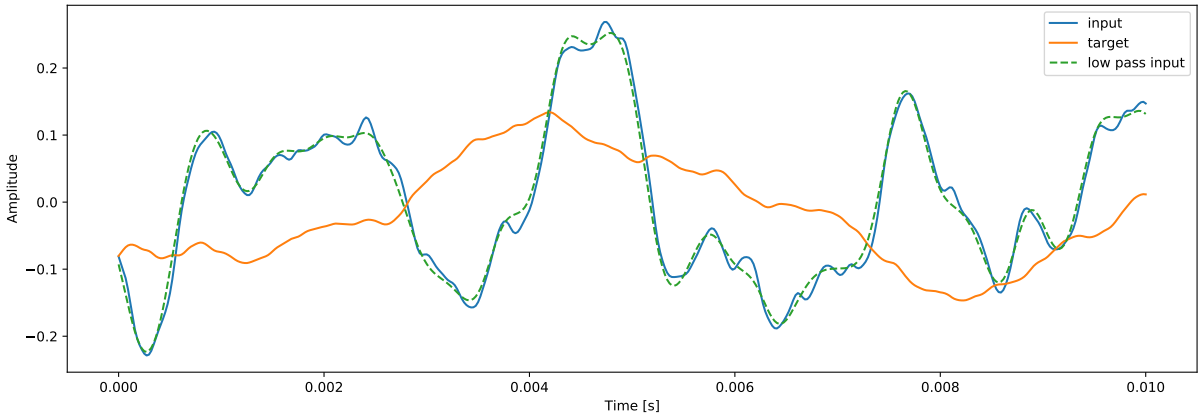


Figure 2.11: Time domain representation of the input, target and low pass input. The last two are the signals feed into the network.

Before entering in the network loop, the training data are pre-processed. Since the majority of the energy of the target is concentrated around lower frequencies, a low pass filter (Fig 2.12) is applied to the input training signal. We use Butterworth digital low pass filter [52]. Figure 2.11 shows the input and target signals fed into the network.

In order to have balance between training and validation data, each audio is split into 2 parts. Doing so we increase the number and variability in the selection process. 70% of these segments are assigned to the training data and 20% to the validation. Once the splitting is defined, the training and validation arrays are obtained concatenating the respective audio segments. Furthermore, for the test data we used an audio which is a mixture of guitar playing styles.

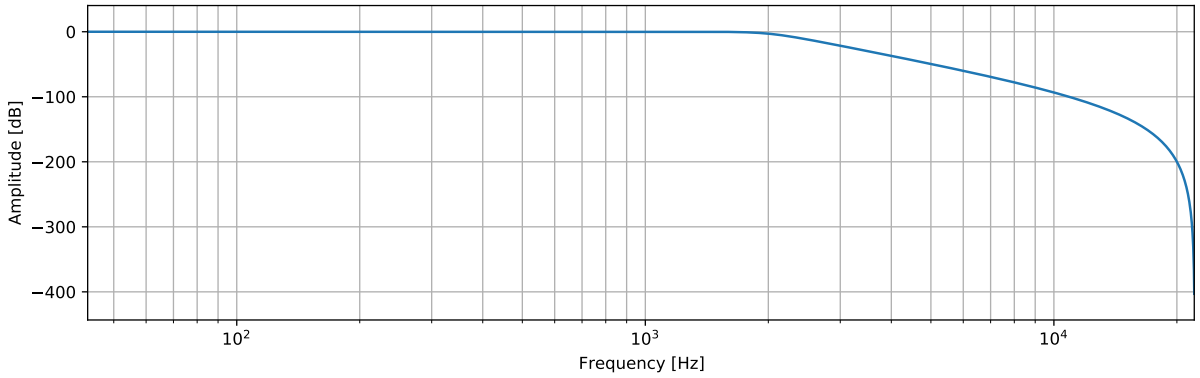


Figure 2.12: Low pass digital filter with cut-off frequency 2kHz.

In order to be processed by the neural network, the dimensions of the three data arrays are modified and they are converted into tensors. A *tensor* is a generic n-dimensional array to be used for arbitrary numeric computation, that can be run either on CPU or GPU. Using this element, we can take advantage of the GPU computing power for accelerate the training process. The training array is split into overlapping batches of $segment\_length = 7$ second. Furthermore, we use an *overlap* parameter to control the percentage of overlapping between two consecutive segments. Doing so, we obtain a tensor with dimension $[segment\_length \times n\_batches \times n\_channels]$, where $n\_channels$ is the number of channels of the recording (mono or stereo).

The model is trained using Adam optimizer [53]. The RNN is trained for 1000 epochs. For each epoch, the training data are shuffled in group of 7 consecutive batches. The validation loss is calculated every three epochs. If the validation loss does not improve within 200 epochs, the training stops. The starting learning rate value $LR_i = 0.01$ is decreased dynamically by a multiplicative factor $k = 0.7$ every 3 epochs the validation loss is not improving. Furthermore, to avoid local minima it is also reset to $0.8LR_i$ at

epoch 500 and $0.1LR_i$ at epoch 700. Figure 2.13 shows an example of the learning rate trend for a complete training cycle.

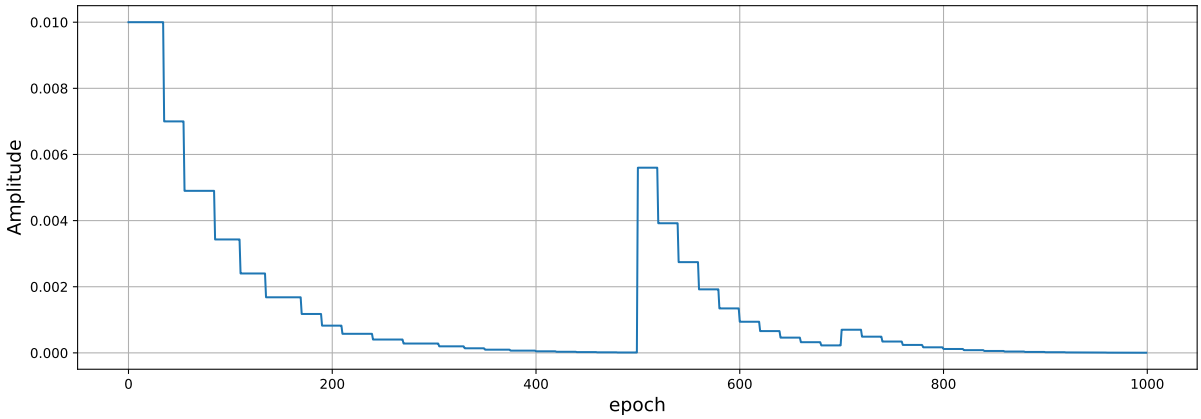


Figure 2.13: Example of the learning rate curve for a complete training cycle.

Once training is completed, the test loss is calculated using the model parameters from the epoch in which the lowest validation loss was achieved. For the training process, a machine with the following characteristics has been used: Intel® Core i7-6700K, 8 Cores with 4 GHz processor, 31 Gb RAM and one NVIDIA® GTX 1080 Ti graphic card [54].

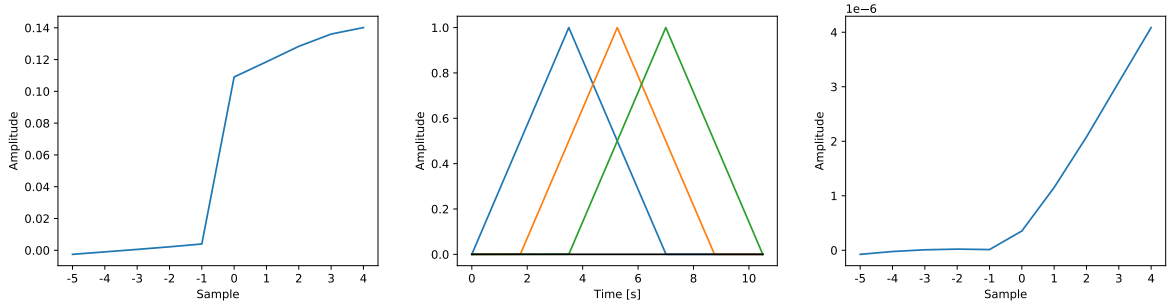
3 | Results

In this chapter we are going to present and analyse the network results. In the first section we describe the steps made in order to obtain the model's output. Moreover, we would like to describe the clicking noise generated by the network and our proposed solution. Furthermore, we compare model outputs with the target signal, both in time and frequency domain, so as to have a general view of the model performance. Finally, a comparison based on ESR (Error to Signal Ratio) between different guitar playing style is presented. In the second section we are going to analyse multiple network's models, highlighting their scores in terms of ESR. In addition, a perceptual analysis conducted by the Author is presented, which underlines the fact that a low ESR value does not necessarily correspond to a better perceptual audio result.

3.1. Audio results

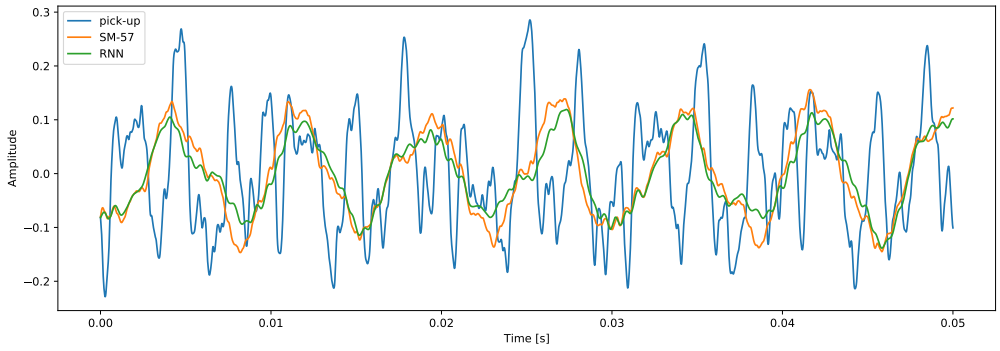
In this paragraph we present the best performing model in terms of obtained ESR value in the test phase. However, as we will discuss in Section 3.2, this good performance does not necessarily mean a better perceptual result.

Recalling Section 2.1 we are going to describe the model performance, in particular the chosen one has an hidden size of 96 . Once the model is trained, as already presented in Section 2.4, the output data are obtained feeding the RNN network with raw input data. The input audio is segmented in overlapping chunks of 7 seconds each with an overlapping factor of 0.75. Each segment is processed by the network and then the entire output is reconstructed with an overlap and add approach. However the network processing has a significant problem: discontinuities between two consecutive segments cause a click noise in the final audio. The reason of this issue regards the time step t as $t = (1 - overlap) * segment\_len$ which has a jump. Figure 3.1a shows the difference in terms of amplitude between two adjacent chunks. In order to solve this problem, we applied triangular windowing where each segment is multiplied by a triangular window (Figure 3.1b) during the overlap and add process. Each triangular window has a unitary amplitude and it share the same length and overlapping factor of the audio chunks.

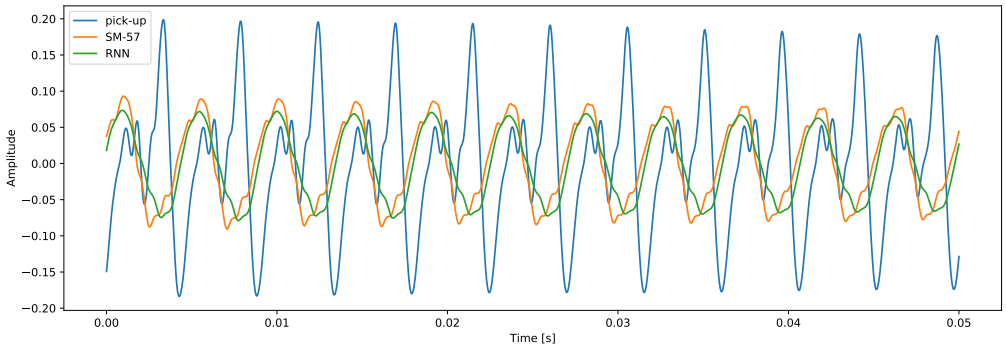


(a) Signal discontinuity. (b) Triangular windowing. (c) Windowing result.

Figure 3.1: (a): Discontinuity between two consecutive audio segments which creates the click noise in the final output audio. (b): chunk of triangular windows array. (c): we can see how the triangular windowing process removes the discontinuity between adjacent segments.



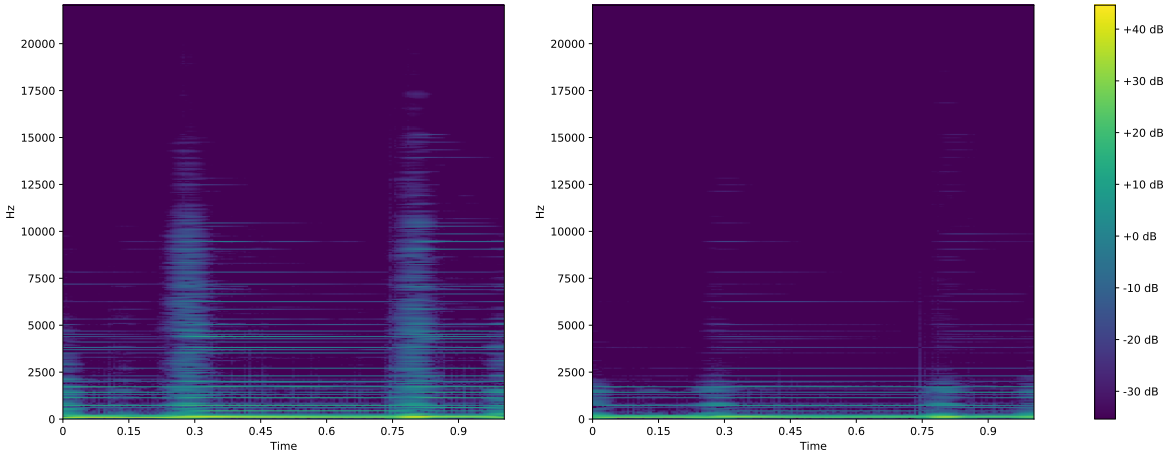
(a)



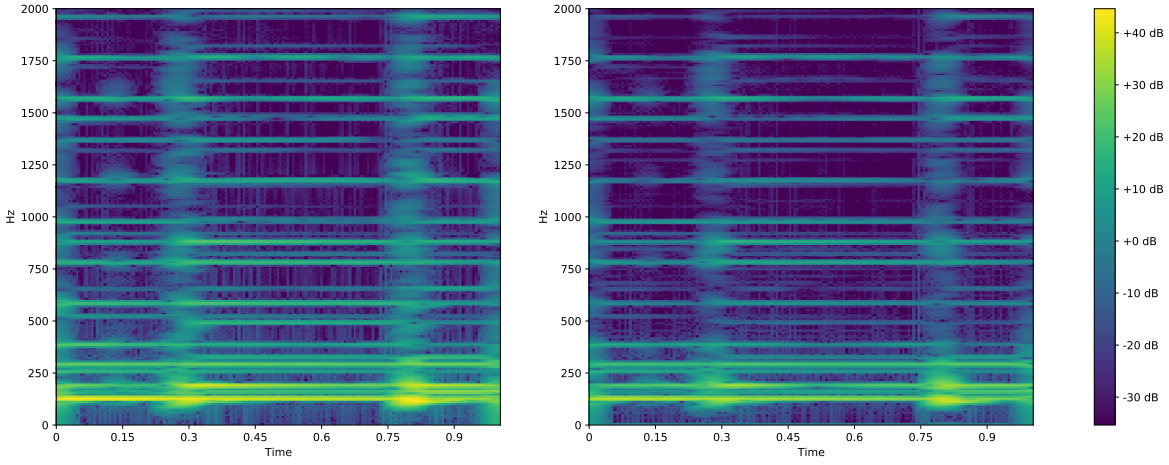
(b)

Figure 3.2: Comparison between recorded signals and the network output in time domain. The image shows 0.05 seconds of audio. The blue one refers to the signal recorded with the pick-up, the orange is obtained using the SM57 microphone and the green one is the RNN output. (a) refers to a guitar strumming audio, (b) is a recording of single notes of an "A blues" scale. All signals are normalized between -1 and 1.

The analysis of the network's audio output is carried out both in time and frequency domain. Figure 3.2 shows results in time domain of two different guitar playing styles. The first image refers to an audio characterized by strumming chords. As it can be seen the RNN signal follows quite well the microphone one, however it struggles during abrupt changes (which correspond to high frequency components), even if the overall trend is respected. For what concerns a single note, in Figure 3.2b we can notice a similar situation; in fact the model is able to follow the target but it is smoother, remarking once again the inability of the model to capture higher frequencies. To sum up, these results



(a)



(b)

Figure 3.3: Spectrograms of microphone and network output signals. The image refers to 1 second of audio. For both images we have the microphone signal on the left and the network's output on the right. (a) refers to the entire audio bandwidth, (b) refers to a low-mids frequency band (0 Hz - 2 kHz).

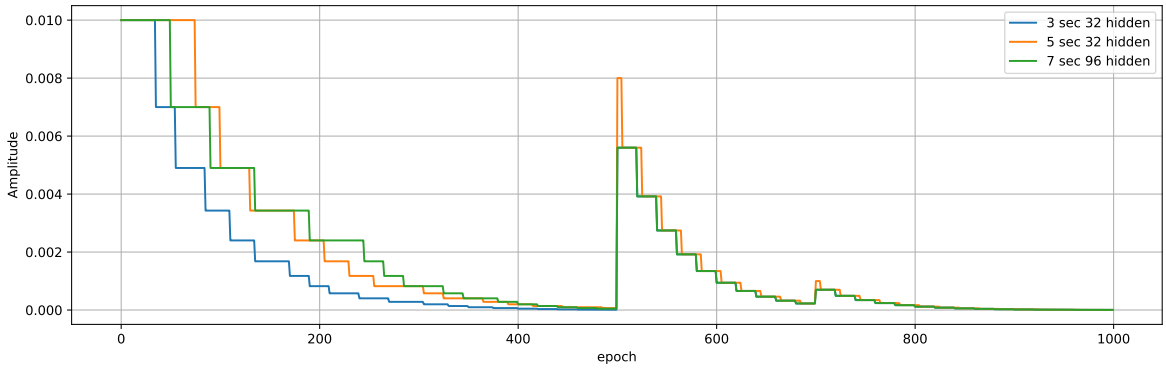
suggest that the network performs in a correct way on the low frequency component and

it has a difficulty in reproducing the higher part of the spectrum.

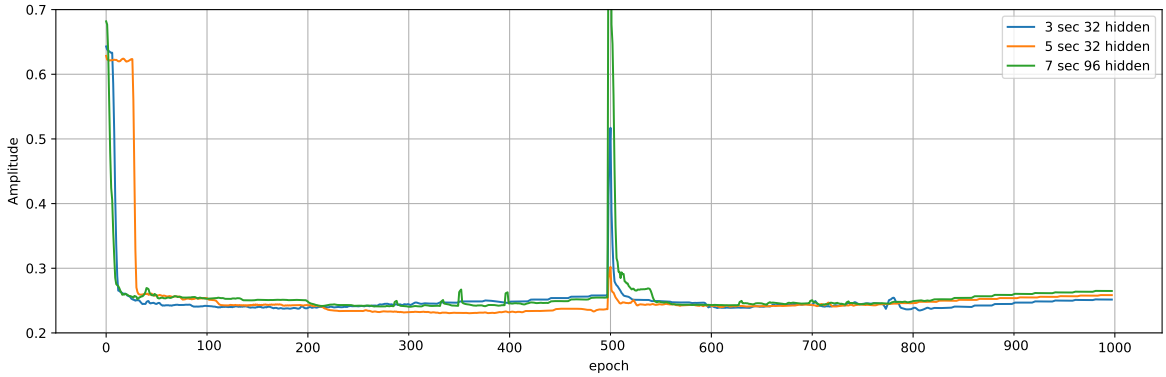
Another aspect we want to deal with concerns some comments about Figure 3.3 which shows the spectrograms of the network's output, both the complete audio bandwidth and a limited lower part between 0 and 2 kHz. On the one hand, Figure 3.3a demonstrates the fact that those frequencies higher than 3kHz are attenuated in the RNN output with respect to the target, reflecting the results obtained in the time domain. The main reasons of this problem correspond to the fact that the model is not complex enough to capture all the high frequencies characteristics of the target signal, or we do not have enough data for the training. On the other hand, Figure 3.3b shows the network performance in terms of low and mid range frequencies (80Hz - 2kHz). It is appreciable how these components are well represented in the model output.

These statements are justified also by the learning curve of the analysed models' majority; in fact Figure 3.4 presents a comparison between learning rate (LR), training and validation curve of three different models. Specifically Figure 3.4a describes the adopted learning rate (LR) for each model by changing the initial learning rate value (LR_i) in the training process. At the beginning LRi is equal to an initial value of 0.01 for all models. Moreover in order to avoid possible local minima of the loss, in epoch 500 and 700 the learning rate corresponds respectively to $LR = 0.8LRi$ and $LR = 0.1LRi$. Figure 3.4b and Figure 3.4c show training and validation curves respectively, in particular all models' training curves reach a plateau value around 0.25 at epoch 200 and even the learning rate's change in epochs 500 and 700 is not able to improve the loss value. A similar trend is observable in the validation graph, a minimum value is reached for all the models at epochs 400 and changes in the learning rate value doesn't improve the validation results. We are reporting only three examples over the complete set of tested models, but we have not manage to go under the plateau value both in training and validation.

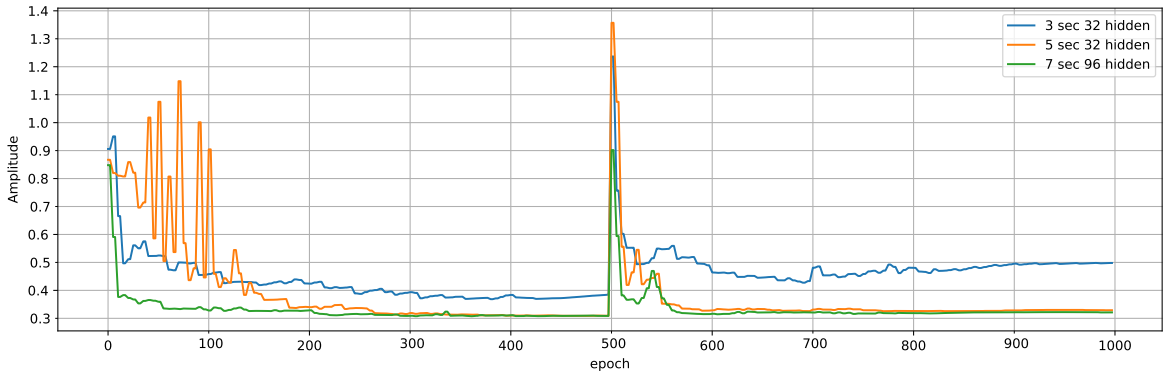
In order to overcome high frequencies limitations of the model previously described, we want to propose a solution based on the consideration of two different network's outputs instead of relying just on a single model. The two summed components are respectively the output of the best performing model in terms of ESR loss and the output of a simpler model (hidden size of 16 instead of 96), which slightly performs in a worse manner in terms of loss but it presents a bigger number of higher frequency components. Once the model is decided, in order to obtain the final audio, we apply a high pass filter with cutoff frequency of 3 kHz to the worst model output. Then we scale it by a constant factor and sum it to the output obtained from the best model. As a result, Figure 3.5 demonstrates the new gained complete spectrum, which now has more energy in the upper range of frequencies with respect to the best performing model one. This change in the



(a)



(b)



(c)

Figure 3.4: Comparison of learning rate, training and validation losses between three models with different segment length and hidden size. (a) Learning rate values in a model training process. (b) Training loss curve. (c) Validation loss curve.

frequency domain is appreciable also from a perceptual point of view; as a matter of fact the addition of the high frequencies makes audible all the overtones of the acoustic steel strings [1]. Moreover, we can hear the characteristic percussive sound of the pick plucking

[1] `https://github.com/EmanueleVoltolini/AI-Powered-Pickup/tree/main/Audio%20Example`

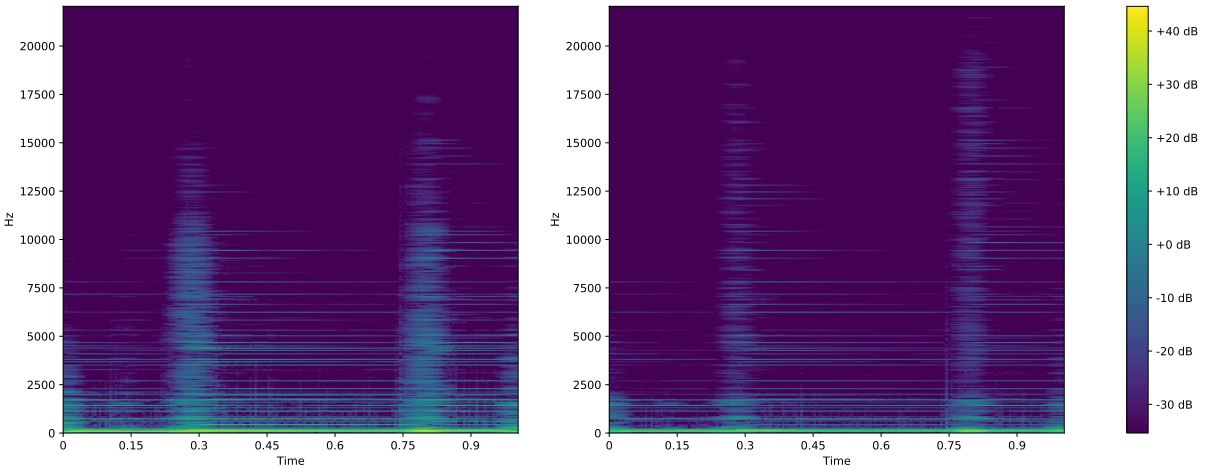


Figure 3.5: Spectrogram of the audio obtained summing the outputs of two different models. We have the target signal on the left and the sum output signal on the right.

the strings. We would like to emphasize that this perceptual analysis is conducted by the author.

Finally, we evaluate the best model's performance in terms of ESR applied to different guitar playing styles. Firstly we segmented the input and target signal in 7 second long chunks; secondly we load the best model and we process the audio; finally we apply the formula 2.3 between the network output and the target so as to obtain the final score. Figure 3.6 shows the overall result where the best ESR is referring to the "single note" audio, which is an A blues scale, probably due to the fact that a note is a simpler signal to model than a full chord.

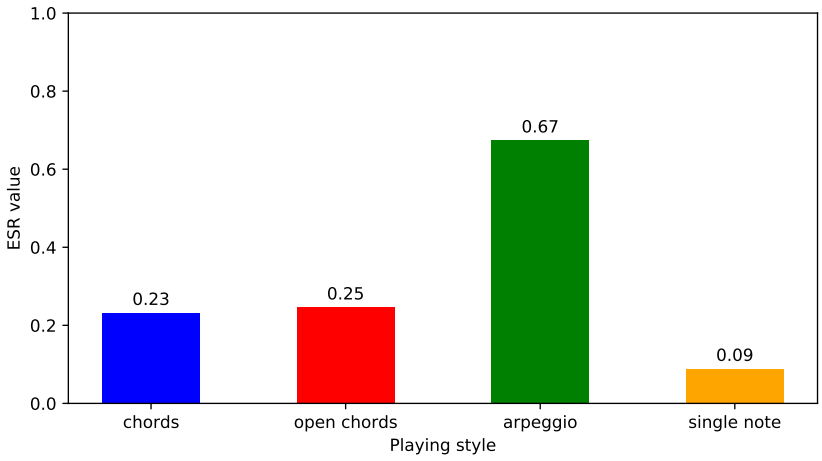


Figure 3.6: ESR comparison between different guitar playing style

In support of this hypothesis, in Figure 3.6 we can see that a lower value of ESR is

reach for "chord strumming" and "open chords" types of guitar playing than a single note, meaning that the network achieves slightly worse. Furthermore we can notice the less significative performance for the "arpeggio" with respect to the other techniques. A possible reason could be a smaller number of minutes of this style than the others in the training set. Moreover, the use of fingers instead of the pick to pluck the strings makes the difference from the majority of the other audios.

3.2. Model comparison

In this paragraph we compare multiple models in terms of ESR and perceptual results. The perceptual evaluation presented is conducted by the author of this thesis.

Table 3.1 shows ESR values obtained by different models in the test process (3.1a) and using "chords" audio (3.1b), one of many signals fed into the network during the training. Therefore, the best overall performing network is the one characterized by a hidden size of 96 and a segment length of 7 seconds (as previously described in this chapter), which has great results both in the test and with the chords audio. Moreover, another remarkable model is the one with 32 hidden size and 5 seconds of segment length, which is really close to the best one. In general, good results can be found in all networks which have 5 or more seconds of segment length and more than 8 hidden size.

hidden size	8	16	32	96
3 second	0.4824	0.3136	0.3097	0.4878
5 second	0.2928	0.2780	0.2614	0.2714
7 second	0.3244	0.2718	0.2749	**0.2604**

(a) ESR values test.

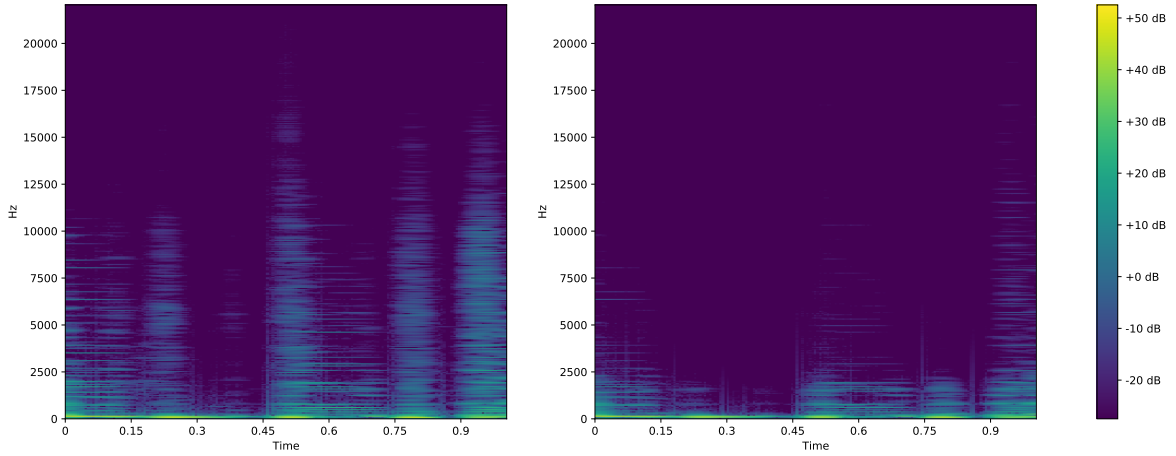
hidden size	8	16	32	96
3 second	0.4087	0.3055	0.3210	0.4827
5 second	0.2757	0.2634	**0.2440**	0.2540
7 second	0.3241	0.2634	0.2649	0.2463

(b) ESR values training chords.

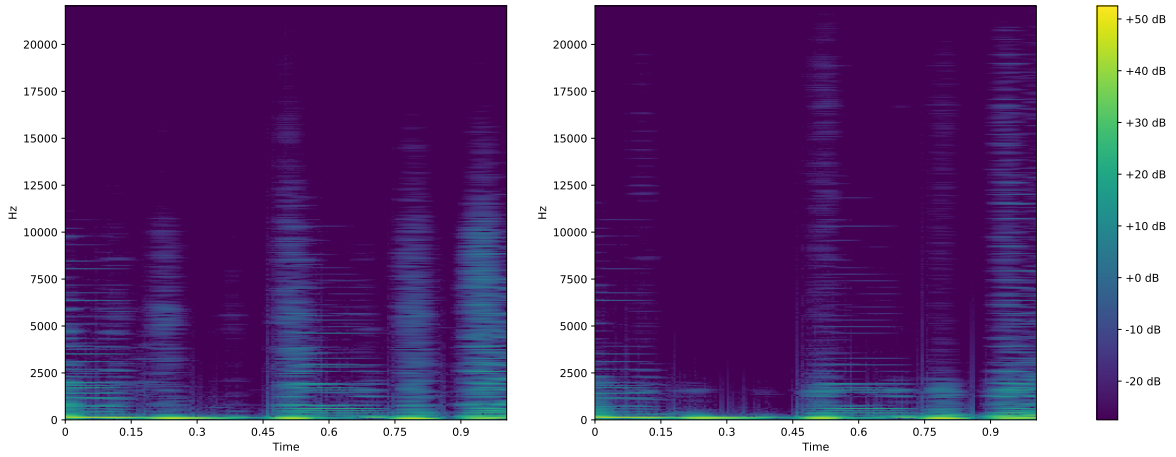
Table 3.1: Error to Signal Ratio (ESR) of the test data (a) and training chords data (b) for each neural network model. In the first raw of both tables we have the dimension of the model hidden size. On the left we have the length of the segment into which the signal is divided.

Until now we analysed networks output only in terms of quantitative scores (ESR), but it is interesting to notice that the best value in the ESR score does not correspond to the best perceptual result. In addition, we would like to point out that all the perceptual analysis and relative comments come from the author of this thesis, but we try to motivate our statements with signals' spectrograms we discuss about.

Since the aim of this section is the comparison among different models, for sake of simplicity we denote model A the network with the 7 seconds segment length and hidden size of 96, model B the 5 seconds segment length and hidden size of 32 and finally model C the 3 seconds segment length and hidden size of 32.



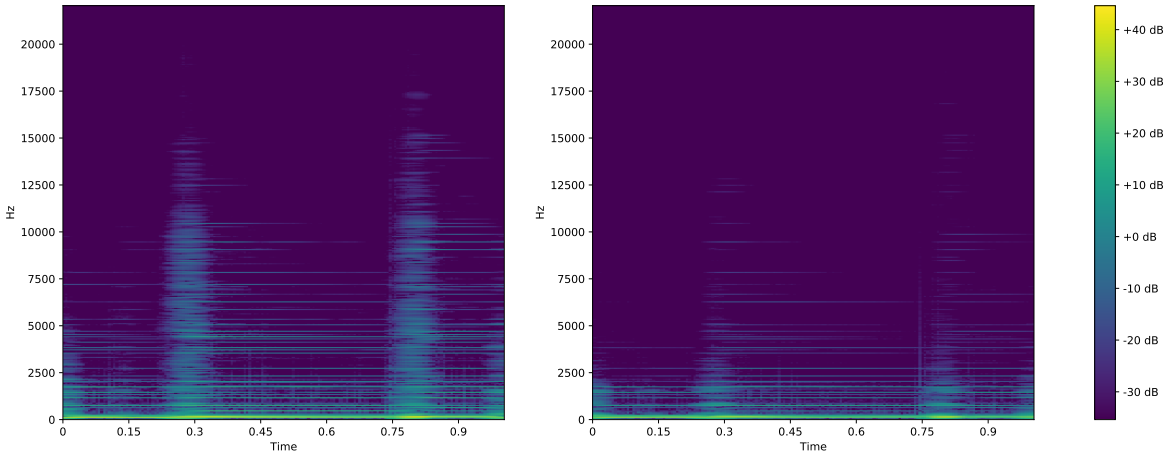
(a) Spectrograms 7 sec 96 hidden model - test audio.



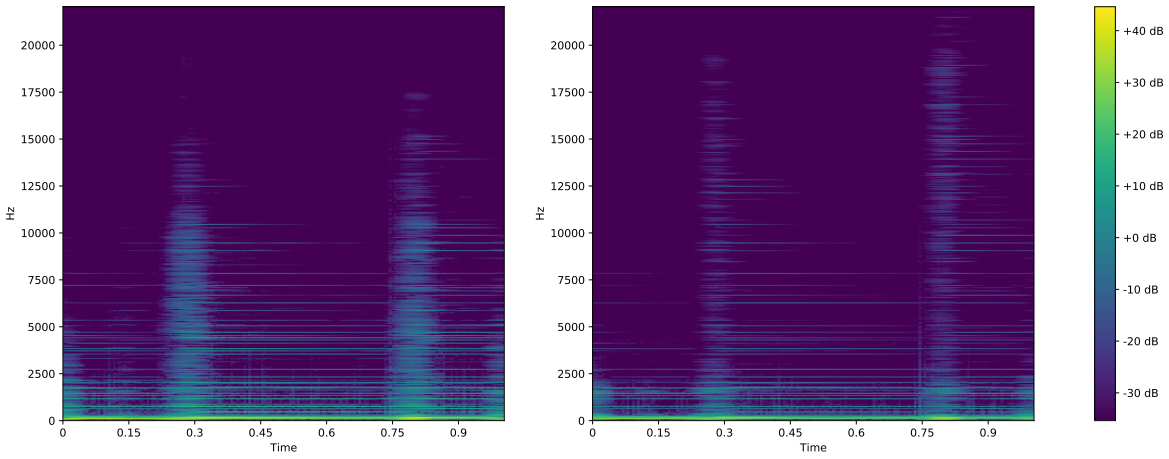
(b) Spectrograms 3 sec 32 hidden model - test audio.

Figure 3.7: Comparison between the spectrograms of two different model for the test audio. In both images the target signal is shown on the left. On the right we have the spectrogram relative to the 7 seconds 96 hidden model (a), and to the 3 seconds 32 hidden model (b).

At first glimpse we can notice from previous tables that models with 3 seconds of segments' length have the worst performance due to their big ESR score. However, from the perceptual point of view, model C seems to be more similar to the target signal of the other two models' audio results. Figure 3.7 and Figure 3.8 show spectrograms of model A and model B outputs respectively in comparison to the model C one. At first sight we can observe that in both the best performing models' spectrograms higher frequencies are attenuated with respect to model C, which has more energy in the upper frequency bands.



(a) Spectrograms 5 sec 32 hidden model - "chords" audio.



(b) Spectrograms 3 sec 32 hidden model - "chords" audio.

Figure 3.8: Comparison between the spectrograms of two different model for the "chords" audio. In both images the target signal is shown on the left. On the right we have the spectrogram relative to the 7 seconds 96 hidden model (a), and to the 3 seconds 32 hidden model (b).

This characteristic is reflected in model C audio timbre because the output appears more

similar to the target microphone recorded signal, in fact we can appreciate the played chords' overtones and the sound of the pick plucking the strings. On the other hand, the other two models' outputs have a darker tone and they lose all characteristic acoustic sound's brightness of metal strings. Furthermore, model A and B present some distortions which are not present in the output of model C.

These results are comprehensible since this thesis work is based on researches concerning sounds deriving from distortion amplifiers and pedals which are really different from acoustic guitar ones.

4 | Conclusions and future developments

This thesis aimed to black-box modeling acoustic guitar pickup - microphone sound using deep leaning model based on a recurrent neural network (RNN) with a long-short term memory (LSTM) unit. The network has shown its ability of following the trend of the target microphone signal in time domain, given as input the pick-up one. However, the model is not able to capture properly the high frequencies components of the spectrum, which are attenuated for frequencies greater than 3 kHz.

To the best of our knowledge, no previous researches are done on this thesis' task. Therefore, our main contribution to the state of the art is given by the demonstration that the acoustic guitar pickup - microphone sound modeling can be done using deep neural networks. Although the results of the single LSTM model are not good enough compared to the microphone recordings, we managed to show that the network is able to follow the target signal in the time domain. Moreover, the proposed solution based on the combination of two different models seems to produce appreciable auditory results, comparable with the original microphone recordings.

As we describe in section 3.2, a possible problem we can highlight corresponds to the fact that the best model output does not correspond to the best perceptual audio results. Since the perceptual analysis is conducted by the author of this thesis, we suggest as a future development to verify the perceptual analysis with a proper test such as webMUSHRA (MUltiple Stimuli with Hidden Reference and Anchor) [34].

We also know there is room for improvements regarding the used data acquisition process, which has been done in a small room with no special acoustic treatment. A possible solution we suggest is to redo the data acquisition process in a controlled environment such as an anechoic chamber. As a consequence all room's spectral components contributions are eliminated. Moreover, we found that all the proposed models reach a plateau in the training process. Because the complexity of the task, a possible cause could be the lack of data. Therefore we propose as a future development to expand the dataset with a new

set of acoustic guitar recordings using the same equipment. In this sense, a further step could be to try modeling different type of microphones, defining different training dataset for each one of them.

As previously mentioned, the model is not able to capture high frequencies. As a consequence, we suggest to investigate new neural network approaches, which may combine the time domain approach and the spectral one. A possible solution could be found in TCNs (Temporal Convolutional Networks) [5] presented by Christian J. Steinmetz and Joshua D. Reiss, in which this new model shows promising results in comparison with the classic RNN one. Another interesting investigation in this sense could be a comparison among different loss functions, finding the best fit for the task we aim to. Some examples in addition to the loss function used in our thesis are found in [55]: log hyperbolic cosine, short-time Fourier transform and multi-resolution STFT. Moreover, would be better to implement some metrics that takes into account also the perceptual aspects in addition to the mathematical parameters.

Following A. Wright [4] and J. Steinmetz's works [5], we suggest to do a real-time implementation of the model to see its computational effort, which could be compared with two or more other neural network models as a further step.

To conclude, we believe this work represents a first step in the black-box modeling of acoustic guitar pickup-microphone sound and we hope that our contribution can help deep neural networks to establish themselves as solid approach to this newer field of research.

Bibliography

[1] Jyri Pakarinen and Matti Karjalainen. Enhanced wave digital triode model for real-time tube amplifier emulation. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(4):738–746, 2009.

[2] Kurt James Werner, Vaibhav Nangia, Julius O Smith III, and Jonathan S Abel. Resolving wave digital filters with multiple/multiport nonlinearities. In *Proc. 18th Conf. Digital Audio Effects*, pages 387–394, 2015.

[3] Felix Eichas, Stephan Möller, and Udo Zölzer. Block-oriented modeling of distortion audio effects using iterative minimization. *Proc. Digital Audio Effects (DAFx-15), Trondheim, Norway*, 2015.

[4] Alec Wright, Eero-Pekka Damskägg, Lauri Juvela, and Vesa Välimäki. Real-time guitar amplifier emulation with deep learning. *Applied Sciences*, 10(3):766, 2020.

[5] Christian J Steinmetz and Joshua D Reiss. Efficient neural networks for real-time analog audio effect modeling. *arXiv preprint arXiv:2102.06200*, 2021.

[6] Johannes Imort, Giorgio Fabbro, Marco A Martínez Ramírez, Stefan Uhlich, Yuichiro Koyama, and Yuki Mitsufuji. Removing distortion effects in music using deep neural networks. *arXiv preprint arXiv:2202.01664*, 2022.

[7] Kurt James Werner. *Virtual analog modeling of audio circuitry using wave digital filters*. PhD thesis, Stanford University, 2016.

[8] Jatin Chowdhury. A comparison of virtual analog modelling techniques for desktop and embedded implementations. *arXiv preprint arXiv:2009.02833*, 2020.

[9] Facundo Bre, Juan M Gimenez, and Víctor D Fachinotti. Prediction of wind pressure coefficients on building surfaces using artificial neural networks. *Energy and Buildings*, 158:1429–1441, 2018.

[10] Xing Hao, Guigang Zhang, and Shang Ma. Deep learning. *International Journal of Semantic Computing*, 10(03):417–439, 2016.

[11] Michael Nielsen. http://neuralnetworksanddeeplearning. com/chap1. html, 2017.

[12] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Overview of supervised learning. In *The elements of statistical learning*, pages 9–41. Springer, 2009.

[13] Horace B Barlow. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.

[14] DE Rumelhart, GE Hinton, and RJ Williams. Learning internal representations by error propagation. in the pdp research group (eds.), parallel distributed processing: Explorations in the microstructure of cognition (vol. 1, pp. 318-362), 1986.

[15] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168, 2006.

[16] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149, 2018.

[17] Suzanna Becker. Unsupervised learning procedures for neural networks. *International Journal of Neural Systems*, 2(01n02):17–33, 1991.

[18] Wikipedia fdeloche. Recurrent neural network. `https://upload.wikimedia.org/wikipedia/commons/b/b5/Recurrent_neural_network_unfold.svg`.

[19] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[20] Saul Dobilas. Lstm recurrent unit. `https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e`.

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

[22] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49. JMLR Workshop and Conference Proceedings, 2012.

[23] Francois Chollet. Autoencoder. `https://blog.keras.io/building-autoencoders-in-keras.html`.

[24] Marc'Aurelio Ranzato, Fu Jie Huang, Y-Lan Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition.

In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007.

[25] Sascha Lange and Martin Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *The 2010 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2010.

[26] Walter Hugo Lopez Pinaya, Sandra Vieira, Rafael Garcia-Dias, and Andrea Mechelli. Autoencoders. In *Machine learning*, pages 193–208. Elsevier, 2020.

[27] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International conference on artificial neural networks*, pages 52–59. Springer, 2011.

[28] Jiayang Xu and Karthik Duraisamy. Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics. *Computer Methods in Applied Mechanics and Engineering*, 372:113379, 2020.

[29] Yifei Zhang. A better autoencoder for image: Convolutional autoencoder. In *ICONIP17-DCEC. Available online: http://users. cecs. anu. edu. au/Tom. Gedeon/conf/ABCs2018/paper/ABCs2018_paper_58. pdf (accessed on 23 March 2017)*, 2018.

[30] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE international conference on computer vision*, pages 5736–5745, 2017.

[31] Matti Karjalainen and Jyri Pakarinen. Wave digital simulation of a vacuum-tube amplifier. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 5, pages V–V. IEEE, 2006.

[32] Zhichen Zhang, Edward Olbrych, Joseph Bruchalski, Thomas J McCormick, and David L Livingston. A vacuum-tube guitar amplifier model using long/short-term memory networks. In *SoutheastCon 2018*, pages 1–5. IEEE, 2018.

[33] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[34] Michael Schoeffler, Sarah Bartoschek, Fabian-Robert Stöter, Marlene Roess, Susanne Westphal, Bernd Edler, and Jürgen Herre. webmushra—a comprehensive framework for web-based listening tests. *Journal of Open Research Software*, 6(1), 2018.

[35] Dimitrios Giannoulis, Michael Massberg, and Joshua D Reiss. Digital dynamic range compressor design—a tutorial and analysis. *Journal of the Audio Engineering Society*, 60(6):399–408, 2012.

[36] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[38] BS Series. Algorithms to measure audio programme loudness and true-peak audio level. 2011.

[39] Sercan Ö Arık, Heewoo Jun, and Gregory Diamos. Fast spectrogram inversion using multi-head convolutional neural networks. *IEEE Signal Processing Letters*, 26(1):94–98, 2018.

[40] Marco A Martínez Ramírez, Emmanouil Benetos, and Joshua D Reiss. A general-purpose deep learning approach to model time-varying audio effects. *arXiv preprint arXiv:1905.06148*, 2019.

[41] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. *arXiv preprint arXiv:1806.03185*, 2018.

[42] Fabian-Robert Stöter, Stefan Uhlich, Antoine Liutkus, and Yuki Mitsufuji. Open-unmix-a reference implementation for music source separation. *Journal of Open Source Software*, 4(41):1667, 2019.

[43] Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis Bach. Music source separation in the waveform domain. *arXiv preprint arXiv:1911.13254*, 2019.

[44] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. Cnn architectures for large-scale audio classification. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 131–135. IEEE, 2017.

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan

Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[46] ovation-celebrity-cs24-4-g. `https://www.ovationguitars.com/guitars-six-strings/CS24-4`.

[47] Shure sm57. `https://www.shure.com/it-IT/prodotti/microfoni/sm57`.

[48] Roland duo-capture ex. `https://www.roland.com/it/products/duo-capture_ex`.

[49] Cyril Cappi, Camille Chapdelaine, Laurent Gardes, Eric Jenn, Baptiste Lefevre, Sylvaine Picard, and Thomas Soumarmon. Dataset definition standard (dds). *arXiv preprint arXiv:2101.03020*, 2021.

[50] Allen H Renear, Simone Sacchi, and Karen M Wickett. Definitions of dataset in the scientific and technical literature. *Proceedings of the American Society for Information Science and Technology*, 47(1):1–4, 2010.

[51] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[52] Ivan W Selesnick and C Sidney Burrus. Generalized digital butterworth filter design. *IEEE Transactions on signal processing*, 46(6):1688–1694, 1998.

[53] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[54] Nvidia geforce gtx 1080 ti. `https://www.nvidia.com/en-gb/geforce/graphics-cards/geforce-gtx-1080-ti/specifications/`.

[55] Christian J Steinmetz and Joshua D Reiss. auraloss: Audio focused loss functions in pytorch. In *Digital Music Research Network One-day Workshop*, 2020.

List of Figures

List of Tables

Appendix A

All the code relative to this thesis work can be found at:

`https://github.com/EmanueleVoltolini/AI-Powered-Pickup`

Acknowledgements

I would like to give my thanks to my supervisor Professor Fabio Antonacci.

I would express also my gratitude to my co-advisor Sebastian Gonzalez who guided me through all the thesis work pushing me to do more and enjoy what I'm doing.

I thank my colleague Alessio for all the help and suggestions. Moreover, a special mention goes to Paolo who has been a fellow thesis student and a good friend, which was always been ready to help me. Another mention goes to my friend Francesca who assisted me in all the writing of the thesis. Finally, I want to express my immense gratitude to my friends and colleagues Cecilia, Giacomo and Martino for the wonderful journey that this master degree has been with them.

Another special mention goes to my girlfriend Chiara, you have always been there to back me up in this last period and throughout my journey at Polimi, even when you didn't know you were doing it.

A special thanks and "argh!" to my fellow friends of Appa Pirata, what chaotic years have been.

From the bottom of my heart I want to thank my family, for all the support in these years and in my all life and for all the good they wanted me.