Faculty of Sciences and Tecnology

Department of Informatics Engineering

Deep Learning for Dynamic Music Generation

José Maria da Costa Simões

Dissertation in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems advised by Prof. Fernando Penousal Machado and Prof. Ana Cláudia Rodrigues. Faculty of Sciences and Technology / Department of Informatics Engineering

June 2019

UNIVERSIDADE Ð COIMBRA

Abstract

In the last decade, Deep Learning (DL) algorithms have been increasing their popularity in several fields such as computer vision, speech recognition, natural language processing and many others. DL models, however, are not limited to scientific domains as they have recently been applied to content generation in diverse art forms - both in the generation of novel content and as co-creative tools. Artificial music generation is one of the fields where DL architectures have been applied. They have been mostly used to create new compositions exhibiting promising results when compared to human compositions. Despite this, the majority of these artificial pieces lack some expression when compared to music compositions performed by humans. In this document, we propose a system capable of artificially generating expressive music compositions. Our main goal is to improve the quality of the musical compositions generated by the artificial system by exploring perceptually relevant musical elements such as note velocity and duration. To assess this hypothesis we perform user tests. Our results suggest that expressive elements such as duration and velocity are key aspects in a music composition, making the ones that include these preferable to non-expressive ones.

Keywords

Machine Learning (ML), Artificial Neural Network (ANN), Deep Learning (DL), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), Note Sequences, Music Composition, Melody, Velocity, Duration.

Resumo

Na última década, a popularidade dos algoritmos Deep Learning (DL) tem vindo a aumentar em diversos campos como a visão computacional, reconhecimento de fala, processamento de linguagem natural, entre muitos outros. No entanto, a aplicação de algoritmos DL não se limita a domínios científicos, tendo estes sido recentemente usados para geração de conteúdo em diversas formas artísticas - tanto na geração de conteúdo novo como ferramenta de auxílio criativo. A geração artificial de música é um dos campos artísticos onde arquitecturas DL têm vindo a ser aplicadas. Estas têm sido maioritariamente usadas para criar novas composições, revelando resultados promissores quando comparadas a composições humanas. Porém, a maioria destas músicas artificialmente geradas apresentam falta de expressividade comparativamente a músicas tocadas por humanos. Neste documento, propomos um sistema capaz de gerar artificialmente composições musicais expressivas. O nosso principal objetivo passa por melhorar a qualidade das músicas geradas pelo nosso sistema através da exploração de elementos musicais perceptivamente relevantes, como velocidade e duração. Para avaliar esta hipótese, realizámos testes de utilizador. Os resultados obtidos sugerem que elementos expressivos como a velocidade e duração são fatores-chave numa composição musical, fazendo com que as músicas que incluam estes elementos sejam preferíveis a composições sem qualquer tipo de expressividade.

Palavras-Chave

Machine Learning (ML), Artificial Neural Network (ANN), Deep Learning (DL), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), Sequências de Notas, Composições Musicais, Melodia, Velocidade, Duração.

Contents

Acronyms

AF	Activation Function. 17, 19, 20
ANN	Artificial Neural Network. xii, 3, 16, 18, 19, 26, 28, 31, 32, 54, 92
CNN	Convolutional Neural Networks. 26
DL	Deep Learning. 2, 26–28, 36, 88, 92
FFN	Feed Forward Network. 18, 20
GAN	Generative Adversarial Networks. 26
LSTM	Long Short-Term Memory. xiii, 3, 22, 24, 26–32, 34, 35, 38–40, 55–57, 88, 92, 93
ML	Machine Learning. 2, 3
RNN	Recurrent Neural Networks. 3, 18–22, 24, 26, 29, 31, 32

List of Figures

List of Files

Chapter 1

Introduction

The efficiency of machine learning algorithms is remarkable when considering its applications - from content filtering to recommendations, these algorithms exhibit their adaptability and effectiveness when applied to user-based services [40]. Besides these commercially-oriented applications, artificial learning methods provided good results in different areas, especially in content recognition and identification (such as faces or objects) and even text transcription from speech. All this is possible using Deep Learning (DL) - a branch of Machine Learning (ML) - which allows computational models composed by multiple processing layers to learn representations of data with several abstraction levels [40].

Although there's not a consensual definition of the term (i.e. Deep Learning) [16], its potential traces back to the date of its emergence in 2006 when a DL architecture overcame standard classification techniques for manual features and patterns in images [25]. This success and re-emergence can be justified by remarkable technological advances in addition to the current massive amount of data available to the average user, which didn't exist before [16].

Nevertheless, the plasticity of DL algorithms isn't limited to *simple* classification problems. In fact, a new field - that has been increasing in the last years [16] - holds the application of these techniques to content generation such as image, text and music. Regarding the scope of the present work, we'll be exploring these methods for music generation.

Even though the primitive nucleus of this domain isn't recent at all [43], music composition aided by ML algorithms has been drawing more attention in recent years and hence development, although in some partial attempts [16]. In 1958, Iannis Xenakis generated artificial compositions through Markov Chains. This system consisted of a set of probability matrices that resulted from an extensive analysis of musical elements and corresponding formalization. This establishment of laws assigned a certain probabilistic degree to a musical note (or set of notes) of being, or not, played at a given time [63]. A great restraint of this model was that sequences of notes were limited (i.e. highly predictable), and that is due to the fact that Markov Chains trained with a set of musical compositions could only generate subsets of the first one so that note sequences would be highly identical to the training set. In the last quarter of that same century, the same goal was pursued, yet this time via Recurrent Neural Networks (RNN) (see section 2.2.1). Even though this approach suppressed Xenakis' Markov Chains limitations, it exhibited a lack of coherency regarding the generated compositions mostly due to its short-term memory absence [43].

Some years later, in 2002, Douglas Eck and Jürgen Schmidhuber presented a quite capable solution to overcome this melodic gap [27]. As mentioned above, RNNs show good evidence of learning temporal structures yet they fail when it comes to short-term claims. The authors based their research on the recognized efficiency of LSTM units in domains where temporal measurements are crucial and in context-sensitive learning [27]. Their results were considered substantially positive [27] in the way that it was shown that by using LSTMs the network acquires the capability of learning the general structure of a music composition. Besides that, the network showed to be capable of generating new pieces based on the training set without sounding necessary like it in its fullness [27].

The power of LSTMs drew the attention of common users in 2015 when Andrej Karpathy's article *"The Unreasonable Effectiveness of Recurrent Neural Networks"* had massive disclosure. Additionally, in 2016 Google Brains' Project Magenta - which comprised artificial content generation - opened its doors to the average users, sharing the code and protocols of some of its projects [43]. All these factors enabled a wider investigation of LSTMs capabilities concerning music generation.

As a matter of fact, the quality of the generated compositions can only be measured if some stable metrics are defined. Whereas in *simple* classification algorithms the output can be easily calculated regarding its efficiency, it all becomes a lot less linear when it comes to the artistic creation of music.

By the time we're writing this document, different types of ANN have been studied regarding their ability to learn musical compositions and generate new ones, but only a few have taken into consideration other aspects besides note sequences. Designing an ANN to produce extremely good melodies has been proved to be a difficult task as it is hardly comparable to human compositions. However, some research has been conducted on this subject taking into consideration other elements besides note sequences. Aforementioned Google Brain's Team - Project Magenta [1] - have been developing several open source architectures to explore the role of ML in music and other forms of art. Regarding Project Magenta's research on artificial music composition, there's a clear interest in exploring different musical features like expressive timings and dynamics [10]. It is clear to agree that notes are an important part in a music composition - if not the most important. Nonetheless, we believe that by exploring other aspects of compositions such as expressiveness and musical performance - and the impact they might have in music - we'll be able to better understand not only their importance but also how they might be used to improve a music composition.

In this document, we detail a model capable of generating artificial expressive compositions. The generated compositions are tested along with human performances in order to study the influence that expressiveness has on musical compositions.

1.1 Contextualization

Before going any further in this document, we hereby aim to clarify some changes applied to the dissertation's scope. Originally, the main goal of the proposed work was to generate multimedia artefacts through evolutionary - or machine learning - algorithms guided by neurofeedback data. As we can acknowledge, these can be divided into two major components: artefact generation and neurofeedback data collection. However, the preliminary work carried through the first half of the semester - and corresponding results - led to a re-definition of the main problem. These plan refinements were needed due to some following constraints:

First, the artefact generation has proven to be complex enough. As the generated artefact (in our case, music) was to be guided by human emotions (i.e. neurofeedback tracking), the quality of the output had to meet some criteria to induce a specific emotion [37]. This can be easily understood with two distinct examples: To induce a bad mood (e.g. anger) the algorithm could generate random notes following no harmonic rules so that when performed would be perceived as completely unpleasant to our ears. This could probably lead to some kind of discomfort, maybe even upset the listener to the point of inducing anger, and yet the output couldn't be considered tonal music (i.e. compositions centered around one tonal pitch - see section 2.1.1), but just an arbitrary sequence of notes. Another case is the importance of diversity or variation in a music composition, although as simple as it can be. To illustrate this example, let's say that the listener enjoys listening to a single piano track but not to the point of loving it. If submitted to a long period of piano tracks generated by the algorithm (specially if containing repetitive melodic phrases) , he or she would eventually get bored or even fatigued and this would not necessarily mean that the music was not that pleasant, but rather repetition led to a different mood [28]. As detailed further in this document (section 3.4), we depict the complexity involved in multimedia artefact generation, especially music.

Secondly, the neurofeedback devices under test exhibited lack of reliability. The device used to conduct these experiments was the Emotiv Epoc, from which no raw data could be extracted. This meant that there wasn't neither a promising or convenient way of obtaining local measurements of the brain's activity and hence a reliable way of extracting emotional behavior data.

Facing these major predicaments, the main goal was partially redesigned, focusing entirely on music generation itself. This means that the neurofeedback component was excluded from the work plan. This changes not only allowed us to reallocate our efforts but also to deeply explore the complex yet challenging field that is music generation through artificial neural networks and how expressiveness influences a music composition.

Although the first quarter of the present semester was spent on exploring the validity and practical utility of the already mentioned neurofeedback device, it was crucial to draw conclusions on the topic. A more detailed description of this opening investigation and tests can be found as an appendix [Appendix A (9)] at the end of this document. With this, we intend to progress throughout this document according to the newly defined goals having no interference with a problem that was already left behind.

1.2 Goals

As stated before, expressive elements - such as velocity and duration - are thought to be key aspects of music performance, affecting the expressiveness of the piece and the emotions that it evokes [29][60]. Nevertheless, most work in the field of music generation focuses mainly on melody and tends to ignore duration and velocity [18].

The main goal of this thesis is to contribute to the advance of the state of the art in the field of artificial music generation by building a model capable of learning to generate musical compositions that include melody, velocity and duration information.

To attain this goal we must address several subgoals, namely:

- The need to create a base model able to generate melodies (note sequences);
- Expand upon this model making it capable of generating melodies with expressive variance (velocity and duration).

Furthermore, to assess the impact of velocity and duration we conducted a wide variety of tests (i.e. User-Testing).

As mentioned previously, our work is based on the premise that velocity and duration are important aspects of music. We began by assessing their impact by depriving human created musical pieces of this information and evaluating the response of the users. Then we conducted the same kind of analysis on musical pieces generated by our system, assessing how velocity and duration affect the perception of quality of the pieces. By comparing the observations regarding these two groups we'll be able to better understand not only if our model generates good music pieces but also how expressiveness impacts the pieces.

Furthermore, as a secondary goal we apply artificial elements generated by our model to human composed melodies, and test how artificially generated expressiveness may affect music pieces composed by Humans.

1.3 Document Structure

The present document is organized as follows:

- **Background**, where we detail the theoretical basis for this thesis regarding Musical Matter (Music Theory and Music Psychology) and Artificial Neural Networks (particularly Recurrent Neural Networks and Long Short-Term Memory).
- **Related Work**, where we highlight the most relevant/recent works in the domains of Music Generation Algorithms, as well as the observed constrains (Main Shortcomings).
- **Preliminary Work**, where we present the first experimental results along with a detailed explanation regarding the networks' architecture and process of generating melodies (note sequences). In addition, we provide an analysis regarding these preliminary results, observed constraints and considerations to develop a prototype of the model.
- **Prototype Development**, where we detail the development process of the model, explaining its basic structure as well as some decisions made along its conception. We also present an analysis of the artificially expressive compositions generated.
- **User-Testing**, where we discuss the methods used to create a validation process, also an overview of the gathered results.
- **Discussion and Future Work**, where we discuss the obtained results, providing an analysis of the conducted work as well as some considerations on future work
- **Work Plan**, where we give a thought to the progress made in this first semester, also presenting the work plan for the second semester.
- **Conclusion**, where we reflect upon the work that has been made and its results, in a final note on the progress of the developments conducted during both semesters.

Chapter 2

Background

2.1 Musical Subjects

2.1.1 Music Theory

Music has an impact on a person's state of mind, having an effect on one's mood and behaviour and has proved to be a tool for self-regulation [32]. This influence on the human response depends on multiple factors, as it is a subjective field. For the present work, these effects are partially relevant not only to measure a composition quality but also how it leads to emotions throughout our senses and perception. We briefly address this topic in section 2.1.3 (Music Psychology).
We hereby provide a concise explanation of musical practices and its underlying systems. We do not intend to explore technical approaches or to delve into deeper theoretical details. Rather, we'll be describing the basis of the musical theory. The following topics of Music Theory are based on Catherine Schmidt-Jones and Russell Jones' *"Understanding Basic Music Theory"* (2007) [53], where they provide a compact and clear introduction on this subject.

In general, musical compositions - as complex as they might be - can all be expressed by a standard representation. Logically, the performance of expressiveness has to be excluded when writing down a composition as it is intrinsically related to the performer interpretation. Apart from that, all information can be represented by special universal symbols. Musical sheets can be seen as a language and can be interpreted in the exact same way. For such, a system of diverse elements representing actions, context and terms is used to describe musical compositions. Below we list the more important notations and terms to keep in mind for the current work.

2.1.2 Notations

The Staff

The staff is the basis for written music, usually known as *common notation* [53]. It consists of five horizontal lines in which most notes are placed (either on the lines or on the spaces between them). The staff may also have vertical bar lines on it which divides the staff into sections called measures or bars. To define the end of a certain section a double bar line is used to point its end (the same applies for the end of the composition). Staves are read from left to right. We can have more than one staff at a time, especially when there's more than one instrument to be played. In such cases, these staves are placed below one another [53].

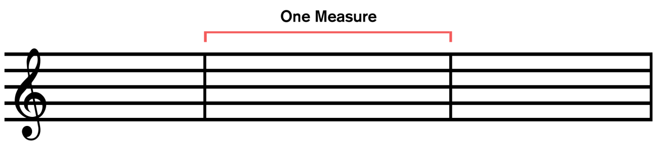


Figure 2.1: Single Staff

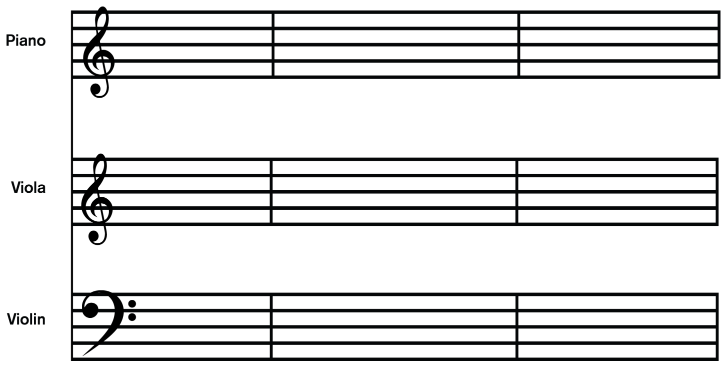


Figure 2.2: Multiple Staves

The Clef

The clef is the very first symbol that appears at the beginning of a single staff. Clefs are extremely important to *decode* the rest of the composition as they define where each of the existing notes is placed on the staff (i.e. on or in-between the parallel lines). Musical compositions are written in either Treble Clef or a Bass Clef.

The Treble Clef defines that the second line from the bottom holds a *G* note (the figure below shows that it is placed where the symbol curls around). The Bass Clef works in a similar way, telling us that the second line from the top holds an *F* note. With these declarations, we can easily relate any symbol to a specific note by identifying where it is placed as the notes sequence are invariably the same. The reason behind having different positions for the notes - defined by these different Clefs - is because each note can have a different pitch, as we're about to see [53].

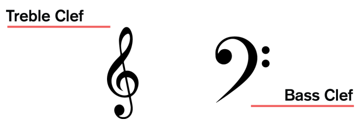


Figure 2.3: Treble and Bass Clefs

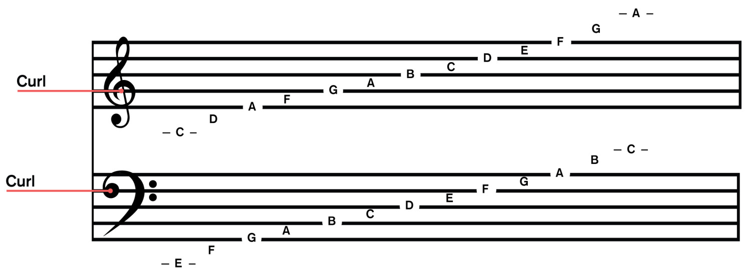


Figure 2.4: Treble and Bass Clefs on the Staff and Notes' Positions

Pitch

The pitch of a note is how loud it sounds, regarding its frequency. With this, we have seven notes represented by letters: *A*, *B*, *C*, *D*, *E*, *F* and *G* (*la, si, do, re, mi, fa* and *sol* in Romance languages). These seven letters name all the natural notes within one octave. The octaves are basically the same set of these notes (along with the Sharps and Flats), but twice or half the frequency of vibration of the other. Sharp and Flat notes are between two natural notes. A Sharp sign means that the note is one half step higher than the natural note and a Flat sign means it is one half step lower.

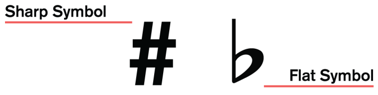


Figure 2.5: Sharp and Flat Symbols

Figure 2.6 depicts these notes on a piano. The white keys are natural notes, whereas the black ones are either Flats or Sharps. There are two important factors to keep in mind: The E natural and F natural are one half step apart, meaning that there's no other note between them. Also, once Sharps and Flat notes are relative to the natural note, some Sharps and Flats are the same note, so the symbology depends on the natural note. Including these mentioned pitches, we then have five more pitches leaving us with a total of twelve pitches within an octave.

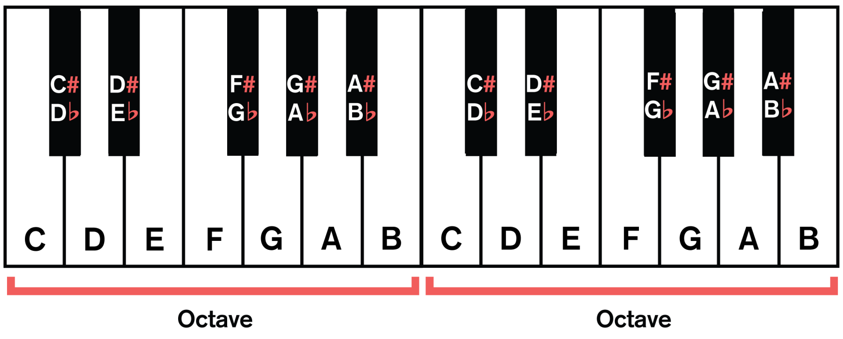


Figure 2.6: Notes and Sharps/Flats on a Piano

Key Signature

The key signature comes right after the clef symbol on the staff. Key signatures can either be a Sharp or Flat symbol and we can have more than one on the staff. Depending on the symbol, it basically tells us that a specific note is either sharp or flat (or natural note, if there's none) and for the current sheet and composition it should always be played like that. This declaration affects all the notes on the line or space in which it is positioned, that's why we can have more than one symbol.

Duration

Another critical factor that must be described is a note's duration in written music. To get a reliable summary of this topic, Figure 2.7 shows the most common note lengths. A note's length is determined by its shape (symbol) and visual adornments. As we can see from Figure 2.7, we can have a full note (whole note) or divisions of it. Note lengths work just like fractions in arithmetic in the sense that two half notes or four quarter notes last the same amount of time as one whole note. As stated in section 2.1.2, staves can be divided into measures (or bars). If we have a whole note written in one measure, it basically means that that specific note should sound (i.e. last) from the beginning to the end of it.

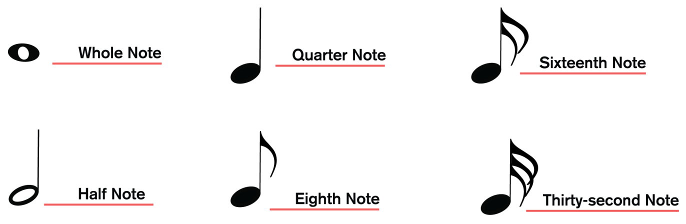


Figure 2.7: Most Common Note Lengths

Although having a whole note or two half notes is the same in terms of duration and occupation of a certain measure, it's different when it comes to how those notes are played. For instance, a whole note should be played (i.e. hit) only at the beginning of the measure, letting it sound until the end of it. But if we have two half notes, one after each other, it means that when we actually play those notes, we hit the first note lasting half of the measure and then hit the second note half of a measure after hitting the first one.

When we have a sequence of notes, the adornments on the notes (called flags) can be grouped (connected) by beams, as illustrated in Figure 2.8.

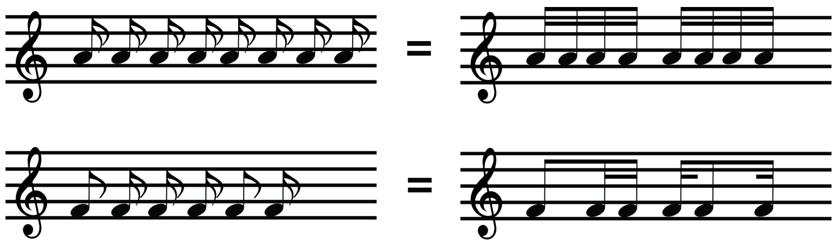


Figure 2.8: Notes with Beams

To sum up this short description of the basics of written music and its notations, we can put it all together: To find out the pitch of a note, we first look at the clef (that tells us where all notes are from that) and the key signature (so we can see which notes are affected by it), and we then find the notes on the line or space. The higher a note sits on the staff the higher it sounds, and the way it is drawn defines its duration.

Definitions

Now that we've introduced the primary notations, we define some essential terms associated with musical performance, often called the basic elements of music [53]. Again, we reiterate that there's a lot more to be said about these subjects. We'll be focusing on a clear definition of these elements as a way of better understanding the current work.

To have a common music composition, one must have a sequence of notes, although music genres or styles differ and each one has its own flow and a musical composition can be composed only by percussion instruments. Still, we'll be centring our attention in compositions that follow the traditional aspects and elements of western music. As a song can be created containing only a single instrument and knowing that even with one single instrument note patterns are diverse, these various elements can be grouped into different types, as we list below.

Melody: Is a sequence of notes with a single voice (instrument or vocal). A melody can be either monodic - or monophony - (only one note is played at the same time) or polyphonic (more than one note can be played at the same time) [16].

Polyphony: Multiple voices (intended for more than one voice or instrument) [16].

Accompaniment: (to a given melody). This can be a counterpoint (composed of one or more melodies) or a sequence of chords (i.e. a harmonic set of pitches played simultaneously) that provides some associated harmony [16].

Harmony: When more than one pitch is sounding at the same time, the result is harmony [53].

So, with those terms pointed out, we can have music based only in rhythm, with no pitches involved. We can also have a piece of music that is just a melody or even a melody with rhythm accompaniment. But as soon as there is more than one pitch being played at the same time, we then have harmony. Harmony doesn't mean that one must necessarily have chords sounding at the same time: we only need to have two different pitches together to have harmony [53].

It's also important to note that harmony doesn't have to be completely "harmonious", which may lead to dissonance. We'll be discussing this in the next section.

As advised by Catherine Schmidt-Jones and Russell Jones [53], below there's a suggested listening list to better understand the difference of the mentioned terms and elements:

Melody (Monophony): Gregorian chant - *"Dies Irae"*

Melody (Polyphonic): Bizet's Carmen - *"March of the Toreadors"*

Polyphony: Johann Bach - *"Fugue in G minor"*

When it comes to songwriting, or even playing along or improvising, there are some rules that can be followed in order to have a piece of pleasantly sounding music. Therefore, to be familiar with the particular notes that a specific music piece is likely to have musicians study scales, that basically are a set of expected pitches for a given piece [53].

The logic behind these rules (i.e. scales) will not be addressed in this document not only because it is an extensive field in music theory's domain but also due to some preliminary results that strongly suggested that this rules can be learned via observation of its applications and not by necessarily learning them directly, as we demonstrate in Chapter 4.

However, these musical arrangements and set of notes that *should* or not be played in a given musical scenario are highly correlated with the human brain and perception. It is then an important aspect to look at in order to achieve a better notion of how to define a good piece of music, more specifically a generated one as in our case. In the next section we discuss the main aspects of how we perceive music.

2.1.3 Music Psychology

Music and its inherent sounds may arouse emotions in humans. Although this seems a common and well-known statement, there's an interesting paradox in this relationship as pointed out by Patrick N. Juslin (2013): Looking at each one separately, we have music as an abstract form of art that is totally not an everyday life concern and, on the other side, human emotions, biologically evolved reactions related to human survival [37]. Still, one seems likely to induce the other, sometimes at a profound level as the areas of the brain that are activated by emotional music (i.e. a music piece that arouses any kind of emotion) are similar to those associated with such strongly rewarding activities and stimuli as games, drugs, food and sex [32]. However, this is not an absolute event as not all music pieces arouse emotion, being this dependent on the music itself and the listener's own tastes, temperament or even culture. It is relevant to clarify the difference between perception and arousal of emotions, as one may simply perceive a particular emotion in the music and not feel some emotion as a response to it [37].

Even though different pieces of music (i.e. genres or music style and its elements) don't please every person in the same way, there are some factors that may lead to an agreement concerning the minimum requirements to have an audible (i.e. tolerable) music composition. We mentioned before that a western music composition can be written in a certain key - a guide to following notes or chords. That being said, music in a particular key tends to use only some of the many possible notes available (i.e. regarding all the existing notes) within a scale associated with that specific key that can be seen as expected pitches for that specific piece of music [53]. In informal terms, these notes sound *right* when played together according to a given key, and within a scale they are usually arranged from the lowest to the highest (or the other way around) in a pattern that usually repeats within every octave [53].

Every sound - ordinary and every day "noises" - *"comes in its conceivable pitch or group of pitches"* (if we have more than one) (Catherine Schmidt-Jones [53]). Naturally, we can experience a lot of sounds simultaneously, sometimes experiencing the so-called "white noise" that basically is every pitch at once, so that no particular pitch is heard or perceived [53]. With this, we can have through contrast one of the main reasons why music is pleasant to our hears: It can be seen as ordered sound, in the sense that only a few of all the possible pitches are used [53].

Among the possible combinations of note sequences that make a music composition, we define these *pleasant* or *unpleasant* conjugations with two important terms: Consonance and Dissonance. Notes that sound good together when played at the same time are called consonant [53]. We can develop this idea by considering that chords consisting of only consonances sound pleasant and *stable.* On the other hand, notes that are dissonant can sound harsh or unpleasant when played at the same time. Although dissonance may seem like it must be avoided, that isn't necessarily true. First, it depends on our musical background and culture, and that's because different cultures usually apply different note sequences and scales - that's why some musical pieces remind us of some specific country or location, like the Arabic scale or other exotic ones. Secondly, dissonance doesn't always mean that the notes being played would sound totally out of context (or out of tune). In fact, some chords containing dissonance are considered *unstable* and when we hear them we expect them to move on to a more stable chord (moving from dissonance to the consonance that is expected is called resolution) [53]. And this is important to a music composition, as this tension created by dissonance that is later resolved by consonance injects variation and excitement to the music. So this entanglement is important as music that only contains consonance may sound too simple (which might lead to boredom) and music only consisting of dissonance can be difficult to listen to as the tension built is never resolved [53]. Either way, there are some dissonances that are undoubtedly unpleasant to our (western) ears, causing some notes to seem out of tune. This will be taken into consideration when evaluating the quality of the generated music composition.

Finally, we must also save some time to analyze the role of performance in a musical piece. As earlier referred in this chapter, some factors regarding expressiveness aren't described or written on a score, leaving these elements dependent on the performer himself. As described by Pere Ferrera et. al (2005), apart from the main elements that constitute a written musical piece - such as notes and rests -, we can also find some marks or annotations for expressive effects, such as *vivace* or *allegretto* which define basic tempo markings. However, not all music elements are explicitly represented on the score, as *"the composer normally annotates only exceptions to the standard performance"* [29]. Considering the expressive notations marked on the score, we can surmise that different notations would result in different interpretations (or impressions) on the same musical piece by applying these performance changes. The possible influence of the performer over a written piece can also be supported by some cultural factors, as throughout the history of music we did not only observe transformations regarding compositional styles but also in the performance practice [13]. Keeping all of this information in mind, it is interesting to observe that some factors affecting how a musical piece is perceived are actually not part of the composition itself (at least the written part of it), but rather subject to various interpretations of the performer.

2.2 Artificial Neural Networks

Artificial Neural Network (ANN) are, in their essence, an engineering approach of biological neurons [55]. Therefore, to reach a good level of understanding of these structured systems it is crucial to acknowledge background information about its central inspiration. In biological terms, a neuron (or nerve cell) is fundamentally a special biological cell that processes information [36]. As described by Richard H. Hall (The Neuron, 1998), *"The neuron is the fundamental unit which makes up a nerve pathway, neural firing (neurotransmitter release) takes place at the level of the neuron and many aspects of the physiology-behaviour relationship can be explained in terms of activity at the neuronal level"* [31]. This is due to the fact that a single neuron has in its own main function the ability to carry electrical impulses (that can be seen as information), resulting in a series of connections known as nervous system [31]. The nervous system monitors and controls the entire body - as for the case of the central nervous system (brain and spinal cord) - by its peripheral divisions which are distributed to all the muscles, organs and tissues [35]. In short, the main neuron structure has three important components: the body, axon and dendrites. The first one holds the information needed for the neuron to produce its own material. As for the last two, they're part of the *input* and *output* process: the signal (i.e. impulse) generated from other neurons are received through the dendrites; after such, the signal generated by the cell body is transmitted to other neurons via the axon [36]. The axons are subdivided into multiple terminals at the end of which we can find synapses - an elementary structure in which neurotransmitters are released to enhance or inhibit the receptor neuron's own tendency to emit electrical impulses. As described by Anil K Jain et al. (1996), *"The synapse's effectiveness can be adjusted by the signals passing through it so that the synapses can learn from the activities in which they participate"* [36].

Keeping this brief explanation in mind, we can now transpose this knowledge to describe the basic architecture of an ANN. Generally speaking, the main goal of these types of systems is to generate an output from one or more inputs that are fully dependent on the connections that form the net. For such, we must understand the behaviour of a simple artificial neuron.

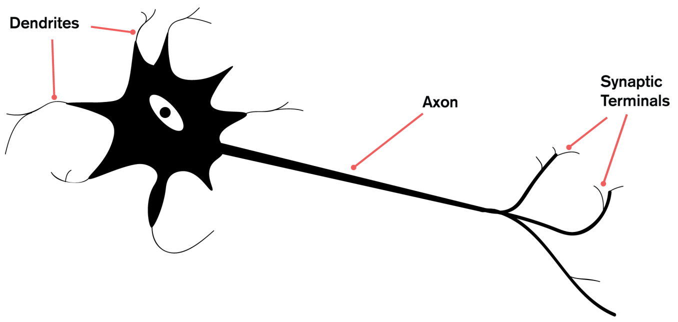


Figure 2.9: Sketch of a Human Neuron. Adapted from Anil K Jain et al. [36]

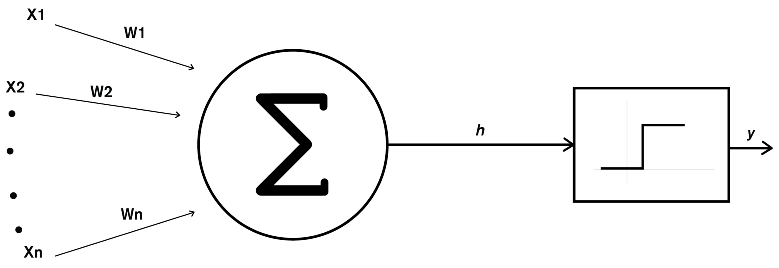


Figure 2.10: Artificial Neuron. Adapted from Anil K Jain et al. [36]

The figure above illustrates a global view of a single artificial neuron. It computes the weighted sum of all the inputs generating the output y according to some Activation Function (AF) defining the threshold. In this specific case the AF used is the Binary step, which sets the output value to 0 if the weighted sum of all inputs is less than 0, or to 1 if it's equal to or greater than 0. There are various types of AFs, and we'll be addressing this topic later in this chapter. Recalling the biological neuron analogy, we can now identify its main components once the weights work similarly to a neuron's synapse, as they operate over the respective associated input. This *control* over the inputs can determine its influence on the desired output. For instance, if we set the weight w_1 to 0, then the input x_1 won't hold any kind of influence on the output due to the multiplication between these two variables regardless of its value. Moreover, if we assume that each input of each artificial neuron corresponds to a specific output of other we would get to a point where we have not only one, but a much higher number of neurons: and that's an artificial neural network.

This set of linked artificial neurons are stacked in three types of layers: The *input* layer, *hidden* layer and the *output* layer. The first layer receives all the input values and goes through the *hidden* layer - where the calculations related to each neuron are performed - all the way down to the *output* layer in which we *collect* the generated result. The *input* and *output* layers must have a defined size according to the problem. For instance, if the input is a vector of three numbers and we want the network to generate only one value then the *input* layer must consist of three neurons and the output only one. As for the *hidden* layer, it can consist of a variable number of neurons. Moreover, we can even have multiple *hidden* layers, each of them containing parallel neurons performing calculations. This way the *hidden* layers of a neural network learn to represent its inputs, firing a set of calculations in order to formulate a final result which will be taken as the predicted output [40].

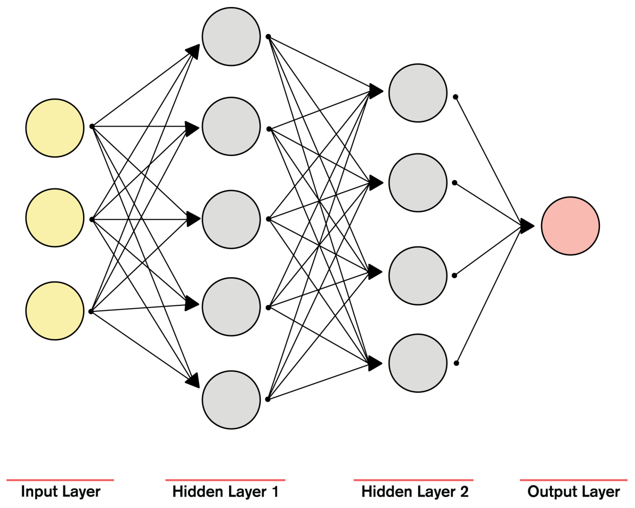


Figure 2.11: Feed-Forward ANN with two hidden layers

Different types of ANNs can be grouped into two main categories: feed-forward networks (in which graphs have no loops) and recurrent networks (in which loops occur based on the established connections) [36].

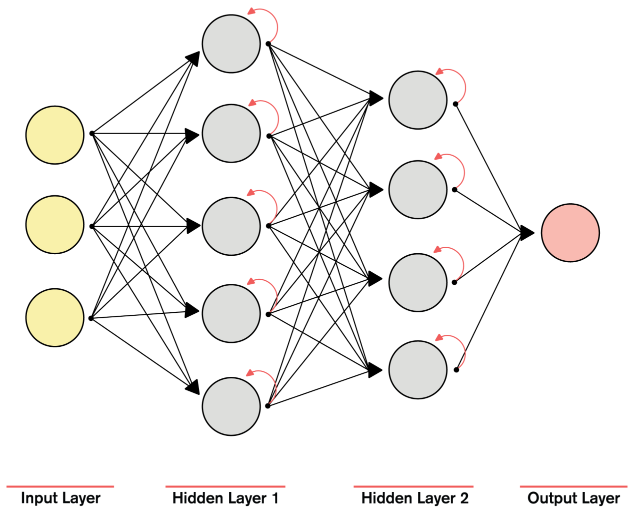


Figure 2.12: Recurrent Neural Network with two hidden layers

These different architectures imply distinct behaviours. As for the Feed Forward Network (FFN), its operations can be defined as static in the sense that they produce only a set of output values and not a sequence. This also means that these networks do not keep any kind of memory of the previous states, hence each output is thoroughly independent. On the other hand, Recurrent Neural Networks (RNN) can be taken as dynamic systems, as their internal state is recurrently updated as new outputs are generated [17]. For the present work, we'll be implementing an RNN architecture for it is crucial to preserve information of previous states and outputs.

As described by Anil K Jain et al. (1996), a unified definition of learning is difficult to formulate. Either way, the learning process of an ANN relies on the problem of updating the network's architecture - especially the connection weights - improving the output for a specific task [36]. A good example of the learning process in an ANN is the back-propagation algorithm. This algorithm changes the connection weights depending on the net error, which can be taken as the difference between the expected output and the actual one of the network. The weights are then changed from the connections closer to the output to the ones right after the *input* layer [47]. In formal terms, the back-propagation algorithm computes the gradient of an objective function with respect to the weights of each layer of the network - which is basically a practical application of the chain rule for derivatives [40]. A technical and mathematical description of back-propagation can be found in Yann Le Cun's *"A Theoretical Framework for Back-Propagation"* (1988) [22].

In a final note, we'll take a broad overview of the most common activation functions regarding ANNs. Knowing that an artificial neuron calculates the weighted sum of its inputs, is easy to notice that the result of this calculation is not bounded in any way (i.e. it ranges from minus infinity to plus infinity). Back to the analogy of the human neuron, we recall that the biological neuron may or not *fire*. To control this an AF is needed, either to bound the neuron's output and to decide whether it *fires* or not (i.e. the output value) [62]. We then have different types of AF, the most common ones described in Figure 2.13

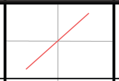
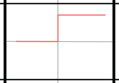
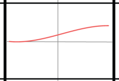
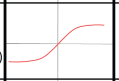
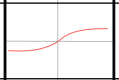
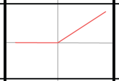
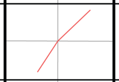
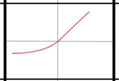
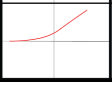
Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary Step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (or Soft Step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh (hyperbolic tan)		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLu)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parametric Rectified Linear Unit (PReLu)		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x)\alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

Figure 2.13: Most Common Activation Functions. Adapted from Sagar Sharma [54]

These functions map the calculated weighted sum of its inputs (x) to a new value according to the function being used. This normalized value will then proceed to other neurons as a new input. Reasonably, different AFs imply different output results - or, at least, different measurements - and although some seem to be more effective than others, it always comes down to the context [62].

2.2.1 Recurrent Neural Networks

RNNs are dynamic systems - meaning that its internal state changes over time on the course of multiple iterations. This dependency on previous states introduces a notion of time to the model making it capable of handling sequences with greater accuracy [42]. In practical terms, the network processes an input sequence (one element at a time), keeping in its hidden units a 'state vector' containing information about the history of all past elements of that same sequence [40]. This ability to memorize events is a key success factor when applying RNNs to tasks such as predicting the next word in a sequence [40] [44].

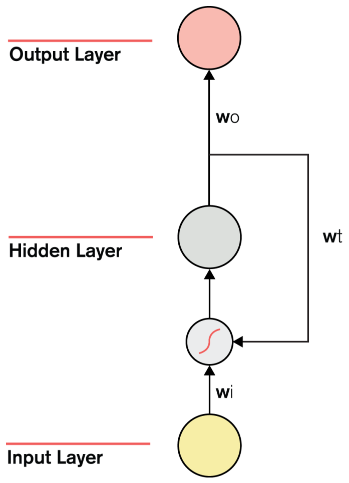


Figure 2.14: Simple RNN. Each transition has a specific associated weight w by which the last output must be multiplied.

The efficiency of RNNs are reiterated by the aptitude of also generating sequences, so that we can have a sequence as an input and get another one as the output. This enables a wide range of possible approaches, like having one input and one output, one input and a sequence of outputs and more combinations of these [38].

To describe the performance mechanism of an RNN, its basic calculations - just as for the FFN - are obtained through feed-forwarding the input value (or values) and not all nodes (i.e. neurons) need to be connected to others or even to itself. Nevertheless, edges that connect adjacent time steps (denominated recurrent edges) form cycles throughout the iteration process preserving information of the current state of that neuron - information

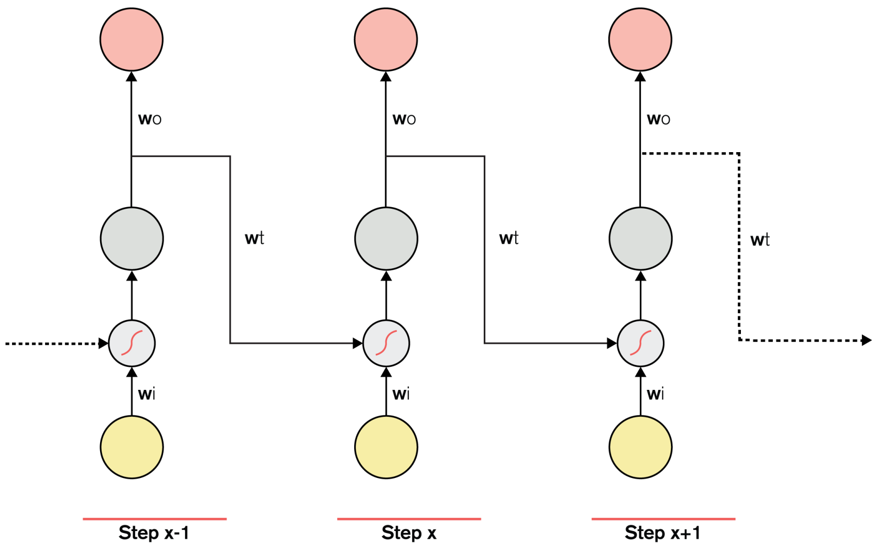


Figure 2.15: RNN through time

that can be used to influence its own next state or other neuron's. For example, given a neuron with a recurrent edge and a time t, it receives information (i.e. input) from the current data X_t and also the previous state data h_{t-1}, with h being the neuron's output after the respective calculations. This simple example illustrates how previous states directly influence the current one by including them in the neurons calculations [42]. Hence, we can understand how these memory restrictions may impact the final output.

Regarding the network training we'll be focusing on gradient-based methods which are the most common ones, although other procedures have been proposed like derivative-free approaches or convex optimization [14] [52]. These gradient-based methods must be deeply related with the model parameters (i.e. weights) and the loss function (which essentially measures the difference between the desired output and the predicted). This relation is essential in the way that based on the measured error (via the loss function) the gradient information is applied to all weights in order to modify them accordingly: a practical application of the back-propagation technique. Through time the defined loss function (ideally) minimizes enabling quantification of the network's performance accuracy, repeating the above-mentioned steps until convergence is reached [14].

Although this seems an almost perfect scenario to handle sequences and state dependencies, RNNs had been found to be difficult to learn to store information on long-term [40][42]. The main problem causing this harness is the nominated vanishing or exploding gradient problem. These two cases occur during back-propagation when the gradients are being scattered all the way to the initial layer. The vanishing gradient problem happens when the gradients have small values and having to go through all the network they diminish exponentially, being almost null by the time they reach initial layers preventing these to learn (i.e. update their values). On the other hand, the exploding gradient problem has the contrary effect, getting larger and eventually *crash* the model as the gradients get multiplied by weights higher than one [42][11].

2.2.2 Long Short-Term Memory

As stated by Bengio et al. (1994), experiments have shown that gradient descent becomes increasingly inefficient when the temporal span of the dependencies increases when training RNNs for long-term dependencies [12], mainly due to the difficulties presented in the last section. We can illustrate this deficiency with an extremely easy example, similar to the one presented by Christopher Olah (*Understanding LSTM Networks*, August 2017) [48]. Given a scenario in which we want to predict the next word in a sentence using an RNN, let's say we have *"Yellow is a colour"* as the input sentence, being the word 'colour' the one we are trying to predict. In this example, we don't need much more information on the context as the sentence as it is is enough to suggest that 'colour' is the upcoming word. In such cases RNNs can easily learn these dependencies where the gap between key-words for contextualization and where it's needed is relatively small. Let's now assume the same situation but this time with a long textual dependency. For instance, the sentence *"I've had Spanish classes and (...) but I can not speak fluent Spanish"*, being the last word the one we want to predict. It's quite likely that the word we are trying to predict is a language, but to know exactly which we need information from the very beginning of the sentence. Contrastingly, in this example the gap is quite bigger and as it gets bigger our RNN is not capable of learning these fundamental connections [48].

One way to overcome this is by augmenting the network including in it a state of explicit memory [40]. Originally proposed by Hochreiter and Schmidhube in 1997 [14] [33] Long Short-Term Memory (LSTM) networks solve this problem [40]. LSTMs are a special kind of RNNs capable of learning long-term dependencies, explicitly designed to avoid this issue [48]. This is possible once these networks use a special hidden unit (a memory cell) that serves as an accumulator in the sense that it has a connection to itself. This means that it copies its own real-valued state and accumulates the external signal, but another important factor is that this self-connection is multiplicatively gated by another unit that learns to decide when to erase (or forget) the content of that memory [40]. This improves the network's performance in accurately modelling both short and long-term data dependencies and also attempts to solve the mentioned vanishing gradient problem by not imposing any bias towards recent observations [14] - this can be achieved due to the fact that each memory cell contains a node with a self-connected recurrent edge of a fixed weight one (which means that every value processed through it stays exactly the same), ensuring that the gradient can pass across many steps without vanishing or exploding [42].

Figure 2.16 illustrates an LSTM module. To understand how each module operates, we'll briefly describe each of its components based on Christopher Olah's walkthrough [48] which depicts these units in a quite straightforward fashion.

In Figure 2.17 we can see the first step of the process (assuming that this is not the first iteration, meaning that we have some information from before to work on), called the

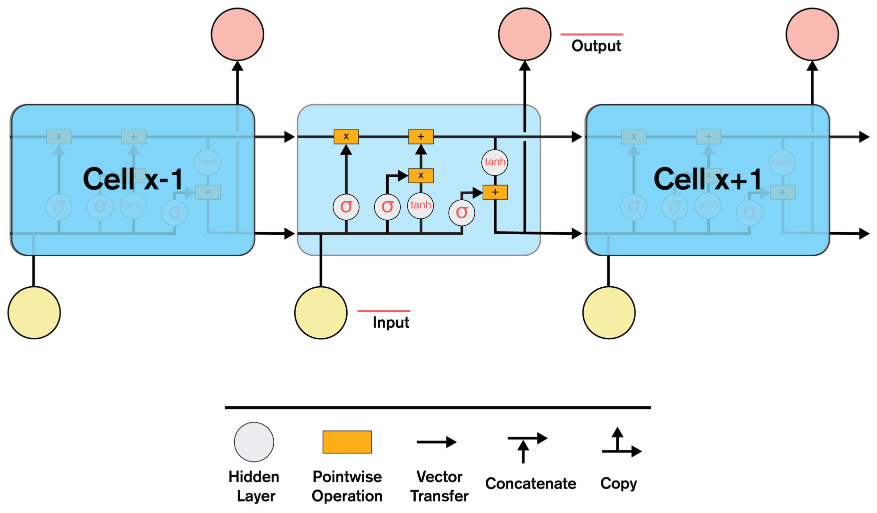


Figure 2.16: LSTM Unit (or cell) through time. Adapted from: Christopher Olah [48]

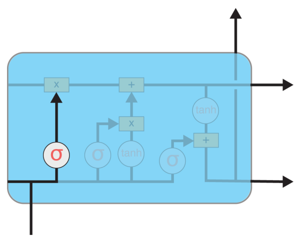


Figure 2.17: LSTM Flow - 1. Adapted from: Christopher Olah [48]

forget layer. Here is where it's decided the past information to be deleted. That decision is made by submitting the values to a *sigmoid* function which outputs values between 0 and 1 (zero means forgetting and 1 can be seen as keeping it all).

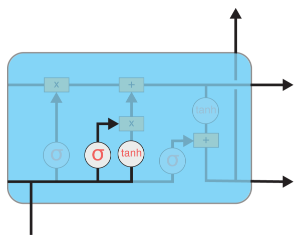


Figure 2.18: LSTM Flow - 2. Adapted from: Christopher Olah [48]

Next, the exact opposite is calculated (i.e. information to keep). This process is divided into two parts. First, we have another *sigmoid* function but this time to decide which values to update. Simultaneously, a hyperbolic tangent function *tanh* is applied to obtain a vector of new candidates to be added to the state. Both vectors are then combined (i.e. multiplied) to generate the state update vector, which consists of the new candidates scaled by how much was decided to update each state value.

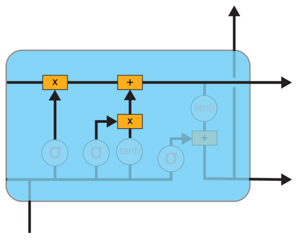


Figure 2.19: LSTM Flow - 3. Adapted from: Christopher Olah [48]

After the last two calculations, resulting each in different vectors accordingly, the *old* cell state is then updated by applying these vectors to it. The *old* cell state is multiplied by the first vector in order to obtain a new one leaving behind the values we previously decided to forget. Then, we add the state update vector acquiring new information.

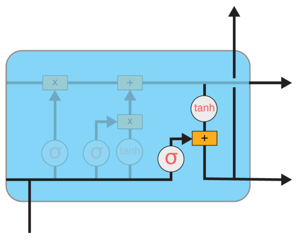


Figure 2.20: LSTM Flow - 4. Adapted from: Christopher Olah [48]

Finally, all this gathered and updated information is carried to the final gate in which it is decided what information should be passed to the output. It will be a filtered version of our cell state (the updated one). The *sigmoid* layer determines which cell state's elements should be part of the output. The newly updated cell state goes through another *tanh* function to bound the values (from -1 to 1). After such, these two last vectors are multiplied being the final vector another filtered version. This is our output, which will also be used in the next iteration as an input (previous state).

It's fundamental to keep in mind that these presented steps are a minimalist way of describing the process. Each represented gate has its own set of weights that control the inputs of each phase and layers. Also, the architecture shown above is the most commonly used one, meaning that throughout the years some variations have been proposed [48].

The effectiveness of LSTMs on handling sequences when compared to usual RNNs was one of the main reasons that led us to follow this architecture as an initial approach, as music composition is highly correlated with sequences. Furthermore, another important aspect that took a major role in this decision is the set of conclusions taken from related work on this field. Next chapter will cover this discussion.

Chapter 3

Related Work

3.1 Music Generation Algorithms

In the introduction of this document, it was stated that music generation via Deep Learning (DL) algorithms has been gaining popularity in the last years. However, it still is an under-explored domain in its plenitude [16] being this one strong reason that led us to direct our focus to this particular field. Regarding the primarily established aim of this thesis, another important aspect of this type of artefacts is its relevance while emotional channel and conceivable mood regulator [32].

We also pointed out the good performance of ANNs, more specifically RNNs and subsequently LSTMs as a starting point for this work. Jean-Pierre Briot et al. (2017) [16] explore in a single book a wide analysis of different approaches to this problem in which in the spectrum of ANNs other types of networks are used besides RNNs. Some of these approaches used Convolutional Neural Networks (CNN) architectures that are usually applied to problems of graphical representation (i.e. images) once this type of architecture is mainly conceived to learn a hierarchy of features that can be used for classification. This hierarchy is obtained by successively convolving the input with learned filters [58]. This means that the network doesn't learn from each individual element but instead from an area (convolution) formed by nearby elements [16]. However, the verified success on this last field (images) didn't seem to last once applied to music generation [16]. As the authors explain, this lack of success might be explained by the fact that images have in principle invariant properties and features, while musical dimensions regularly assume the contrary (e.g. note pitches aren't metrically invariant) [16].

Another reviewed approach was conducted using Generative Adversarial Networks (GAN), an architecture that is also based in convolutional models. The basic conceptual idea behind this particular type of ANN is to simultaneously train two networks, one that generates samples and other that estimates the probability of those sample being either from real data or from the first network [16]. Having this, the main goal is to train the first network to a level that its generated samples can't be distinguished from real-world data. Looking at this architecture when applied to music generation we note that the generated compositions (i.e. note sequences) are overly fragmented when compared to other methods, also lacking musical aptitude [24]. Hao-Wen Dong et al. (2018) used a similar model to create multi-track compositions (i.e. more than one instrument), concluding that the obtained results cannot be compared to human compositions either at a musical or aesthetic level [23].

Andrej Karpathy reiterates the effectiveness of LSTMs units, exploring the application of this architecture in text generation in different contexts provided by various sources for training - such as Shakespeare or Wikipedia entries -, obtaining coherent and detailed texts [38]. This capability of keeping in memory various sequential states - and acting upon them - produce consistent effects when trained with sequences of notes, producing decent rhythms and showing the ability to create small note sequences in a melodic and harmonic manner [39].

3.2 Music Representation Forms

Although different DL architectures naturally assume an important role while studying different practical approaches regarding artificial music generation - or any other field -, deciding how to represent data is crucial to the classification process and desired output [64]. The importance of data representation in DL is transversal to all its domains and branches, providing a simpler or manageable way to extract useful information from training data [64], being artificial music generation no exception when it comes to this topic.

On the same book mentioned earlier, Jean-Pierre Briot et al. (2017) [16] list some possible and used data representations considering its nature. The authors divide the representations into three dimensions (or stages): training input, generating input and generated output. We consider that this separation is helpful in order to understand how these three levels - although necessarily related - assume important individual characteristics on their own. The training input (i.e. the dataset provided for the training) may differ from the generated input - for instance, one could use a training dataset consisting of musical compositions composed of several instruments but use only the piano melodies as the generated input. The same logic can be applied to the generated output, depending on what we are willing to generate. In some cases the reverse can happen, that is all the mentioned dimensions may have the same representation - like using a training set consisting of melodies and use them directly as generating input so the algorithm can learn how to generate its own melodies (generated output).

One possible way of representing a music composition is by using its audio signal [16], which can be obtained via raw audio signals or its audio spectrum (through a Fourier transform) [51]. Another potential method that can be used is by describing the data through symbolic representation - a method used in most of the experiments regarding artificial music generation using DL [16] and that can assume different forms. One is by using data encoded in MIDI (Musical Instrument Digital Interface) format, which essentially are files that comprise event messages regarding a certain music composition. These events hold information about each played note, also defining when one begins and must end, sequencing all existing notes that compose the piece.

Piano roll representation is another alternative and the most frequent type of representation used [16]. In this case, notes are represented through time regarding its presence on the roll. This can be seen as a vertical axis consisting of all possible notes (all the keys in a piano, for instance) where the horizontal axis serves as a time reference, creating a matrix where the note to be played is represented by its corresponding value on the vertical axis and its duration by its length along the horizontal axis. When comparing this last representational method to MIDI representation it evidences some limitations, especially by not having any information on where a specific note should end - which makes a long note indistinguishable from two short ones [16]. Another way of representing musical events is by processing it as pure text. This is done by addressing letters to all distinct notes and appending to it other letters (or numbers) that represent a piece of specific information about it (like the relative octave or duration). Finally, a music piece can be represented by a lead sheet. A lead sheet is another way to describe (or write) musical notation specifying its essential elements. Although it is a well and commonly known form of representation, not many artificial systems have used this type of notation [16].

3.3 LSTM in Music Generation

Keeping in mind the idea that a music composition can be seen in its essence as a sequence of notes, it is clear to conclude that the quality characteristics of a composition (i.e. if it sounds pleasing or not) is necessarily dependent on the notes correlation (or conjugation). This thought reinforces the fact that to study how ANN can be used to generate musical content it is fundamental for the network to have a perception regarding past events (i.e. memory). As discussed in the beginning of the present chapter, LSTM showed to perform efficiently when it comes to past event awareness - especially in text and natural language processing. These insights motivated us to search deeper for recent research and experiments regarding the use of LSTM to generate musical compositions that might come out as fruitful to our work.

3.3.1 Research and Experiments

In 2008, Douglas Eck et al. [26] introduced a *"music-specific sequence learner"* using an LSTM architecture. As explained by the authors, this specific type of architecture was chosen based on the need of dealing with long-term sequences once *"LSTM's architecture is designed to allow errors to flow backwards in time without degradation"* - a consequence of the absence of the vanishing gradient problem detailed earlier (section 2.2.1 Recurrent Neural Networks). The network was trained using a MIDI dataset in order to generate music. As the authors clarify, although the model's outputs are artificial generated musical compositions, the main focus of this experiment was not to evaluate - and eventually improve - the quality of the music but understanding whether the model showed the capability to learn global stylistic constraints [26]. The results were reported as good ones in the sense that the model learned musical structures and long-timescale dependencies in a time series. As this research was conducted mainly to get a better understanding of how LSTM units would perform in this field, this specific model doesn't treat performed music [26].

A few years later, Aran Nayebi et. al (2015) compared the performance of LSTM and Gated Recurrent Units (GRU) - an architecture similar to LSTM units except it has only two gates (reset and update gates) whereas LSTMs have three (input, output and forget gates) [20] - regarding music generation [46]. With this specific experiment, the authors aimed to produce compositions that sounded unique and musically coherent and also analyze through comparison the performance of both architectures. In addition, this test was conducted using raw audio waveforms as an input (i.e. training data), consisting of rock and electronic dance music. As the authors reflect, the pieces generated by the LSTM architecture *"were significantly more musically plausible than those of the GRU"*.

Another experiment strengthens the LSTM's capability of learning sequences of musical events - this time with training data represented as text [19]. On their research paper "Text-based LSTM networks for Automatic Music Composition", Keunwoo Choi et. al (2016) present a model consisting of a text-based LSTM designed to learn relational patterns in text documents representing chord progressions and drum tracks - considering the scope of our work, we analyzed the chord progression results. The study was also set to compare character-based RNN (which predicts a letter corresponding to a single note, for example) and word-based RNN (predicting a vector representing a chord), both containing LSTM units. An interesting point of this experiment is the assumption taken by the authors, who assume that *"there is no constraint on the form of the text representation of music"* in order to observe if the network is able to learn musical patterns and structures based on such a *"weak assumption"* [19]. As a global view of the gathered results, the authors conclude that both architectures provided well-structured results, detailing that the networks learned *"the local structures of chords and bars after a sufficient number of iterations"*.

In 2017 Feynman Liang et. al. presented the "BachBot", an end-to-end automatic composition system designed not only to compose but also to complete musical compositions in the style of Johann Bach (chorales) using LSTM. The training set was composed by Bach chorales in MusicXML - a file format for representing musical notation (as digital scores). The music scores were encoded into sequences of tokens limiting the symbolic representation to pitch and rhythm - all representing polyphonic scores [41]. To test BachBot's generated pieces success in a measurable way the authors developed a publicly accessible musical discrimination test provided online. In these tests the participants were presented with five tasks, each composed by two audio tracks - one originally from Bach and another one produced (or synthesized) by the BachBot model. The results gathered after the tests phase showed that the participants distinguished BachBot creations from Bach ones only 51% of the times, which the authors consider being a suggestion that *"BachBot successfully composes and completes music that cannot be distinguished from Bach significantly above the chance level"* [41].

Nikhil Kotecha et. al (2018) have also used an LSTM architecture to generate music and have interestingly provided a qualitative analysis of the network's performance [39]. Observing the report on this topic, the authors detail that the samples produced by the network were good at creating small melody parts and simple harmonies [39]. Also on this qualitative analysis regarding the training sessions, it is explained that when trained for 30 minutes the network generates *"sparse and significantly less consistent and coherent"* musical compositions. In contrast, the authors claim that when trained for 2 hours the observed difference in the results was positively *"dramatic"* [39], which could illustrate the importance of sufficient training time (naturally dependent of several factors besides the type of network itself). On a final discussion note, the authors also suggest that the network provided better results more quickly when trained with a set consisting of 22 Bach Fugues than when trained with a 120 Piano-de-MIDI set, finally stating that *"it became evident that very long training times were required"* in order to produce *"decent music"* [39].

Although all the above-mentioned experiments and research support the idea that LSTM architectures can learn to generate coherent melodies and especially learn some musical structures and correlations, our present work is also focused on the generation of expressive music and also on how it may influence the perception - or even quality - of a music composition. On January of the present year, Filippo Carnovalini and Antonio Rodà pointed out the importance of expressiveness in a music piece - considering its performance [18]. Regarding expressive performance in computer-generated music, the authors argue that *"the aspect of expressive performance is often overlooked by researchers in algorithmic composition"*, once that most of the times these type of algorithms only learn how to generate the pitch and the quantized duration of notes, regardless of some possible dynamics [18]. The aspect of music expression and dynamics in a composition are crucial elements to avoid a song sounding 'mechanical' when performed by a computer, a point where we agree with the authors when they explain that *"if we want the result to be as musical as possible* *the expression aspects cannot be overlooked"*. Considering this concept of how important expressiveness can be for a music composition, in the next section we'll look deeper into some algorithms provided by Project Magenta which we believe that not only have been a great contribution in artificial content generation but also explore some elements relevant for our present work.

3.3.2 Magenta Project

Just as we have mentioned before, Google Brains' Project Magenta [1] has also used RNNs to generate musical compositions - mainly LSTMs. Among all of the provided projects concerning the study on how machine learning can be used to create artistic content - Magenta doesn't only focus on musical matters but also image, as for instance Image Stylization and others [9] - there are some models that focus only on musical content generation. Some of them focus on musical interactive systems with an underlying ANN designed to promote human-computer interactions, but others were conceived only to generate musical content. An example of this is the Melody RNN, a model that applies language modelling to melody generation using an LSTM in order to create a melodic sequence based on the training set used for learning [4]. Another example of a music generation model is the Polyphony RNN - a model that again applies language modelling but in this case to generate polyphonic music (also using LSTM units) [6]. Instead of being designed to generate only note sequences (i.e. single melodies), Polyphony RNN was conceived to be capable of generating and modelling multiple simultaneous notes - curiously, this model was inspired by the BachBot model mentioned in the last section in the way that polyphony is modelled [6]. Nevertheless, Magenta's Performance RNN [5] is the model that appears to be closer to our present scope and purposes.

Performance RNN model is also based on an LSTM architecture designed to generate music with expressive timing and dynamics [10]. As detailed by the authors, this model was thought considering the essential role dynamics and expressiveness play in music [10]. The Performance RNN was trained with MIDI files of live piano recordings, having a vocabulary consisting of some MIDI events: two separated events that specify when each note should be played and when it must stop; time-shift events which specify when to move to a different note; and velocity events that hold numerical information regarding how hard a note must be played (i.e. intensity) [10]. The results are presented as excerpts around 30 seconds each and described by the authors as lacking overall coherence but still *"quite expressive"*. This type of network and model are - to our knowledge - the closest to our intentions in the sense that it takes into consideration the expressive factor of a music composition.

Based on this analysis, we decided to explore RNNs' capability - concretely LSTM units - of generating musical compositions that could be considered good pieces regarding not only their melodic sequence but also their expressiveness. As we have seen, this type of

ANN has shown promising results in this field, although to our knowledge not yet explored - and especially tested - to the point we're aiming to.

3.4 Main Shortcomings

Having defined the architecture to be used, some research was made in order to find similar existing models that could be useful to our work. Project Magenta has diverse architectures expressly conceived to generate artificial artistic content - from music to paintings. The given models by Magenta are open source meaning that anyone can use and train them [9]. Although adopting these already implemented networks would mean significant progress to our development phase, these algorithms exhibit some limitations: These networks can't be changed in any way - or if they can, then there's no documentation stating this in a clear way. Also, and although Magenta has various different projects, only a few focus on music generation - only one studying expressiveness. Quoting Magenta's developers on the performance of this particular network (the aforementioned Performance RNN) we observe that *"the performances generated by the model lack the overall coherence that one might expect from a piano composition; in musical jargon, it might sound like the model is "noodling"— playing without a long-term structure"*. We consider that having total control over an ANN is not only important to have the possibility of controlling it throughout all its parameters but also to get a good understanding of all the procedures. That being said, we decided to implement our own network to avoid these possible issues.

Evaluating a music composition might be difficult and abstract. George Papadopoulos et al. (1999) exposed two main issues regarding the evaluation of compositions by artificial algorithms. The first one is related to the quality of the output and the agents involved in this examination, where the authors identify the absence of real experts (i.e. professional musicians) in this task [49]. Another aspect reported as a flaw is that for systems that only generate melodies (i.e. sequences of notes) these are easier to accept as good as they don't assume any harmonic context. Quoting the authors, *"most melodies will sound acceptable in some context or other"*. Furthermore, the authors recognize the difficulty of these systems to work with concepts of creativity, mainly because we don't seem to have a clear or consensual idea of creativity itself [49] [15].

With this, it became clear that expressiveness in artificial music is not yet explored as many other fields, especially when it comes to its influence on a music composition which to our knowledge hasn't yet been tested. These main shortcomings and limitations led us to choose the defined approach (concerning the chosen LSTM architecture) and preliminary experiments gave us a more practical and closer overview of the main issues observed on musical compositions generated by our LSTM that we now list and describe below. A more detailed explanation of these first tests and all the details regarding its practical results will be described in the next chapter (4 Preliminary Work).

- **Velocity**: The velocity associated with a note defines how it is played (i.e. intensity). We consider this element to have influence on the expressiveness of the music composition.

- **Duration**: The duration of a note or a set of notes (i.e. a chord) defines for how long should it sound (as stated in section 2.1.2). The variation of this element throughout the music composition might be important to mind as it adds variety, preventing all notes to last for the exact same time.

- **Pattern Diversity**: The music composition must contain diversity in its patterns (i.e. sequences of notes) so that the music doesn't fall into an ineffective cycle of notes leading to monotony.

- **Musical Stability**: Opposed to the last point, patterns must hold some consistency in a way that the music composition doesn't sound like random notes played with no criteria. In that sense, there must be a commitment between both cases.

- **Measurable Quality**: We consider important to have a validation process and not letting the quality of the artificially generated pieces be evaluated only by the members of the present work. Measuring a music composition is a sensible subject, mainly due to its high level of abstraction. But considering that the quality of music is always dependent on the listener, we consider that a user-testing phase might be a good measure to understand its tendencies.

Chapter 4

Preliminary Work

During the first half of this first semester our efforts were focused on testing the neurofeedback device provided - and related research required -, according to what was the firstly defined work plan. As we briefly explained in the first chapter (1 - Introduction) the obtained results led us to reformulate the dissertation scope. This primary research and test phase and resulting decisions can be found in Appendix A (9) where we detail all the work done until this reformulation.

Regarding the latest established goals, the first step was to gather information on the subject not only to acquire a substantially broad background knowledge but also to consider the (possibly) best approach to our problem. That drove us to build an LSTM model in order to have as much control as possible over this architecture.

4.1 First Network

The initial implementation of a basic LSTM designed to learn musical compositions (sequences of notes) was based on the works of Sigurður Skúli [57] and Tianya Mingyue [45], that provided a step-by-step overview of how to apply this architecture to a simple music composition generation. It is important to note that although these similar applications could, in fact, generate music there are some flaws to these models as pointed out by the authors, like the absence of variations regarding notes' duration, their position (offset) and velocity [57].
The network was created and trained using Keras [3], an open source neural network library (in python) that facilitates the network's implementation process.

Our first LSTM layers to be tested was composed of 256 hidden nodes after its *input* layer. These *hidden* nodes define the number of neurons of the LSTM layer. As described before, there is no settled rule that tells us how many neurons one should have as it differs depending on the goal and work. Still, we can state that if we have a higher number of neurons (let's say 512 instead of 256) the network gets more powerful, yet it takes more time to train as its complexity rises - and, as we discuss later in this document, higher complexity is not always desirable.

We then added a *dropout* value of 0.3 (as defined on the given models). *Dropouts* prevent overfitting (i.e. when the generated output is a lot similar to a piece of the training data) by probabilistically excluding a fraction of the layers' connections, which means that for that iteration the randomly selected units don't get updated.

After adding more LSTM layers, we also inserted a *Dense* one to the set of *hidden* layers. Each neuron in a *Dense* layer receives all the outputs from the previous layer, thus it is fully connected.

Figure 4.1 below illustrates the architecture being used in a simple diagram consisting of three LSTM layers, three *Dropout* layers, two *Dense* layers and one *activation* layer.

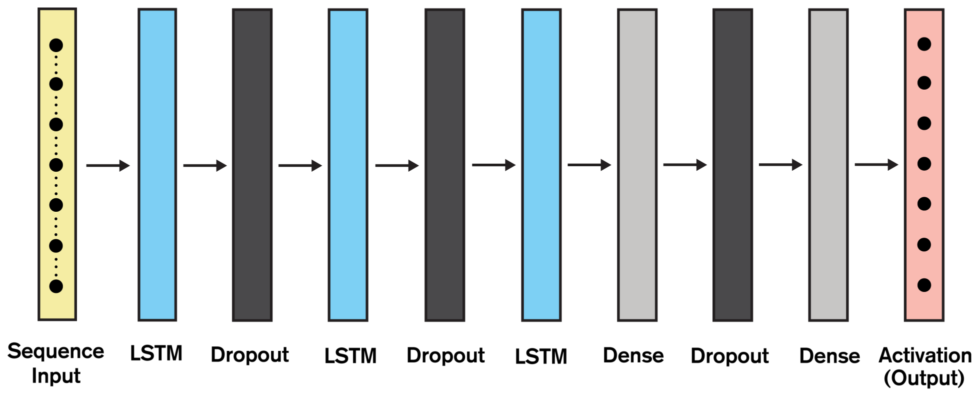


Figure 4.1: First Network Layer Architecture

It is important to mention that the first and last layers (*input* and *output* layers) need more attention when defined once they're highly relevant for each intended work. For instance, the *input* layer must have the same size as the desired input, and the *output* layer must have its size defined according to what we are willing to generate. This means that if we're trying to predict the number after a sequence (e.g. [1,2,3]) then our *input* layer must match this vector size, and as for the *output* layer it must have the same number of neurons as the number of possible integers (e.g. 1 to 10). Again, we reiterate that the number and types of *hidden* layers we are using in this first network were implemented that way in order to observe its performance. These elements can and should be changed to evaluate its impact on the output.

In the *activation* layer a $softmax$ function is used, which assigns a probability to each value (in our case, a note) meaning that for every prediction the value with higher probability assigned to it is going to be the outputted one (i.e. predicted).

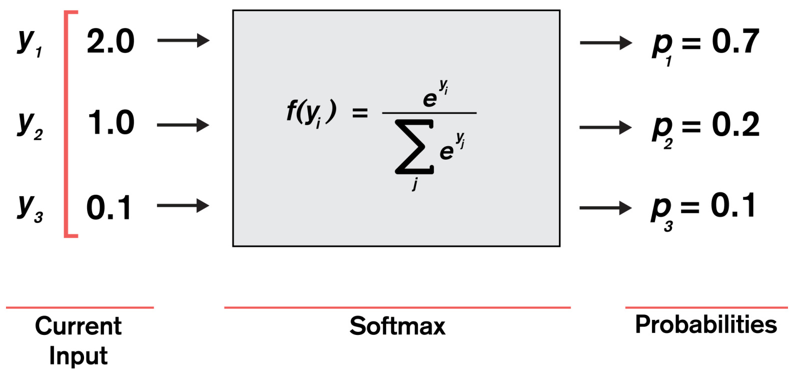


Figure 4.2: Softmax Function Example. Adapted from Uniqtech Co [21]

For the error calculation at each step (i.e. iteration) this architecture applies a *categorical cross−entropy* loss function that measures the performance of a classification model whose output is a probability value between 0 and 1. *Cross − entropy* is useful when a network's output is seen as a representation of independent hypotheses (in our case, different notes) and neuron's activation can be understood as representing the probability (or confidence) that each hypothesis might be true (or correct) [50].

4.2 MIDI

To train the network we needed a dataset - in our case, multiple songs - in order to our model to learn its structures and patterns. As pointed earlier in this document - and stated by Jean-Pierre Briot et al. (2017) -, most of the DL systems used in the field of music generation focus in a symbolic representation [16]. There are multiple ways of getting a symbolic representation of melodies or notes in a certain music composition, like text or piano roll as earlier explained. Another way of achieving this is by using MIDI files, which was our first choice not only due to the great number of MIDI files available online but also considering that we have the available software that allows us to read and play these files, where we can hear the music composition while observing all the existing notes (e.g. GarageBand or MuseScore).

MIDI (Musical Instrument Digital Interface), as described by the last mentioned authors, is *"a technical standard that describes a protocol, a digital interface and connectors for interoperability between various electronic music instruments, software and devices"* [16]. A MIDI file holds event messages of each existing instrument in a song regarding its notes and some of the associated elements like note duration or velocity. For each track of a certain MIDI song (i.e. different instruments) the sequence of notes is gathered into a stream of events containing two main messages that dictate the notes' flow: *Note On* and *Note Off*. The *Note On* event indicates that a note must be played, containing also information that specifies its attributes like the note's pitch (an integer within 0 and 127). For instance, if we'd have a note to be played with a pitch value of 60, then we would have a *C* (Do in Romance languages). The *Note Off* event tells us that that particular note should stop (i.e. be released).

To get this information on the events from the MIDI files we used Musci21 library [8], also written in python, that was mainly conceived to extract all events and related information from a MIDI file by parsing its content. This allows us to build a structured sequence of events in a more simple way, empowering this data to be handled and eventually transpose its representation.

Figure 4.3 shows a simple stream parsing using Music21. Here we can see that chords are represented by all the notes that form them (i.e. notes played simultaneously).

```
  <music21.note.Note F#>
   <music21.note.Note D>
   <music21.note.Note D>
<music21.chord.Chord C2 F#2>
```

Figure 4.3: Music21 MIDI file parsing output (example).

In the last line presented on this output example, we can see that both *C* and *F#* - the hashtag symbol stands for Sharp (section 2.1.2) - notes are followed by the number 2. This number identifies the octave. As we recall from Chapter 2, notes are identified by their pitches leaving us with twelve possible notes (in a standard piano or keyboard) that are repeated throughout higher or lower octaves. As pointed earlier, MIDI encodes the notes within a range [0 to 127]. This maps all possible note to an octave. Figure 4.4 below illustrates this relation matrix.

Octave	**Note Numbers**											
	C	**C#**	**D**	**D#**	**E**	**F**	**F#**	**G**	**G#**	**A**	**A#**	**B**
-5	0	1	2	3	4	5	6	7	8	9	10	11
-4	12	13	14	15	16	17	18	19	20	21	22	23
-3	24	25	26	27	28	29	30	31	32	33	34	35
-2	36	37	38	39	40	41	42	43	44	45	46	47
-1	48	49	50	51	52	53	54	55	56	57	58	59
0	60	61	62	63	64	65	66	67	68	69	70	71
1	72	73	74	75	76	77	78	79	80	81	82	83
2	84	85	86	87	88	89	90	91	92	93	94	95
3	96	97	98	99	100	101	102	103	104	105	106	107
4	108	109	110	111	112	113	114	115	116	117	118	119
5	120	121	122	123	124	125	126	127				

Figure 4.4: MIDI Note to Pitch Table. Based on Tonalsoft [61] and adapted from Tianya Mingyue [45]

4.3 Process

4.3.1 Representation

After gathering all notes from the MIDI files training set we needed to find a representation of these notes that were compatible with our network. As described in the last section, every note in a MIDI file is represented by its pitch and all these sequences are kept in a list after converting each note to a string element. Therefore, we created a new list that contains every note (or chord) existing in the set, excluding repetitions later. This told us how many different elements exist in our training set, which we then used to define our vocabulary. Afterwards, a dictionary is created to map every unique note to a unique number in order to map the sequences.

We illustrate this representation phase with a brief and simple example. Let's say that our training set, after extracting all notes, consists of the following sequence of notes:

$$S_1 = [C, C, B, F, F, A, G, C, E, A, A]$$

We now create a new list (S_2) containing every unique note:

$$S_2 = [C, B, F, A, G, E]$$

S_2 is our vocabulary, meaning that we have six different possible notes. The next step is to map each note to a unique value so that every note can be represented by an integer. We do that by creating a dictionary (D_{map}) where C = 0, B = 1, F = 2, etc.:

$$D_{map} = [0, 1, 2, 3, 4, 5]$$

After that, we can look back to our initial list (S_1) and map the sequence to a new list (let's call it S_{map}) according to each note value:

$$S_{map} = [0, 0, 1, 2, 2, 3, 4, 0, 5, 3, 3]$$

The sequences must be reshaped and normalized to match the network's input format (this required input shape is detailed in Keras' Documentation on LSTM models [3]).

4.3.2 Training

We now detail the training phase. As described earlier, the LSTM is prepared to receive sequences as an input (i.e. its training data) and for each sequence we must declare the desired output so that we can measure the error in order for our model to train. For that, we needed to define the length of the training sequences. In our model, we defined a length of 100 notes but just like other elements and parameters this value can and must be changed in order to evaluate its impact on the output and performance.

To illustrate how the training phase works, let's assume the same data used in the example in the last section. To simplify, we'll be using a sequence length of 3 notes.

$$S_1 = [C, C, B, F, F, A, G, C, E, A, A]$$
$$S_{map} = [0, 0, 1, 2, 2, 3, 4, 0, 5, 3, 3]$$

S_1 is a list with a sequence of all notes in a given dataset, being S_{map} the corresponding integer values. Having a sequence length of 3 to give as input for training, we iteratively generate groups (lists) of three notes and provide the expected note to be outputted.

$$Seq_1 = [C, C, B] \rightarrow ExpOut_1 = [F]$$
$$Seq_2 = [C, B, F] \rightarrow ExpOut_2 = [F]$$
$$Seq_3 = [B, F, F] \rightarrow ExpOut_3 = [A]$$
$$Seq_4 = [F, F, A] \rightarrow ExpOut_4 = [G]$$
$$Seq_5 = [F, A, G] \rightarrow ExpOut_5 = [C]$$
$$Seq_6 = [A, G, C] \rightarrow ExpOut_6 = [E]$$
$$Seq_7 = [G, C, E] \rightarrow ExpOut_7 = [A]$$
$$Seq_8 = [C, E, A] \rightarrow ExpOut_8 = [A]$$

As a matter of fact, every sequence and the corresponding output is mapped to its integer value previously defined. So, within this example, the first training sequence would be the following:

$$Seq_1 = [0, 0, 1] \rightarrow ExpOut_1 = [2]$$

While training, the network will progressively learn patterns in the given sequences, adjusting its internal parameters at each step. In section 4.1 we've explained that the last layer (*output* layer) must match the shape (i.e. number) of different classes of the problem. Following this example, our vocabulary tells us how many values we want at the output. So, having S_2 = [C, B, F, A, G, E], which is the set of different notes we have in our training set, the last LSTM layer will assign a probability to each note. For instance, the first training sequence output (desired) was an F (or 2, its integer value). If after this sequence the last layer showed that the note with the greatest probability was an A, then we would measure this error once we wanted an F note to be predicted. This is done throughout each sequence a defined number of steps (i.e. times we go through these sequences), and eventually the network would update its internal values and states until we reach an acceptable minimum error via back-propagation (see section 2.2.1).

4.3.3 Music Generation

After its training phase, the network is ready to generate a music composition. This is done in a quite similar way as the earlier training. We provide another sequence as input (this time, a single one) but instead of evaluating the predicted note we just take the note with the highest assigned probability and insert it in the last index of the sequence. This process is repeated until the list is full with predicted notes. A simple demonstration of this process can be seen in Figure 4.5 and Figure 4.6.

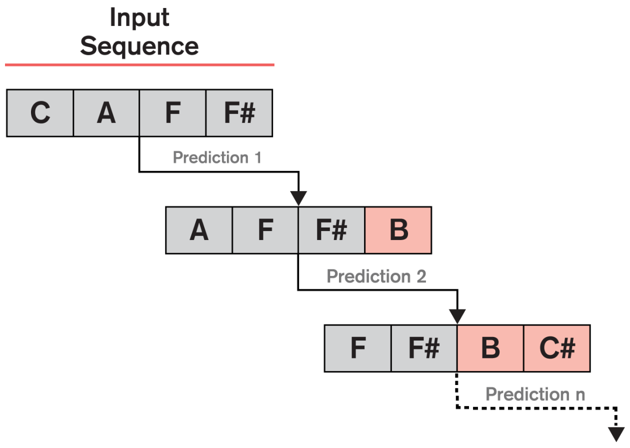


Figure 4.5: Output sequence generation example (for a sequence of size = 4).

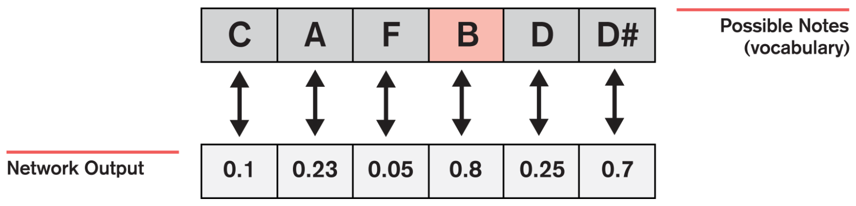


Figure 4.6: Example of a note prediction.

After having our final list with all notes, we then apply the opposite method to decode these values into notes or chords, and we are then able to write it to a new MIDI file using Music21 Toolkit.

4.4 First Results

Before addressing artificially generated expressiveness, we primarily tested our network regarding melody only. These first results served as a primary perception of note sequence generation.

The first training set used was composed of 26 MIDI files (all the files that were available online). All of these files are MIDI versions of *The Strokes'* songs (an American garage rock band). The decision of training the network with this set instead of the usual ones used (like *Bach* compositions) derived from the fact that we are quite familiarized with this band and songs - a lot more than with *Bach*'s or any other classical or baroque piece -, which we considered critical to evaluate the network's performance in an early stage. That way we could tell whether the generated compositions were merely a copy of some sequence provided in the training set or if there were any signs of creativity (note that by 'creativity' we mean the network's ability to produce similar sequences but no exactly the same). In Appendix B (9) we provide a list of all songs used in each training session.

For this first attempt, we cover generated results as the network improves throughout the epochs - each epoch the network trains through the entire dataset - in order to provide a clear view of its evolution on learning how to compose a music piece. Each artificial composition is converted to a MIDI file that was then converted to *mp3* format. For each example we provide a link of the song hosted in SoundCloud, a music streaming platform.

The first music composition was generated only after two epochs of training - with a loss value of 3,4598. Logically, we didn't expect much from it as it hasn't decreased its error in a considerable manner. Still, we used this as a measurable example for comparison purposes on further generations.

As we can see and hear through Figure 4.7 and File 1, the network generates sequences of the very same chord invariably. We've waited for a few more epochs until the loss showed some improvement (i.e. minimization).

After ten generations the network seems to start to recognize the need for variation. These pitch changes are visually and audibly recognized below (Figure 4.8 and File 2). Also, Figure 4.9 shows that the network has learned that shouldn't stick to a single note or chord for the entire composition. Still, this musical composition is looping the same sequence until its end. We've waited until the 30th epoch, where the loss decreased significantly

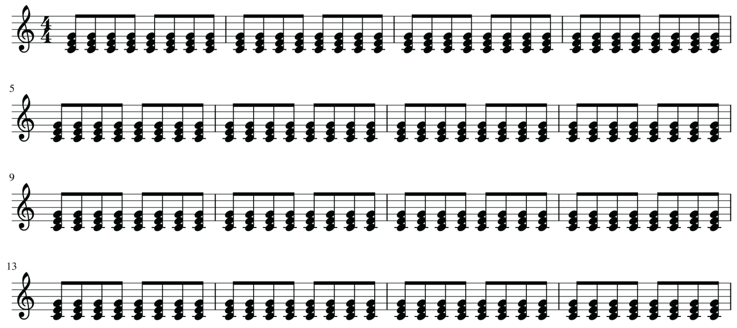


Figure 4.7: Composition after 2 epochs of training (excerpt).

File 1: Composition after 2 epochs (*The Strokes*)
url: https://soundcloud.com/josemariasimoes/file1/s-uaWXr

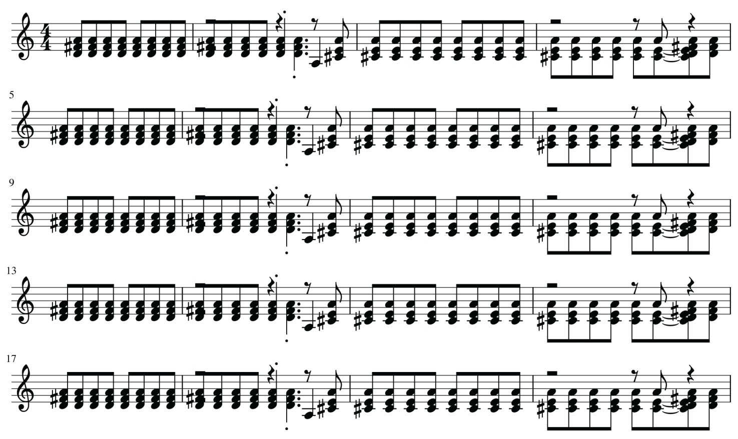


Figure 4.8: Composition after 10 epochs of training (excerpt).

File 2: Composition after 10 epochs (*The Strokes*)
url: https://soundcloud.com/josemariasimoes/file2/s-WJedq

(reaching 0,2166). The generated piece at this stage is provided below.

Looking at the sheet in Figure 4.10, we can see that it is more complex than the last output not only in terms of note variation but also in its capability of playing a wider range of notes (Figure 4.11). We also hear that at the beginning of the song there's a sequence of notes and chords that last for about ten seconds and after that a completely new sequence is heard. This last sequence holds more variations than last output did, but again it falls into a loop until the end. At the 50th training epoch we start to have some interesting results, verifying a loss of 0,0742.

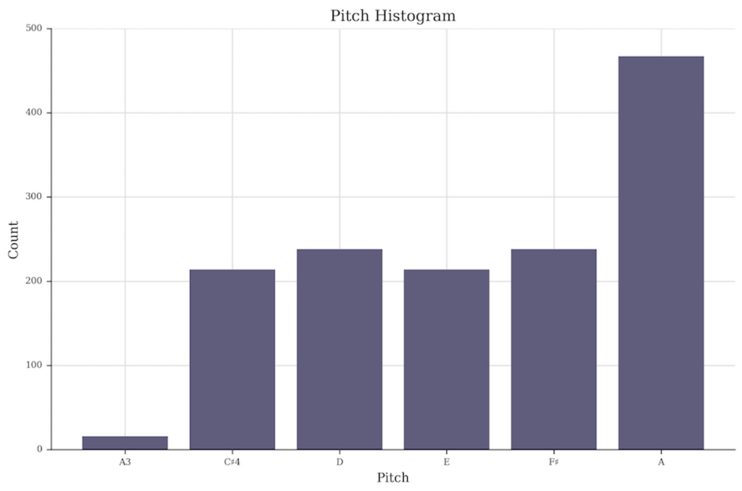


Figure 4.9: Frequency of different notes used in the composition above (File 2).

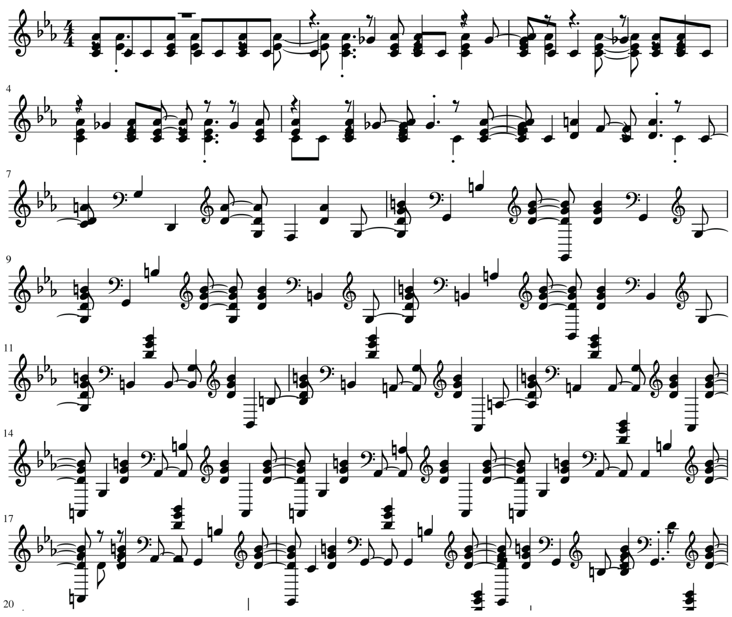


Figure 4.10: Composition after 30 epochs of training (excerpt).

File 3: Composition after 30 epochs (*The Strokes*)

url: https://soundcloud.com/josemariasimoes/file3/s-IvRUw

In this composition we can see that the improvement of the network is mainly aesthetic (i.e. harmony and chosen sequences). The present sheet (Figure 4.12) visually shows that now some low pitch notes are taken into consideration more frequently. Also, there are mere short-term variations in this new composition which suggests that the network is learning some interesting patterns. This can be confirmed by comparing both pitch histograms (Figure 4.11 and Figure 4.13) where we can see that the network is not using more notes but instead is learning how to use and combine them. Another interesting aspect of this composition is that we can observe some sense of musical structure where the composition is no longer using the same sequence to the end: we can hear a sequence pattern during the first third of the song that is then *replaced* by another and a bit before the end the

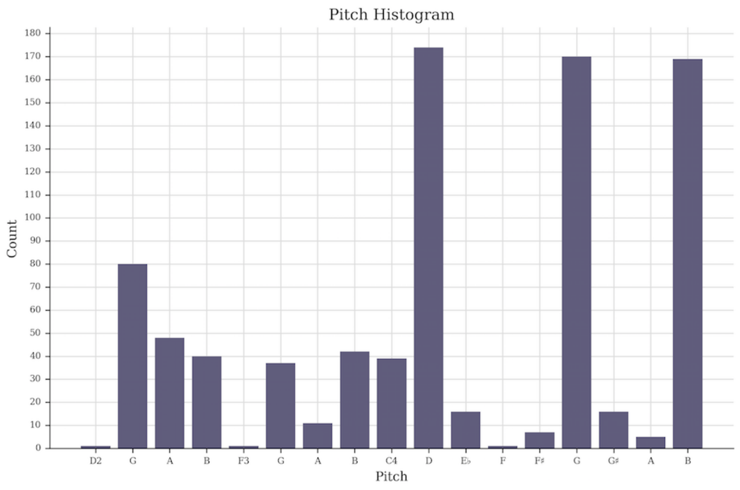


Figure 4.11: Frequency of different notes used in the composition above (File 3).

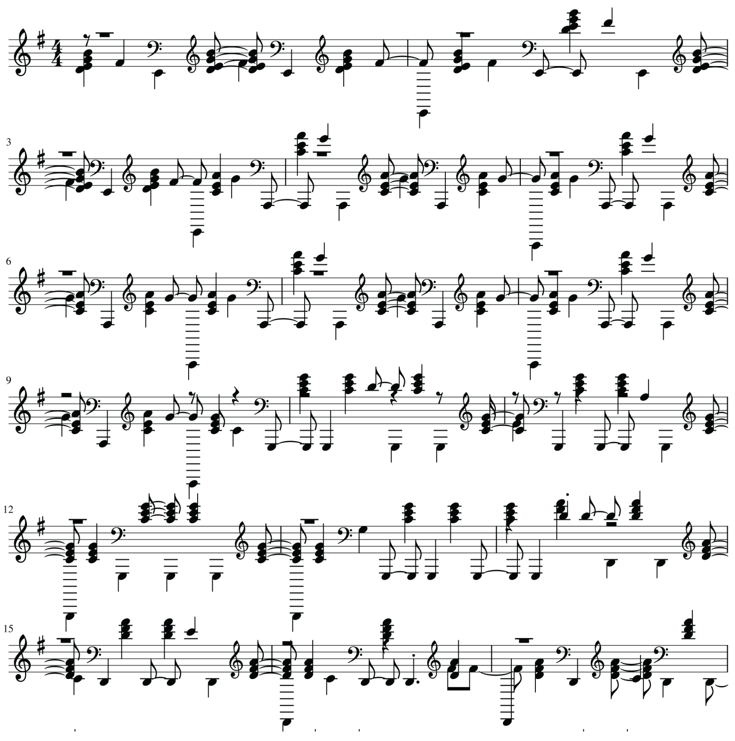


Figure 4.12: Composition after 50 epochs of training (excerpt).

File 4: Composition after 50 epochs (*The Strokes*)

url: https://soundcloud.com/josemariasimoes/file4/s-LC0L7

initial pattern can be heard again giving the idea of a whole or logic to its composition.

The last generated composition is taken from the 100th epoch (the network was set to train during 120 epochs) which was the best epoch with a loss of 0,0556. The results are presented below.

This final composition is interesting in the sense that it has in it three different parts - all

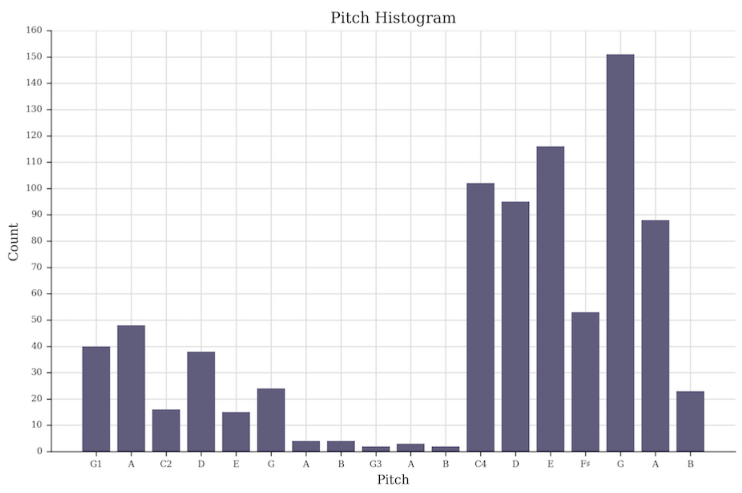


Figure 4.13: Frequency of different notes used in the composition above (File 4).

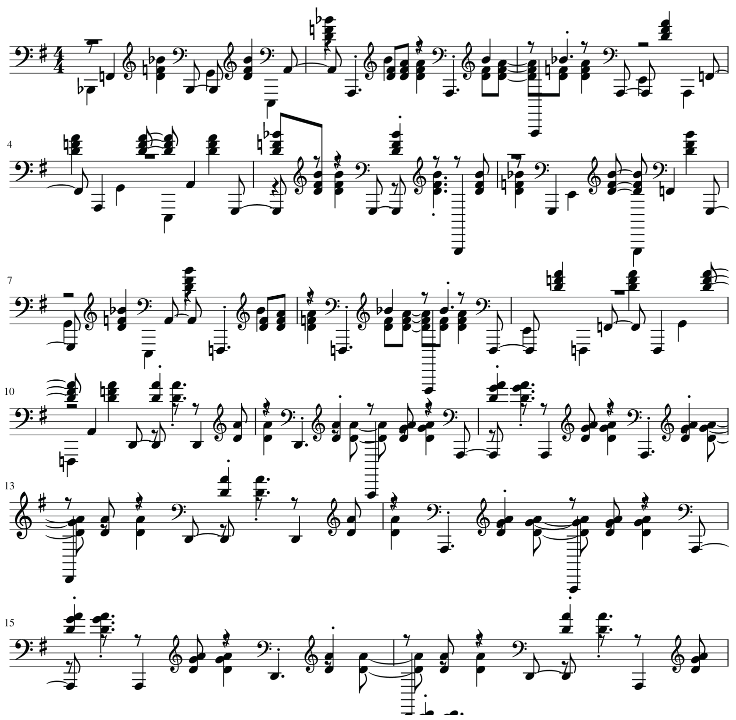


Figure 4.14: Composition after 100 epochs of training (excerpt).

File 5: Composition after 100 epochs (*The Strokes*)

url: https://soundcloud.com/josemariasimoes/file5/s-wXsYV

distinct from the others - but all of them stick to the music's flow, meaning that there's no abrupt change on the key or some pronounced dissonance. Also, in the last part (second half) of the piece we can hear some pitch variations in which the following notes notably follow those changes.

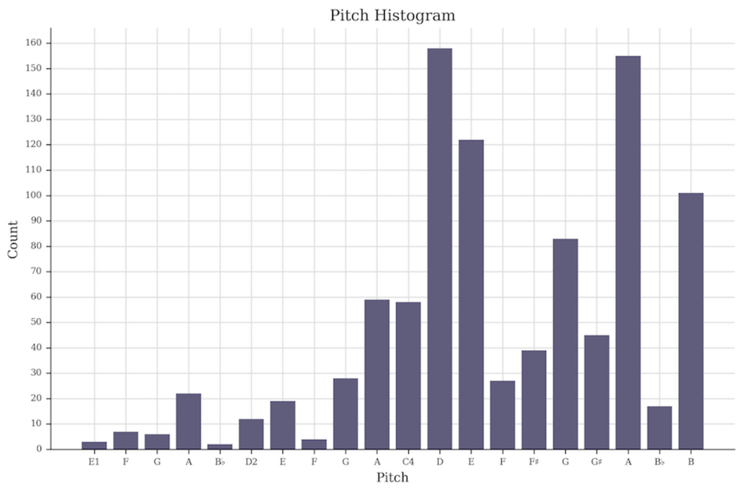


Figure 4.15: Frequency of different notes used in the composition above (File 5).

4.4.1 Observations

These first results were taken as strong evidence that the network learns some basic music rules by itself (considering that the training set follows these rules). In section 2.1.2, we've pointed out that musical compositions must follow some rules regarding the notes that fit in a given key or chord progression. As one might conclude, we didn't explicitly *tell* the network what these rules were in order to avoid complete dissonance or unpleasant compositions, as we considered that once the network is learning from compositions that follow these rules and harmonic laws it would probably learn them via the observed patterns. Not all of the generated compositions were dissonance-free and some notes didn't fit well in a given sequence, but we took that as a training set limitation (that would be proven to be a faulty conclusion as we discussed in section 4.4.3).

Also, this first network proved to be capable of generating different compositions having the same parameters in it. Below, there are two more songs generated by the same network (same weights used to generate this last composition - 100th epoch).

File 6: Another Composition after 100 epochs (*The Strokes*) - 1

url: https://soundcloud.com/josemariasimoes/file6/s-pvOmn

File 7: Another Composition after 100 epochs (*The Strokes*) - 2

url: https://soundcloud.com/josemariasimoes/file7/s-6UwDb

4.4.2 Different Genres

Having presented the initial results in our first attempt, the exact same network was used but with different genres as a training set. The set used before - consisting of *The Strokes'* songs - had little variation in them, as the band usually sticks to a chord (or a progression) for a long time.

The second training set consisted of some songs (MIDI) form *The Beatles.* This decision derived from the fact that these songs have more variations and differ between them more frequently than the last training set. The songs used in this set can be found at Appendix B (9). For this training set we present only pieces generated from the best epoch, as the training process was similar to the last one.

File 8: Composition after 150 epochs (*The Beatles*) - 1
url: https://soundcloud.com/josemariasimoes/file8/s-3mV2T

File 9: Composition after 150 epochs (*The Beatles*) - 2
url: https://soundcloud.com/josemariasimoes/file9/s-aEu3Y

File 10: Composition after 150 epochs (*The Beatles*) - 3
url: https://soundcloud.com/josemariasimoes/file10/s-CjeTi

The first composition shows an interesting initial sequence of notes that might be taken as an intro as it is followed by a pleasant melody until the end of the song without becoming too predictable, yet preserving the same structure and familiar sequences. The most negative aspect of this composition that is in clear need of improvement is that some notes stack and as they are played sequentially, it may sometimes sound repulsive as the keys are played as if the file has crashed.

The second composition provides a clear example of *severe* dissonance. In the last quarter of the composition we can hear notes and chords that just sound as if they don't fit well together. This is a good example of the kind of dissonance that we're willing to avoid.

The last example doesn't seem to follow the same problems as the last one, containing some interesting simplistic melodies with both high an low pitch notes. Again, it gets to a certain point where it seems to be merely repeating the initial sequence.

We've also tested the network using a compilation of jazz songs as training. This didn't reveal many good results as both pieces seem to follow a random flow of notes.

File 11: Composition after 100 epochs (Jazz Various Artists dataset) - 1
url: https://soundcloud.com/josemariasimoes/file11/s-aCPS3

File 12: Composition after 100 epochs (Jazz Various Artists dataset) - 2
url: https://soundcloud.com/josemariasimoes/file12/s-8YBxU

4.4.3 Considerations

Comparing the results from *The Strokes* dataset with *The Beatles* one we can clearly notice that when trained under the latter the network generates more interesting variation patterns, and that's is likely due to the fact that the songs used have more note's variations that *The Strokes*. Still, in both cases, the model showed a good ability to follow a certain key and change-tolerance or resilience in the sense that in most cases when we hear a key change (whether a semi-tone or higher or lower) the sequence following that change seems to be *aware* of that.

Looking at the last compositions - jazz ones - there's no 'logical' pattern or repetition that could give us the sense of a whole. This can have multiple reasons. First, jazz differs from the last two genres in a way that rock/pop songs tend to follow a similar musical structure (verse, chorus, verse, chorus and often a bridge). As one might understand this is not universal, but one can easily agree that when it comes to jazz there's no predefined structure. Another possible explanation for this is the mixture of different authors in the dataset. While in the first two datasets all songs were written by the same band, the jazz dataset was composed of different authors which could difficult the learning task. Allen Huang and Raymond Wu (2016) present similar results when training the network with two different datasets - one solely consisting of *Bach* pieces and other containing classical pieces from different authors -, pointing out that the generated compositions trained on the "*Bach* Only" were *"more aesthetically pleasing than the one trained on a grab bag of different classical pieces"* [34].

Like we have earlier referred, some of the generated compositions contain some severe dissonance, sometimes giving the idea that some notes don't fit in the sequence. This led us to two possible causes: The model isn't capable of fully understanding some harmonic rules or there could be some kind of deficiency in the training set. After some deeper analysis on this, we came to realize that in some files the track used for training contained encoded drum messages. MIDI files hold event messages on the instruments that exist in a specific piece. Each possible instrument is encoded and addressed to a specific track, and percussion events also have a particular encoding [7]. That is because percussion events (like a drum set) aren't seen as notes but rather elements of it. For instance, in a piano each note event holds its pitch that can be mapped to a played note, but in the case a drum kit each event isn't a note but the element of a drum kit to be played at a certain moment. So if we pick a certain drum track and transpose its events to a piano one this will no longer sound as snare or cymbals, but as a pitch. Our first network was conceived to receive as input sequences of notes from a single instrument. While preparing these sequences, the process was the same for each file: We read the file using Music21 library and save the stream (i.e. sequence) of the instrument in the first position. As we later came to realize, some MIDI files encoded the drums or percussion in the first position meaning that those sequences were being used as training sequences, which can disturb the learning process. Having this, and knowing that Music21 instrument parser isn't consistent for what we've experienced, we decided to go through each MIDI file by hand opening each one and register its instruments and channels used.

Based on the results and information gathered, we departed from these preliminary experiments to develop our prototype model.

Chapter 5

Prototype Development

5.1 Expressive Elements

After acknowledging our networks' ability to learn basic musical rules and to follow coherent structures observing the preliminary results, the next step of the development phase was to introduce expressive elements such as note velocity and duration to our prototype. With this, we intended to improve the initial model by building a more dynamic structure (i.e. architecture) capable of learning not only note sequences but also variance patterns concerning musical expression. As described in section 2.1.3, emotion variations induced by a music composition is linked to personal tendencies (or tastes) and cultural background, in specific note sequences (chord progressions, scales). Nevertheless, there is more to it than the notes to be played. Note duration (see section 2.1.2) specifies the amount of time a note must be heard in a given time throughout the composition. Thus, this influences the perception of the note - and even the perception of the entire piece. A note that is played along with other notes but lasts longer will still be heard as the other notes progress through the melody, and a succession of notes with a short duration can easily be perceived as *staccato*, providing a completely different involvement with the piece. Note durations are static in the sense that the staff clearly specifies them (although it is always susceptible to human imprecision). On the other hand, velocity (i.e. how hard a note is hit or attacked) has more to do with the performer himself and although there are some notations that specify certain parts concerning these elements the intensity is always predisposed to variation. Considering the set of notes in a piece and their intensity when performed, it is easy to understand how it also affects the composition and how it is perceived by the listener. In addition, velocity can be used to build the narrative of the piece whether by gently playing the notes or by making them stand out with more aggressiveness.
We departed from this first artificial network to extend our model so it could cover these topics to learn possible patterns via training and eventually generate its own sequences.

5.2 Music Elements Representation

Similarly to what was done in the first network regarding the data collection (notes) from the training set, we have used Music21 library to extract the referred expressive elements from MIDI files. As depicted in Figure 4.3, we obtain all notes encoded in each file by parsing them. Moreover, the library provides methods to access all the registered information embedded in a single note - such as velocity and duration. Figure 5.1 illustrates the output after extracting this data likewise the output from Figure 4.3 but using the provided methods to obtain velocities and durations.

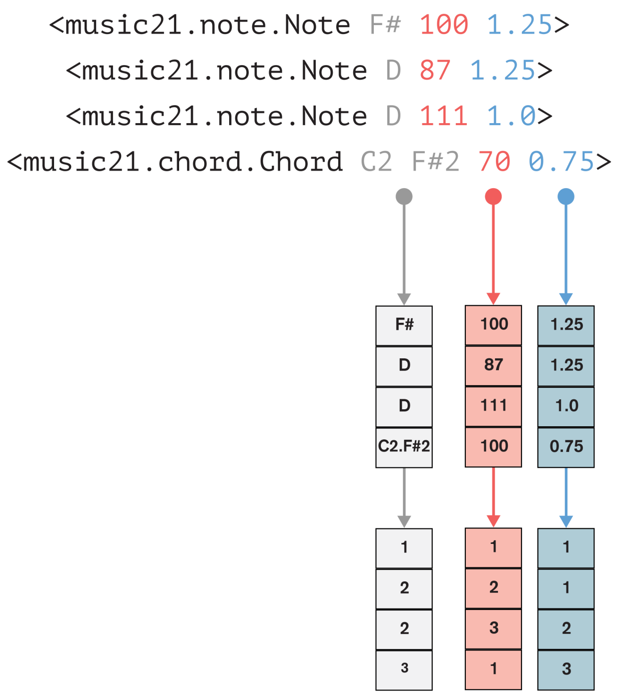
```
<music21.note.Note F# 100 1.25>
<music21.note.Note D 87 1.25>
<music21.note.Note D 111 1.0>
<music21.chord.Chord C2 F#2 70 0.75>
```

Figure 5.1: Music21 MIDI file parsing output - notes, velocities and durations (example)

As described earlier, velocities indicate the intensity of the notes and are represented by integers between 0 and 127 - being 0 inaudible and 127 the loudest value (i.e the most intense). Note durations are represented by floats having the reference value of 1 for a quarter note (see section 2.1.2 - Durations, Figure 2.7). This means that any value below 1.0 is less than a quarter note and above that same value the duration is more than a quarter note. As an example, a duration encoded in the MIDI file with a value of 0.5 corresponds to an eight note, meaning that a whole note has the encoded value of 4.0.

All encoded values (durations and velocities) were kept separately in different lists of values, which means that after parsing all files in the training dataset we have as a result three different lists of extracted elements - Notes, Velocities and Durations.

The representation method used for these new elements was the exact same employed before when dealing with note sequences only (see section 4.3.1 for a detailed description) to prepare the sequences to be in a compatible setup with the one required by the Keras framework. This means that after the extraction phase each set was converted into a new list of unique elements, assigning an integer to each one. Subsequently, a new list was created mirroring the original ones but consisting of its correspondent integer, being this last list the one used to train the network. Figure 5.2 below illustrates this conversion process.
The set separation eased the modelling process as some constraints led us to deal with each element set independently. We address this issue in the next section.



<music21.note.Note F# 100 1.25>
<music21.note.Note D 87 1.25>
<music21.note.Note D 111 1.0>
<music21.chord.Chord C2 F#2 70 0.75>
F#
D
D
C2.F#2
100
87
111
100
1.25
1.25
1.0
0.75
1
2
2
3
1
2
3
1
1
1
2
3

Figure 5.2: Independent representation process (Example)

5.3 Model

Recalling our first model, which consisted of a network designed to receive sequences of integers (notes representation) as an input and then learn its mathematical relations in order to generate its own sequences. This proved to be efficient when dealing solely with notes, but with the inclusion of expressive elements the network would then have to learn possible patterns between three different dimensions (i.e. notes, velocities and durations), with the additional need of handling them simultaneously. Just as expected, this combination of distinct factors proved to be an overload for our model in the sense that it did not show any signs of improvement (i.e. learning, via loss rate estimation) in a trial training session. This constraint can be easily justified by the exponential combinatory possibilities regarding the network's vocabulary, raised by the introduction of durations and velocities to the model. As described earlier in this document, the network's vocabulary is the set of unique values that are recognized by it - as an example, if a music composition is composed by several combinations of four notes, the network's vocabulary would be the set of those same four notes only. This would then mean that taking these new elements into account, a G note that has a duration of a quarter note and is played with a velocity of 50 is seen as different from a G note with a duration of an eighth note played with a velocity of 100 (or even with the same velocity). Figure 5.3 and Figure 5.4 below depict the contrast on identifying possible note patterns on both sequences.

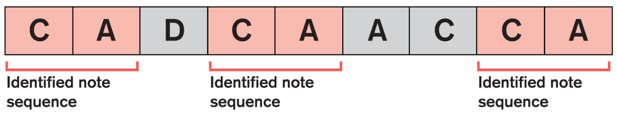


Figure 5.3: Note sequence identification (Notes only)

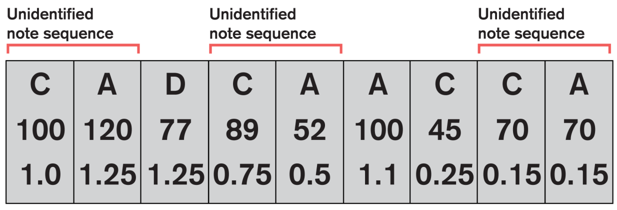


Figure 5.4: Note sequence identification (Notes, Velocities and Durations)

Given the sequence illustrated on Figure 5.3, the network would identify the highlighted note sequences learning that every time a C note is generated there's a high probability of the following note being an A. However, if to each note were appended its velocity and duration values (as a full and unique block), the network would then fail to identify the same $'C \rightarrow A'$ sequence as each C note on the sequence would be different - from Figure 5.4, we see that the first block ($C - 100 - 1.0$) is different from the fourth one ($C - 89 - 0.75$). Therefore, the learning process would be severely affected by this association of terms as some patterns would not be recognized. Inevitably, this shortcoming would not only affect note sequences but also durations and velocities.

The presented issue could theoretically be overstepped by gathering a much larger dataset that could cover the extensive vocabulary and eventually by redesigning the network's architecture. However, this would mean a regression of the work being conducted, leading to a new phase of preliminary tests. To overcome the problem and make use of all the gathered results and previous observations, we simplified the model by expanding it. This resulted in a final model consisting of three identical networks yet independently functional, each of them in charge of processing its own sequence (notes, velocities and durations). In a global illustration of the model's operational process, the MIDI dataset is given as an input to all networks and each one parses the files preserving the sequences holding its specific elements and then left to its training phase. After such, the music composition generation stage is then performed in a cascade manner, layering each predicted sequence. Finally, the elements are then combined in order to shape each note's expressiveness before being converted to a final MIDI file.
Regarding this final prototype's design, the architecture used in the first network was kept (Figure 4.1 - Existent layers), meaning this that each of the three networks that compose the model are identically structured.

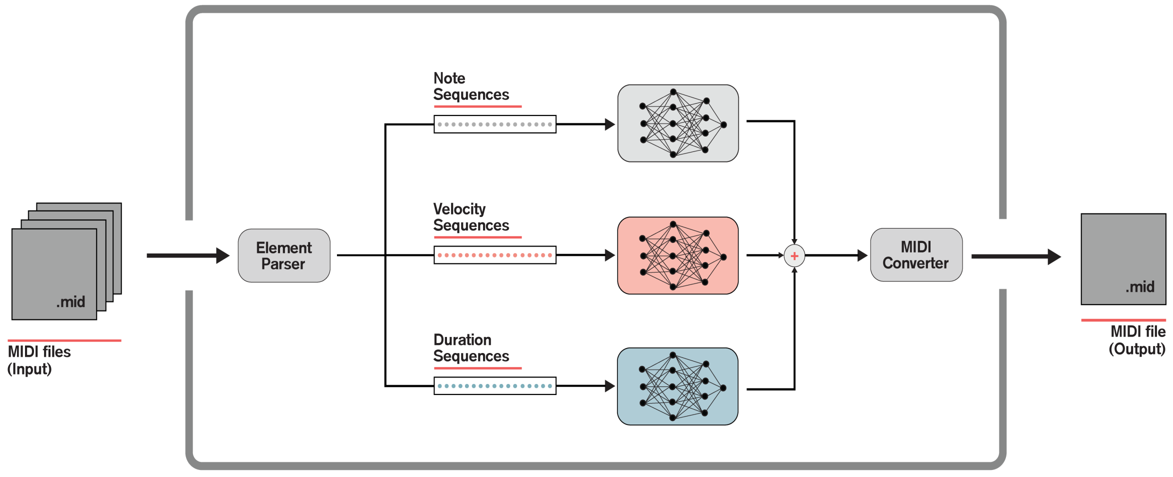


Figure 5.5: Overview of the Model

5.4 Training Process and Adjustments

Having structured the final model, the next step was to set all of the three artificial networks to their training phases. These were sequentially performed in the sense that the networks didn't train simultaneously but rather one at a time, mainly due to computational constraints. Also, this individualization allowed us to perform some adjustments to the networks forthwith - based on the learning rate of the first one being trained (notes network) - and has proved to be a time saver approach - once that having all networks being trained in parallel would exponentially slow down all learning processes. Knowing that all networks are independent of each other and that each handles distinct elements (note that although all three networks were designed to receive sequences of integers, we couldn't state beforehand whether the essence of these different sequences would have an impact or not on the learning process), we had no guarantee that an adjustment to one network would necessarily mean an improvement in any other. However, it allowed us to understand better which modifications could potentially be beneficial in that context since the adjustments could be performed concerning the network itself or even the dataset.

For this new set of training sessions, we gathered a set of 156 MIDI *Bach* compositions files. Knowing from previous experiments (performed in the first semester regarding the preliminary work and tests) that some files may not be eligible due to encoding issues, lack of piano parts or even unnecessary repetitions, the training set was finally reduced to 96 MIDI files after a careful examination. The motivations behind the choice on the composer (*Johann Bach*) were mainly practical in the sense that we were able to find a satisfying number of files in the desired format (also live performances, as we're about to detail further in section 5.4.2 Velocities Network). In addition, and observing the history of similar work, compositions by *Bach* have been a recurrent choice to study the performance of ANNs on artificial music generation.

In the following subsections, we detail each of the three networks' performances during training as well as the adjustments applied and specific modifications that were added.

5.4.1 Notes Network

The first artificial network set to train was the one intended to learn from note sequences - similar to what was done in preliminary work tests. For the first training session, we maintained the setup previously employed and used all the 96 MIDI files as the training set - making a total of 104792 notes. Figure 5.6 exhibits the loss minimization through time.

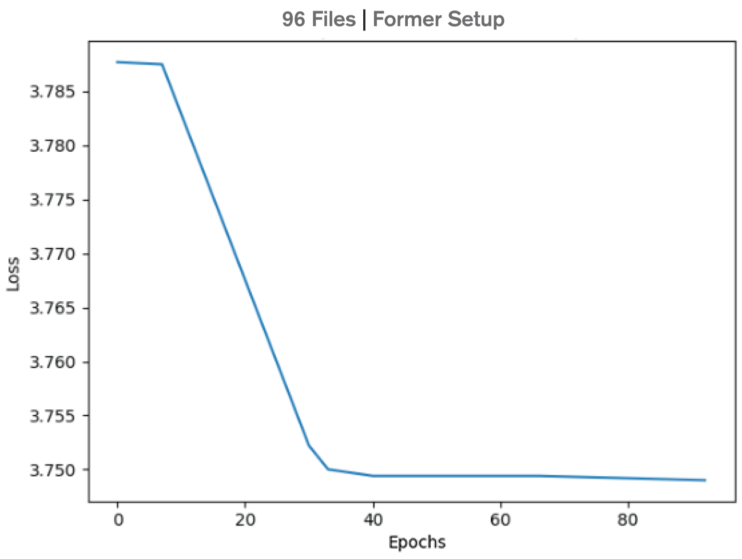


Figure 5.6: Former Network Architecture - Training Performance with 96 *Bach* MIDI files (Graphical Representation)

This training session took about five days to reach the 90th epoch (an average of 18 epochs per day). As we can see from the figure above, the network shows some improvement - yet barely significant - during the first 30 epochs and stabilizes from that point, making no relevant progress. As for the lack of consistent minimization and time consumption - in five days it has shown an improvement of only about 0.04 - we were set to adjust the network.

Our first approach was to make the network more powerful by adding two LSTM layers (with 512 neurons each, meaning that we replicated the existent LSTM layers). With this, we intended to understand how the robustness of the network would impact the learning process - knowing beforehand that more complex artificial networks usually take longer to train. Figure 5.7 presents the training session after these changes.

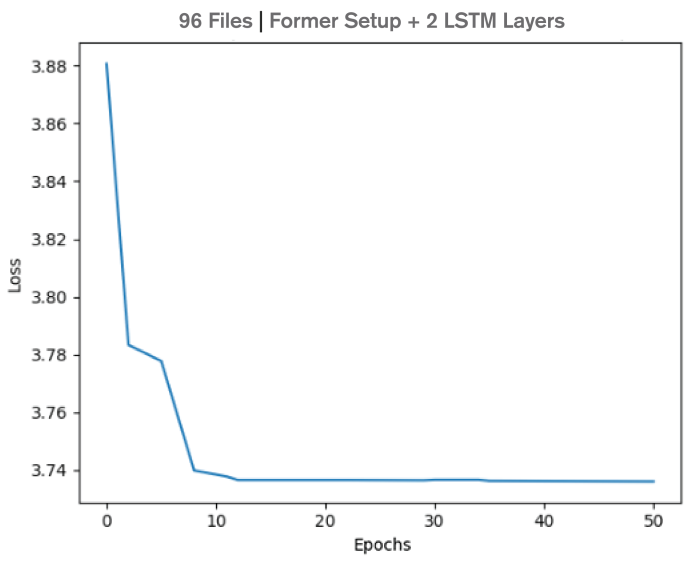


Figure 5.7: Former Network Architecture Plus 2 LSTM Layers - Training Performance with 96 *Bach* MIDI files (Graphical Representation)

Similarly to what we observed in the first training session, we also perceive an initial minimization during the first epochs - although this time it happens after fewer epochs - followed by an undesired stabilization. Although the initial minimization is greater than the first attempt (with this setup we observe an initial improvement of about 0.14) it also wasn't a significant decrease. In addition, the presented training session took also about 5 days, meaning that in the same time frame tested with the first training session this new setup performed 50 epochs - almost half than the first one - showing no reasonable improvement. Regarding our time budget, we were willing to find a better settings combination.

In order to observe the impact of the dataset size on the learning process, we reduced the training set to 36 files. This selection was made based on the number of notes per file and also the style of the composition (e.g. *sinfonias*, *chorales*, *fugues*) guaranteeing the dataset diversity. This new set contained a total of 65488 notes. Furthermore, we excluded both early added LSTM layers making the networks' architecture back to its original structure. On the graph below we can follow its performance with the new training dataset.

On the graph illustrated in Figure 5.8 we observe how these changes affected the network's learning speed, reaching 160 epochs in 5 days. We can also observe an initial improvement just as we have witnessed in both last training sessions, followed by the same stabilization pattern. Although this last stability point is obviously a state to avoid - especially on early stages of training - this can be an indicator that the dataset isn't too short once the loss hasn't decreased too fast to reach zero early (premature convergence), which could be a sign of overfitting.

To improve the learning, we decided to change the network's parameters yet keeping its

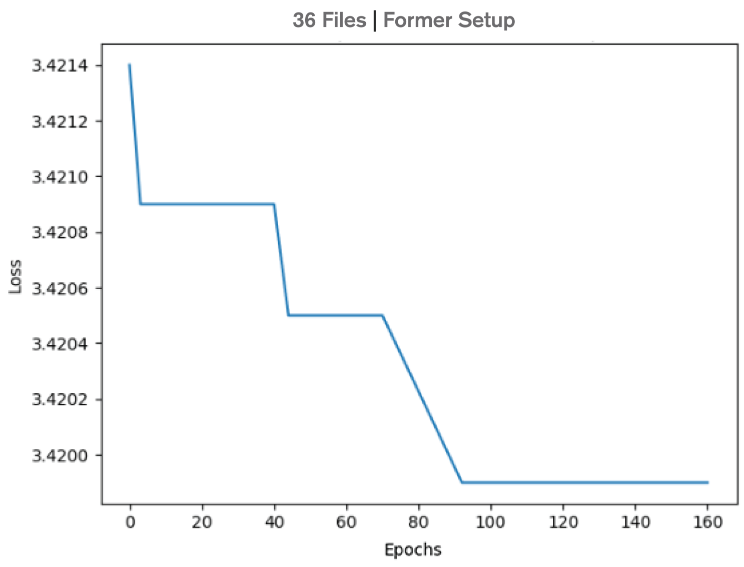


Figure 5.8: Former Network Architecture - Training Performance with 36 *Bach* MIDI files (Graphical Representation)

original architecture. Moved by the will to observe how the network's internal state might influence its learning progress, we reduced each LSTM layer number of neurons from the former 512 to 100. Reducing the neurons in each cell consequently means decreasing the network's complexity level, hence resulting in a less powerful one. However, having a less complex network doesn't necessarily mean that its performance is then compromised. The network as it was - with the former number of neurons in each LSTM cell - could be too complex for the dataset and respective patterns, which might delay the learning process due to its internal complexity. We also reduced the sequence number (heretofore set to 100) to 75. This sequence value defines the number of past elements after which the network makes the prediction on the next one (see section 4.3.2) - in this case, we are dealing with note prediction, so it essentially means that the network will learn to make predictions on the next note based on the last 75. With this adjustment, we wanted to examine its implications on the learning phase having a less complex network.

As we can observe from Figure 5.9, the latest adjustments resulted in the best combination of parameters when compared to the last configurations. As expected, the complexity reduction produced a faster network, performing an average of nearly 75 epochs per day. Looking at the loss value, it shows a consistent minimization pattern. These both factors led us to preserve this model to learn from note sequences. After one month of training (2272 epochs in total), the network reached a minimum loss value of 1.960 after an initial loss value of 3.7545 - a total improvement of 1.7945. Although the minimization hadn't yet shown a state of utter stabilization, some compositions were generated at this point to evaluate their quality. After considering the artificially generated melodies as sufficiently good, we stopped the training session once we had another two training sessions planned until the end of the first half of this second semester (Velocities Network and Durations Network).

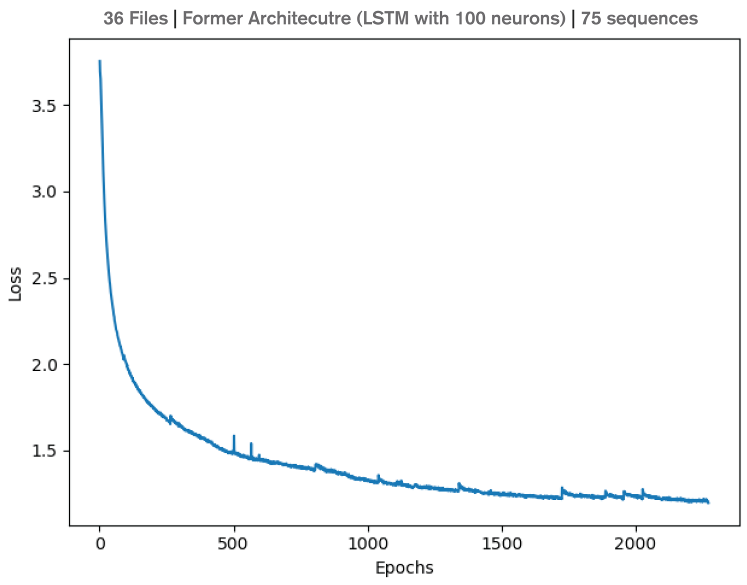


Figure 5.9: Former Network Architecture (LSTM neurons reduced to 100) - Training Performance with 36 *Bach* MIDI files (Graphical Representation)

5.4.2 Velocities Network

The second network was destined to learn the first expressive elements - velocities. As described earlier, velocities are the encoded values in MIDI files that represent the attack of each note or chord (i.e. how hard it should be hit). There are two main ways of registering velocities in a MIDI file: by recording a live session or introducing the values manually. As for the notes, for instance, there is no practical distinction on having the notes recorded or manually registered as they will always assume the same value (i.e pitch). On the other hand, velocities recorded during a live session are genuine in the sense that the emotional tension performed was real, whereas manually listed velocities aren't. In addition, not all MIDI creators introducing the notes and durations manually take into account different velocities, assigning a generic and invariable value - in fact, most of the files in our training set had the same velocity value for all notes during the entire composition.

The mentioned value invariance regarding velocities would cause the network to overfit the values and to learn no expressiveness, most likely resulting in sequence repetitions once it had no sufficiently expressive data (i.e. velocity variance) to learn from. This led us to reject the former dataset. Acknowledging the need of having a new collection of MIDI files comprising natural velocity sequences and patterns, we gathered a new set of files recorded from live performances. This new dataset was created by extracting several MIDI files provided by the International E-Piano Competition [2], from where we extracted different performances of *Johann Bach's* compositions. This allowed us not only to have several sequences of expressive performances regarding velocities but also to assure the pieces weren't performed by beginners but rather by experienced pianists. The different compositions gathered were then reduced to match the number of notes (and thus duration) to ensure an equitable dataset size. This resulted in a total of 62611 velocity elements. This dataset can also be found in Appendix B (9). Figures 5.10 and 5.11 hold the histograms representing the number of different velocity values - as well as their diversity - in each dataset, providing a notion of how the live recorded files differ from the manually encoded ones in terms of note attack.

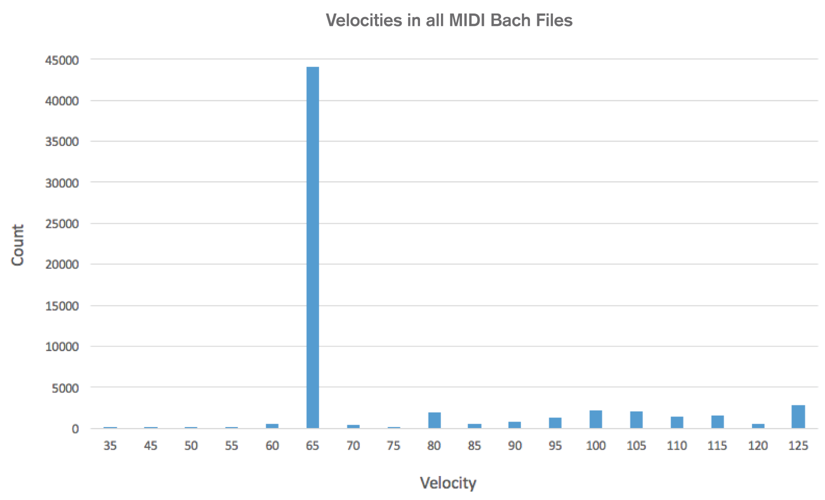


Figure 5.10: Histogram of velocity values frequency - *Bach* MIDI dataset

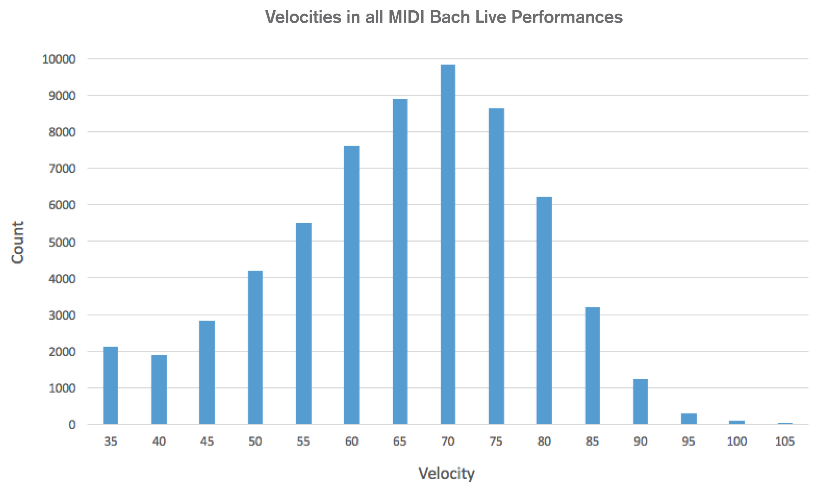


Figure 5.11: Histogram of velocity values frequency - *Bach* Live Performances MIDI dataset

After the live dataset arrangement, the Velocities Network was set to its first training session. Considering the good performance regarding the setup on the last network, the same one was used to train the present session. The graph below illustrates its evolution through time.

As one can observe, the loss value reaches a stability point right after the first epochs not displaying significant improvements. To ensure that this stability status was continuous (i.e. not reaching any better result) we waited some more epochs to verify this fact, which indicated to be true. The learning performance illustrated in Figure 5.12 below also allows us to note some unexpected outliers on the last epochs, meaning that at some moments

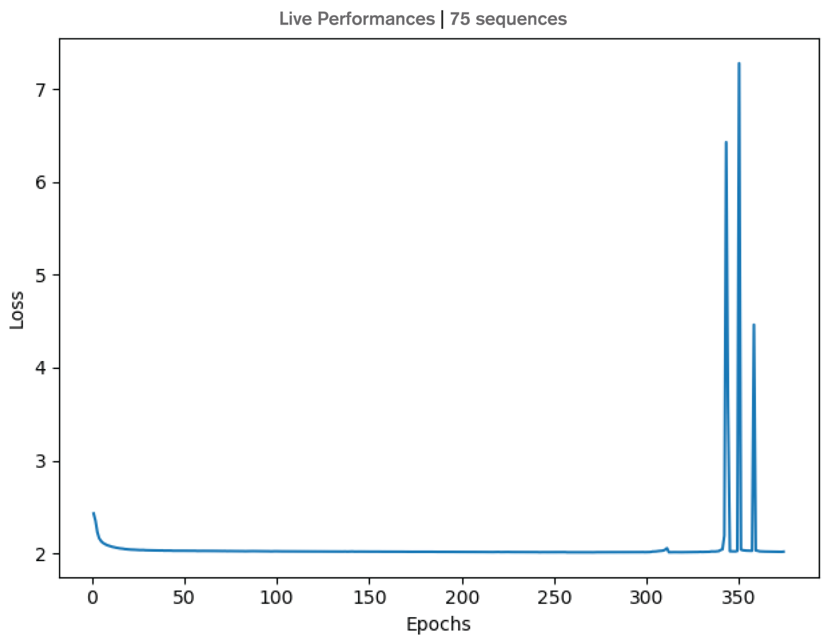


Figure 5.12: Velocities Network - Training Performance with *Bach* Live Performances MIDI files (Graphical Representation)

the loss value 'explodes' in a great manner. Although these scenarios are undesired and to be avoided - specifically in our work -, these extremely high values provided us an indicator that the problem might not be in the architecture itself but in the learning process. Moreover, the same setup provided reasonably good improvements on the last network and although this doesn't necessarily mean that the same must happen in the present case - learning relations between velocities might be more complex to learn than durations or even notes -, we took the approach of adjusting the learning process. The learning process is not only influenced by the network architecture itself but also by the learning rate which takes a significant role regarding the search space. Until this point, the network was programmed to use *RMSProp* (Root Mean Square Propagation), an optimizer that controls the learning rate by estimating the squared gradients and moving (i.e. searching) keeping an average of past gradient values. Believing that the optimal minima were not being reached due to a possible unsuitable learning rate, the former was changed to *Adam* (Adaptive Moment Estimation), an optimizer - another one provided by Keras framework - that essentially decreases the search speed to avoid jumping over desirable minimum [17] [56].

We also introduced a new delimiter algorithm in an attempt to ease the learning process by reducing (i.e. uniform) the vocabulary - and consequently computational cost. Knowing that velocities are encoded between 0 and 127, we examined all different velocities by changing its value directly on a MIDI platform (i.e. GarageBand). With this inspection, we wanted to understand the impact of similar values on the perception of the note, verifying that adding two integers to each value makes no relevant difference (i.e. comparing a velocity of 40 with 42). Based on this notion, the algorithm shifted any value to its closest in sequences of five (e.g. 42 would become 40 and 43 would be shifted to 45). This normalization reduced the velocities vocabulary from 71 to 26 possible values. Below we display the graph that holds the evolution of the last training session, using the delimiter algorithm and after the optimizer adjustment.

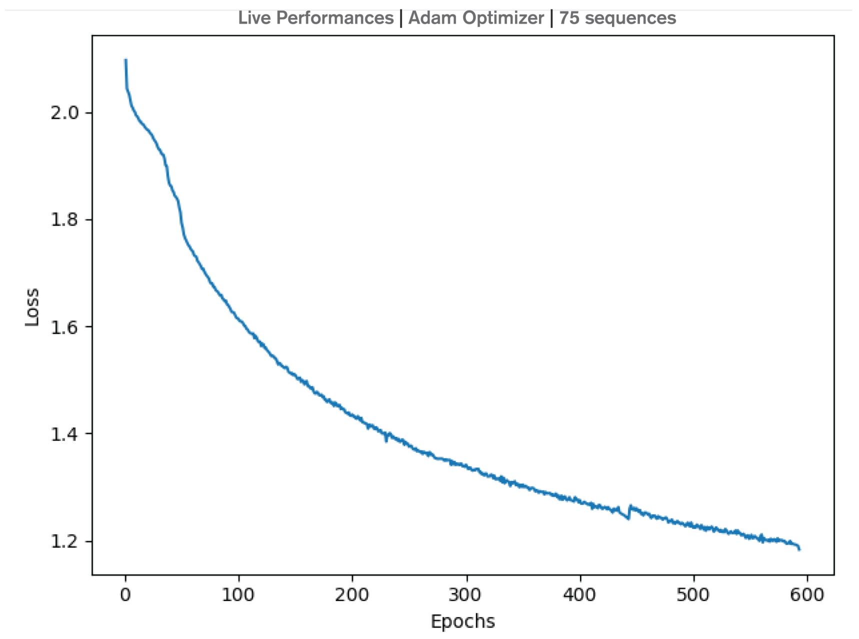


Figure 5.13: Velocities Network (*Adam* Optimizer) - Training Performance with *Bach* Live Performances MIDI files (Graphical Representation)

The loss value evolution presented in Figure 5.13 shows how adjusting the learning rate might affect an artificial network's training process, particularly when compared to the last training session (Figure 5.12). Here we state a clear improvement during eight days of training, outputting a loss value of 1.1838 when the session was stopped to initiate the last network training phase.

5.4.3 Durations Network

Having the last network as a starting point, we applied the same adjustments to the Durations Network: same architecture, LSTM neurons reduced to 100 in each cell and a sequence number of 75 - with the recently changed *Adam* optimizer. With this, we wanted to examine how the same setup would perform over duration sequences - although dealing with sequences as well, the relational patterns between notes or velocities could differ from durations. This means that we could never state beforehand that learning duration sequences are more (or less) complex than notes - or velocities - that build the composition. For the same reasons detailed in the last section, we decided to train this network with live performance durations. Although human performances follow the durations specified on a musical sheet, it is hard to follow strictly those durations without any unintentional change. Knowing that a computer (in this case our model) is quite likely to learn and apply the exact durations, we then decided to use live performances as duration variation training set in order to avoid a possible mechanical feel on the generated compositions.

During preliminary experiments we realized how irregular MIDI files could be regarding their encoding. When analyzing the dataset, we identified some issues regarding note durations when parsing the files as a few notes displayed duration of 0 - meaning that according to the output they should never be heard. However, opening those files and listening to the composition led us to the conclusion that those notes should, in fact, be heard, although for a short period of time. To overcome this instability a short algorithm was included to perform when extracting the sequences from the files. Essentially, this complementary function works as a value delimiter, identifying each duration with a value of 0 and assigning to it a new value of 0.25 (corresponding to a sixteenth note). This generic value was chosen based on its short length, after measuring the zero-valued durations real durations on the musical sheet. Figure 5.14 illustrates the training session under the mentioned terms with the same training dataset used in the best Velocities Networks setup.

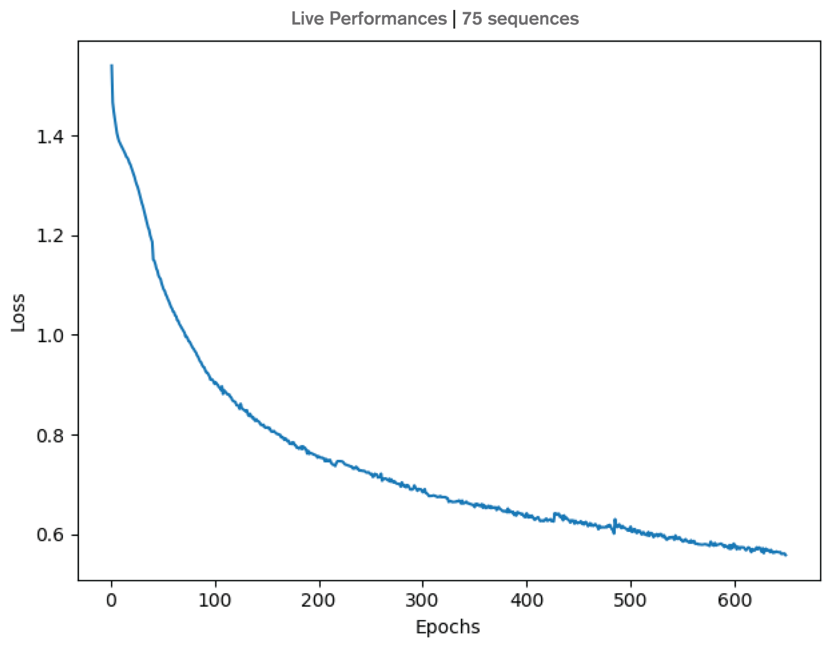


Figure 5.14: Durations Network (*Adam* Optimizer) - Training Performance with *Bach* Live Performances MIDI files (Graphical Representation)

The loss function calculations throughout the training session depict the expected minimization behaviour - as seen on the Velocities Network after changing the optimizer. Starting with a loss value of 1.5405, we observe a quick descent during the first 200 epochs. The training session was stopped at the 834th epoch (corresponding to 11 days of training) outputting a final loss value of 0.526. As we can state by comparing the three training sessions, there are signs of different complexity levels regarding different sequences - once that for the Notes Network it took about a full month to reach stability, whereas the Durations Network only took a week and a half to show signs of rate slowing down. This can possibly be an indicator of the referred complexity, in which note sequences' relations might be harder to learn.

By observing the graph, we can state that there is also no visible stability pattern, meaning that there might be a possible margin of improvement. We decided to begin the generation phase to analyze the final model's ability to generate expressive musical compositions, not only to get an overview of the prototype's capabilities - and even testing its correct operation - but also to consider if it could be labelled as ready to proceed to the testing stage (i.e. User Testing).

5.5 Expressive Musical Compositions Generation

As most of the progress made during preliminary work was kept and used to develop the final prototype, the generation process applies the same logic detailed in section 4.3.3 (Music Generation). Although we had the need to apply changes and adjustments considering the complexity of the model and training sets, the artificial music generation was performed in the same way yet at three different levels (or dimensions). Once trained, to each network (i.e. Notes Networks, Velocities Networks and Durations Networks) was given a random sequence as an input in order to predict a defined amount of elements forming the final sequence. After such, all the predicted vectors were combined to proceed to the creation of the MIDI file that encodes all the predicted events, resulting in the final music composition (having a predefined number of notes and consequently their expressive values). Figure 5.15 provides an extended view of the final prediction phase and consequent composition creation - a more focused and detailed illustration of the same area covered by the full model overview in Figure 5.5.

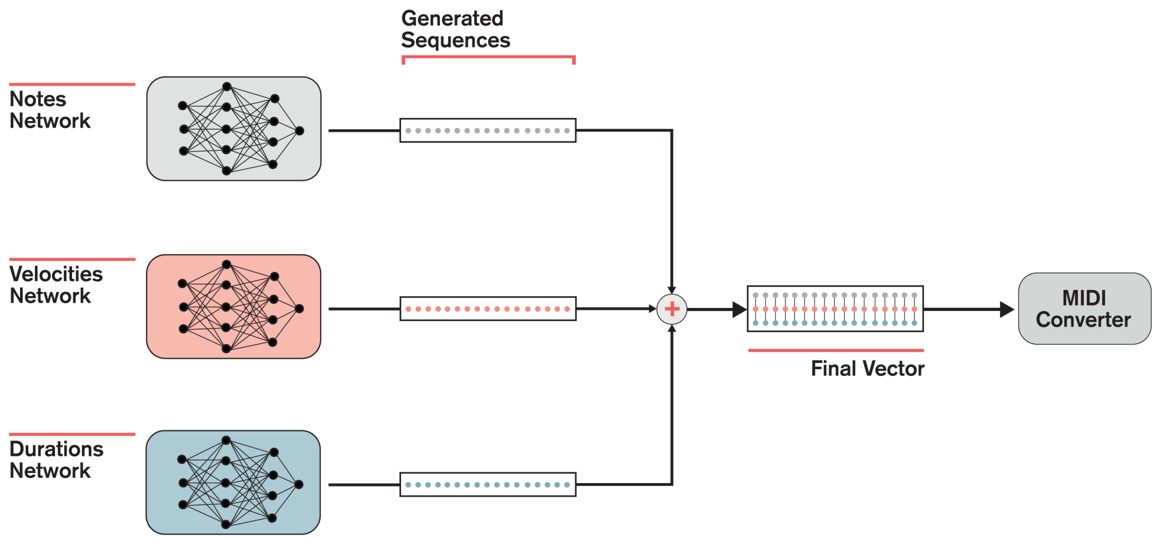


Figure 5.15: Model's Sequences combining process

In order to evaluate the model's capability to produce expressive music, several compositions were generated. For clarification purposes, we'll be referring to this first evaluation phase as 'local tests' once they were performed by the members of the present research group and also to make a distinction between these and user testing (a topic that will be addressed in the next chapter). The initial step regarding these local tests was to make a selection from the set consisting of various artificially generated compositions. We listened to 30 generated compositions and identified the ones containing mostly consonant melodies and the compositions we considered having interesting note sequences or variations. From these last pieces, we selected the ones resembling expressiveness, that is, the ones we considered being the most expressive. The decisions made on expressiveness were based on the full compositions themselves, meaning that we didn't focus on a particular expressive element at this stage (durations or velocities), but rather the impact both had on the musical compositions. This selection process resulted in a final set of 8 musical compositions (potential compositions to be used in User Testing).

Until this point - before designing the prototype of the model - all the artificially generated musical compositions were composed of note sequences only. To all notes generated during preliminary work were assigned default (or generic) velocity and duration values as it was mainly a procedure of understanding the first network's ability to generate stable melodies. Consequently, all those non-expressive compositions had absolutely no variation regarding the expressive elements introduced in this chapter. This means that if we were to describe the expressive behaviours of those pieces through a bi-dimensional graph, we would then have a continuous line on both durations and velocities. Figure 5.16 below illustrates this idea, plotting the expressive elements through time (or note sequence) of a music composition generated by the first network trained with *The Beatles* dataset - in fact, we could be using any composition generated by the former network, as all expressive values were set by default.

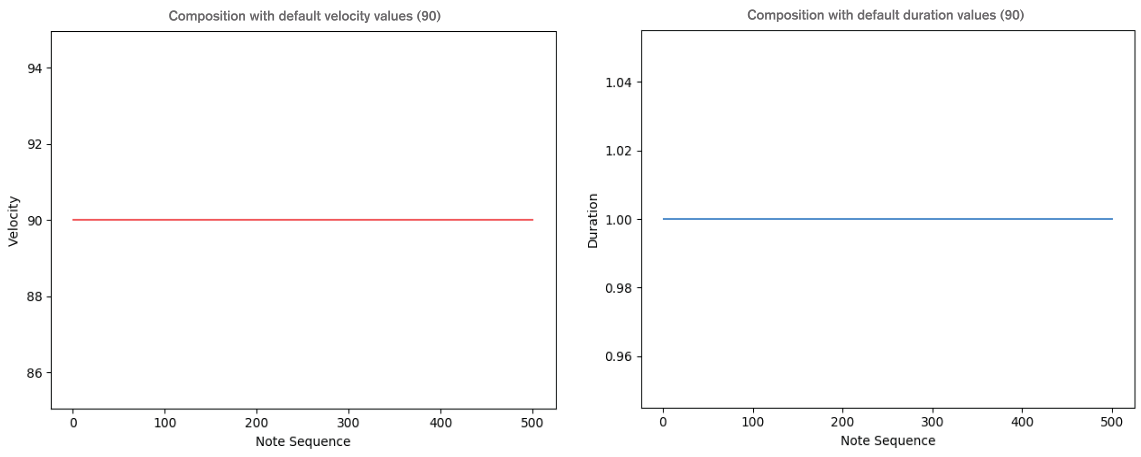


Figure 5.16: Compositions with default expressive values (Velocities and Durations)

The graphs display the distinct expressive behaviours through time, from the beginning to the end of the composition (i.e. from the first note to the last one, resulting in a discrete representation of the piece). The ordinate (vertical axis) represents the expressive value assigned to each note - 90 in the case of the velocities and 1 in durations (or quarter note). Both graphs represent what we have been describing as the absence of expression (or non-expressiveness) in a music composition. We then used the same method to study the new compositions generated by the new model, particularly the ones identified as the most expressive. This enabled us not only to visually perceive its expressive variations throughout the compositions but also to compare expressive behaviours with the non-expressive one (Figure 5.16). The following figures depict this graphical vision of the expressive elements of four pieces generated by the new model. For reference, we recall that velocity values range from 0 (lowest intensity) to 127 (highest intensity) and durations have the lowest possible value of 0.25 (sixteenth note). Below each figure, we provide a link to the audio of each portrayed composition (hosted in the music streaming platform SoundCloud).

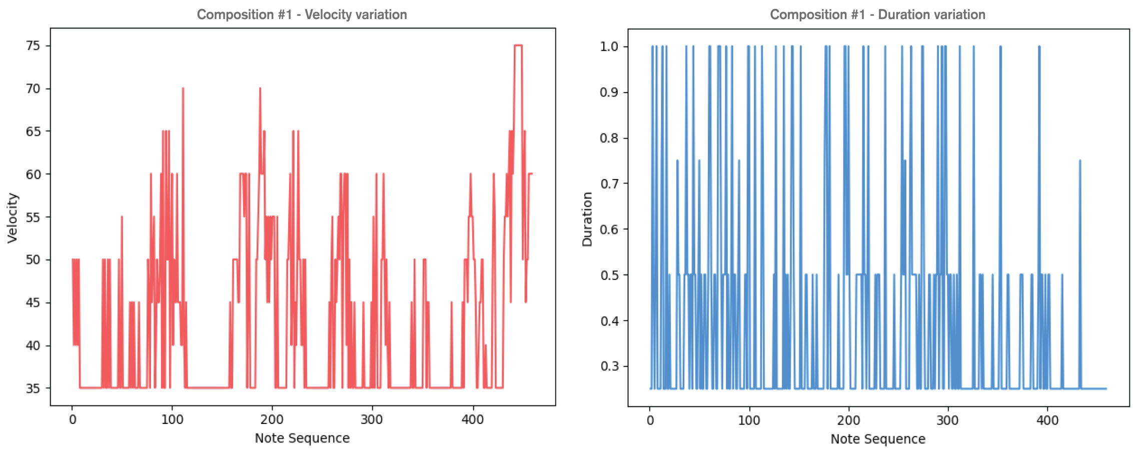


Figure 5.17: Expressive elements variation throughout the piece (Composition 1)

File 13: Model's Expressive Composition - 1

url: https://soundcloud.com/josemariasimoes/file13/s-hP5GP

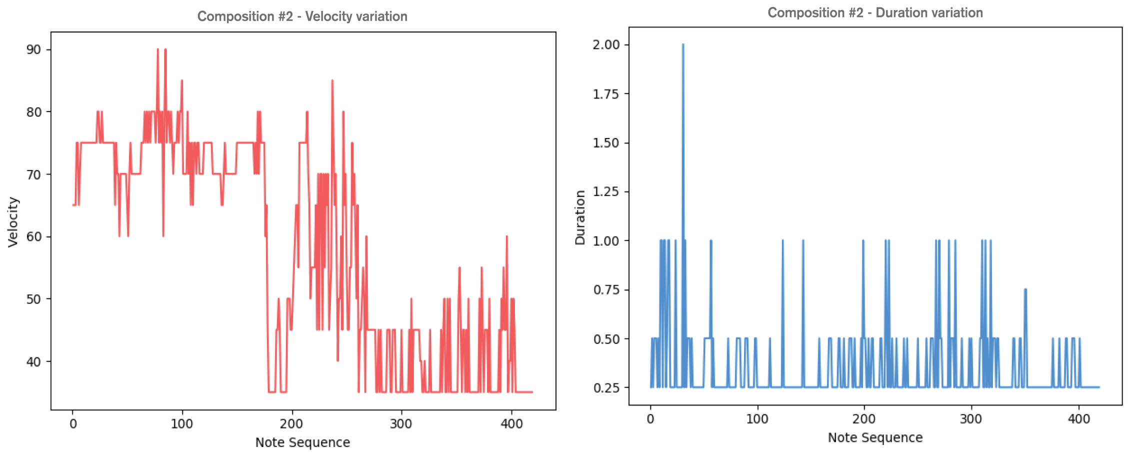


Figure 5.18: Expressive elements variation throughout the piece (Composition 2)

File 14: Model's Expressive Composition - 2

url: https://soundcloud.com/josemariasimoes/file14/s-qg4eg

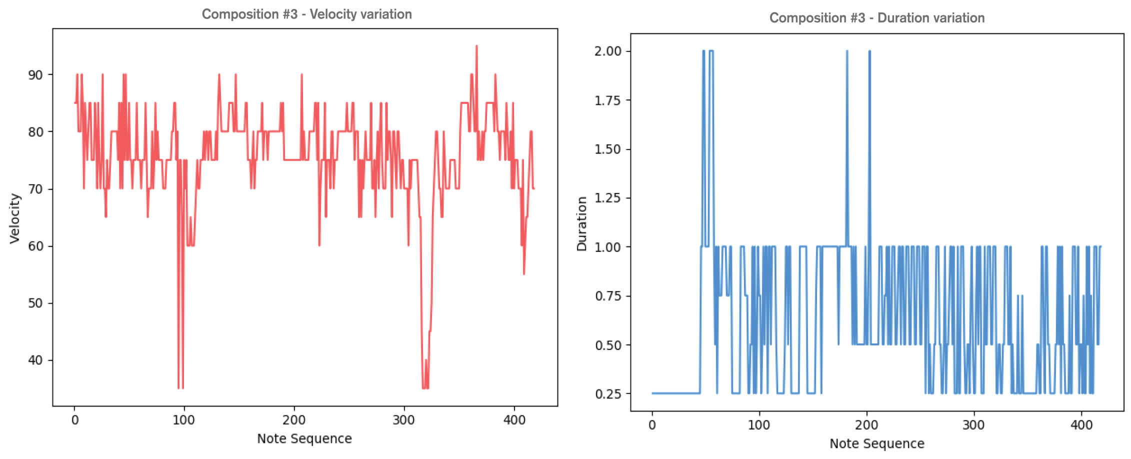


Figure 5.19: Expressive elements variation throughout the piece (Composition 3)

File 15: Model's Expressive Composition - 3

url: https://soundcloud.com/josemariasimoes/file15/s-1U4CP

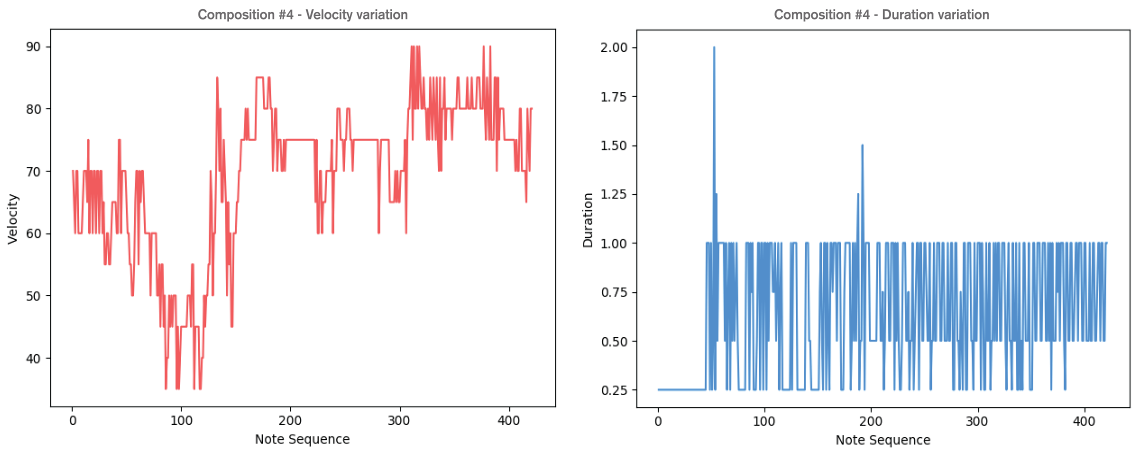


Figure 5.20: Expressive elements variation throughout the piece (Composition 4)

File 16: Model's Expressive Composition - 4

url: https://soundcloud.com/josemariasimoes/fil16/s-HL3sv

The first graphical representation (Figure 5.17) reveals how expressive variations behave during the first composition. In contrast to the non-expressive piece (Figure 5.16) we can now observe several changes in both lines reinforcing the presence of musical expression and diversity regarding its possible values. Inevitably, these changes are visually represented by sudden shifts as the intensity may suddenly drop or increase (velocities) and durations may vary as well. However, we can also note some horizontal lines (i.e. unvarying values) meaning that during a certain note sequence these assume the same velocity or duration until the end. From all these factors, especially when combined, we identify expression. We don't take these regular abrupt changes as evidence of expressiveness solely, but rather the combination of constant values followed by a sequence of variations (or the other way around). Although this might be easily perceived regarding velocities graphical representation, we must state that we consider that variations within note durations are as important to improve the composition.

Another interesting factor regarding these graphical representations is how they can visually depict expressive patterns that represent the compositions "narrative" according to each expressive element. This is easier to observe in compositions that follow more structured patterns, especially velocities. The first composition is quite unstable in terms of velocity values, holding a lot of constant - and drastic - variations throughout the piece. The second one (Figure 5.18), on the other hand, despite holding several velocity changes it tends to compress the range of values during a certain amount of time. This results in a visual narrative as the composition begins with high-intensity values followed by what can be seen as a transition period (i.e. more abrupt changes), and ending with less intense values - or smoother. In contrast, the last composition's velocities (represented by Figure 5.20) present almost the contrary expression regarding note intensity, ending with higher values (almost resembling a *crescendo*). The third composition (Figure 5.19) appears to be the most consistent one in terms of velocity values, having a few abrupt changes and maintaining most values within a roughly defined range, causing the composition to be mostly 'loud' or intense.

Although the graphical representations of durations didn't provide such a direct picture of how those patterns and variations may influence the composition, we can notice the differences between the expressive and non-expressive plots. It is interesting to note how durations can also assume abrupt changes (as seen in Figure 5.17) but still follow a certain range pattern. For instance, in Figure 5.19 the variation line seems to form a pattern between the values 0.25 and 1, with occasional longer notes. The final composition's duration plot (Figure 5.20) provides a similar pattern, yet showing more sequential changes making the graph look denser - which has a totally different impact on the composition despite the similar values range.

The performed local tests allowed us to examine the model's capability to generate expressive musical compositions, confirming that there is a difference between non-expressive and expressive compositions - either by listening to the compositions or by analyzing the variations of the expressive elements graphically. We also verified the model's ability to learn these expressive patterns as it has done before regarding note sequences. Analyzing the set of musical compositions generated by the model, we considered that these compositions demonstrate the impact that expressiveness can have over a music composition. It is also important to reinforce that although our main focus is directed to expressive elements, notes are an important part of a music composition - if not the most important one. Regarding these, we consider that the model has generated pleasant note sequences, with no major dissonance and even resembling some interesting passages. This - which wasn't verified on the pieces generated during preliminary work - might be a result of the patterns applied by *Johann Bach* in its compositions, likely more complex to learn than the other training sets used before. It is also important to note that the networks that constitute the model had significantly more training time, which might be evidence of *Bach's* compositions complexity when compared to the Pop/Rock sets used before (section 4 Preliminary Work). Looking at each generated compositions as a whole, most of the compositions aren't fully variant as some repetitive patterns may emerge during a small period of time, but most of the melodies heard among the selected generations were considered as pleasant, with some interesting expressive moments which we consider to be improving the dynamics of the compositions.

Following this local examination of the expressive compositions generated by our prototype, there was a need a validation process to measure the network results efficiency. In the next chapter, this validation process is detailed.

Chapter 6

User Testing

Music - as for any other forms of art - is always susceptible to a certain level of abstraction which consequently makes it difficult to measure its quality in a simple or straightforward way. Having that in mind, we decided that the best way to validate our artefact was to conduct user-testing sessions where participants would be asked to rate a series musical excerpts. The abstraction involved in this analysis approach and subsequent quantification could lead to undefined results, however having a sample of a population might result in answering patterns that could suggest that the generated compositions are or not good - and whether expressiveness provides some kind of improvement or not. This is also the reason why we decided not to define a specific threshold which might lead to misinterpretations. For example, if expressive compositions have 10% more rate than non-expressive ones concluding that they are in fact better seems a quite strong conclusion. Instead, we decided to analyze the results based on their global tendencies and use those values as indicators of suggesting proof. In sum, we used these tests to examine how the generated compositions would be rated when compared to pieces composed and performed by humans. In addition, these terms of comparison let us understand if there is any sign of influence by the studied expressive elements - whether in human compositions or artificially generated.

6.1 Test Setup

To design the user-test scheme, it was important to narrow down some important points to follow in a pragmatic approach:

- Compare artificial compositions with human compositions.
- Find which expressive element has more influence on human/artificial compositions.

- Assess how full expressiveness influences human/artificial compositions.
- Test if artificial expressiveness can be close to human expressiveness in human compositions.

The first point is used to examine the quality of the generated melodies. The presence of human performances (and compositions) on the tests allows a reliable term of comparison. As earlier discussed, rating a music composition can be quite abstract and misleading but we consider that adding human compositions provides a much more solid context in which we have a reference point. On the second point, we want to analyze how each expressive element (i.e. durations and velocities) influences musical compositions - again, on both human and artificial compositions - and consequently the improvement they provide when combined (third point). By structuring the test with these main points we are also able to explore further questions and address other secondary goals proposed in our current work. The fourth point establishes exactly one of those goals: merge human compositions with artificial expressiveness and study its effect - again, through comparison.

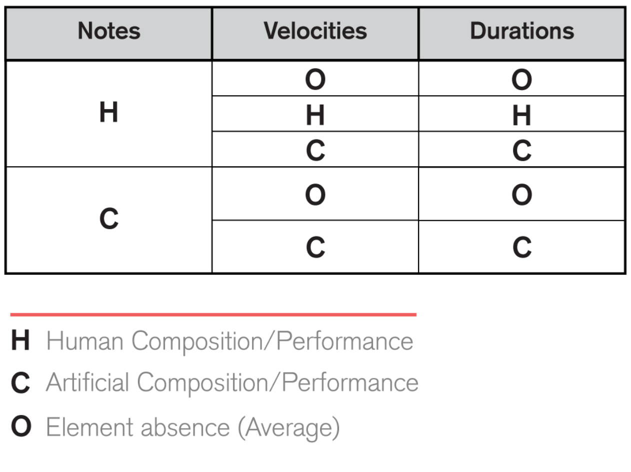
Having these terms defined, we then needed to have two different sets: one consisting of the expressive generated compositions and the other composed of live *Bach* performances. The first set was built with the compositions studied in the last chapter. As for the second one, we opted to use live performances of *Bach* pieces - extracted from the same database used before (International E-Piano Competition [2]) - the files used regarding human performances can be found in Appendix B (9). The choice of using *Johann Bach* compositions seemed a reasonable option once our model was trained with compositions from the same author, a decision taken to promote fairness between both sets to be compared.

As we intended to test several parameters, we decided to use only excerpts of the previously selected compositions. Considering the extension of the user-tests designed, keeping the original length of all compositions would mean an extensive test once some of the gathered live performances could last for 2 or more minutes - as for the generated ones. This could possibly hinder the analytical process, as it would require long user-test sessions - which might be tiresome - and a great amount of concentration. From each of the selected compositions we extracted an excerpt with at least 15 seconds and no longer than 30 - naturally depending on the composition itself. This musical 'shortening' enabled much lighter sessions in which the participants could easily and quickly rehear the excerpt with no need for excessive memory exercise, avoiding undesirable fatigue. We have then chosen 4 excerpts from human performances and 5 artificially generated expressive compositions.

To ensure impartiality on the tests, we developed an algorithm that assesses the singularity of each of the generated excerpts used. This was also a way of searching for any sort of overfitting, which would mean that the generated compositions contained several note sequences identical to the training set - meaning that the tests would serve as a comparison between *Bach* pieces only. It is important to note that defining whether a composition is

similar to another to an extent that one can consider plagiarism or not is not linear. After some research - and to our knowledge - there is no defined value or parameter to identify these occurrences as the cases of plagiarism are always subject to some level of abstract notions existing no standard metric. In addition, we must also note that our case is not completely a search for evidence of plagiarism, but rather guaranteeing that the participants rated melodies that were composed by the model. For that, we needed to assure its differentiation from *Bach* pieces. Having no established threshold, we used the algorithm to analyze this issue.
The results have shown that the excerpt with a higher value had only a sequence of 10 notes similar in all 65488 dataset notes - considering that all excerpts last between 18 and 24 seconds containing an average of 150 notes, a sequence of 10 notes is less than a second and a half. This values led us to conclude that there were no significant similarities between the excerpts used and the training set.

To attend all the aforementioned points, we divided the test into ten different parameters (or sections). These parameters were developed based on the different combinations of cases to evaluate. First, we pointed out all three elements of the musical compositions under examination: Notes, Velocities and Durations. After such, several parameters combinations were determined to include all the aspects to study. Figure 6.1 illustrates the parameters relational table. These parameters were defined assigning the letter '**H**' (Human) and '**C**' (Computer) to identify the origin of the element - that is whether the melody of the excerpt (i.e. notes) was composed by humans - in this case, *Bach* - or artificially generated by our model - computer. The same logic is applied to both expressive elements, assigning an '**H**' to identify human live performances and '**C**' for artificial expressiveness. Figure 6.2 depicts the chosen parameters combinations among the possible ones - as we'll describe later in this chapter, we had the need to leave some combinations aside in order to perform the tests in a reasonable time and not extending them for too long.



Notes	Velocities	Durations
H	O	O
	H	H
	C	C
C	O	O
	C	C

H Human Composition/Performance
C Artificial Composition/Performance
O Element absence (Average)

Figure 6.1: Elements Table (Possible Combinations)

Apart from the letters defining computer (**C**) or human (**H**) compositions/performances,

Chosen Combinations
HOO
HHO
HCO
HOH
HOC
HHH
COO
CCO
COC
CCC

Figure 6.2: Chosen Parameters combinations

a capital letter '**O**' was introduced to represent the 'absence' of a certain expressive element regarding the parameters. As an example, comparing the parameter HHH with the parameter HOO, the first one represents a *Bach* composition performed live with all the expressive elements (unaltered excerpt), and the second one represents a *Bach* composition performed without expressiveness. To totally or partially exclude expressiveness from the excerpts we essentially cancelled any variance within note velocities and/or durations. This was achieved by simply calculating the average value of each expressive element considering the full piece to which excerpt belongs. For instance, considering an excerpt composed of 100 notes belonging to a composition that has 800 notes, we would then calculate the average note duration and velocity in the entire piece. Assuming that these calculations result in an average duration of half note and an average velocity value of 70, as an example, we then apply these values to all of the 100 notes of the excerpt - resulting in a non-expressive version of it. This process was applied to both sets (human and artificial compositions) meaning that for both sets of excerpts we applied this procedure to each excerpt. With that, the capital letter '**O**' notation means the 'absence' (or non-variance) of the specified expressive element.

It is also important to note the parameters where artificial expressiveness is applied to human compositions (third and eighth parameter), as described in our secondary goals. These parameters were attained by extracting expressive elements generated by the model and replacing the expressive ones originally on the excerpt.

By having these different 'configurations' among excerpts we were able to evaluate the impact each element has on it. This can be achieved by analyzing the results via parameter comparison, for instance, COO *vs* COC to evaluate the impact of having variance among note durations. Other parameter configurations could have been added to the tests but we have included the ones we considered required to cover our goals and also to keep the tests not excessively extensive.

6.2 Participants Demographics

In this section, we provide some demographics concerning the participants that performed this user-testing phase. These characteristics are graphically displayed below for informational purposes and sample characterization.

For the present study, we conducted 30 user-testing sessions. The selection of participants held no strict rule apart from the music education level - we were willing to test our goals with a sample in which the majority of the participants had no musical education. This was based on the fact that a person with a background in music theory and practice might be more 'sensible' regarding expressiveness detection.

From all 30 participants, 60% were male and 40% were female (Figure 6.3) of ages between 16 and 51 years old (Figure 6.4) - resulting in the average age of 25.

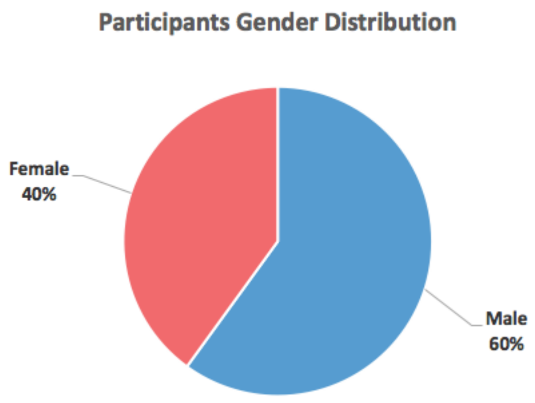


Figure 6.3: Participants gender distribution (in percentage)

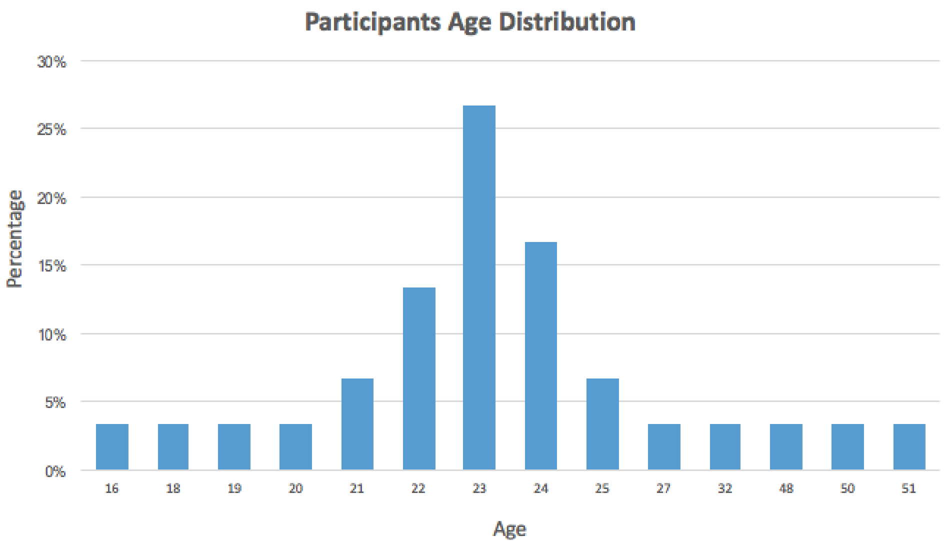


Figure 6.4: Participants age distribution (in percentage)

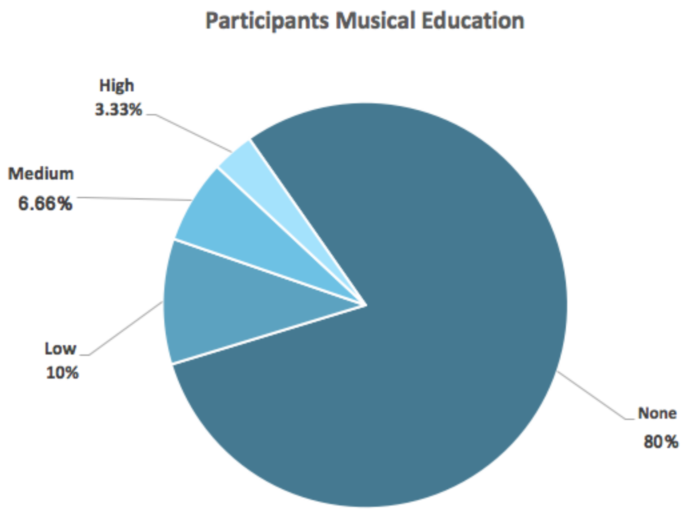


Figure 6.5: Participants musical education (in percentage)

Regarding prior musical education, we separated into four levels for classification purposes: Low (less than one year of formal music education), Medium (from one to three years), High (three years or more) or None. In this categorization, we didn't count mandatory music lessons (i.e. school lessons) as a formal musical education. Of all the 30 participants three had a Low level of musical education, two participants had a Medium level and only one had more than three years of formal musical education. This means that 80% of the participants had no musical education of any kind, as observed in Figure 6.5. Despite the musical background or musical education level, all of the participants were asked to rate the excerpts according to their tastes and opinions only, with no technical terms being mentioned.

6.3 Procedures

The tests were designed in the form of a questionnaire. Each questionnaire was composed of 20 excerpts (two per parameter) with the purpose of ranking them. By having two excerpts per parameter instead of only one we intended to obtain a more clear indication of the participants' rates - for instance, if one rated a certain excerpt with a high value but after that rated another from the same parameter with low values, then it would suggest that that parameter rating should be somewhere between those values. We considered that this approach would increase the reliability of the ratings. Each question (i.e. excerpt) was rated within a range of 1 (*"I don't think this is a good excerpt in any way"*) to 5 (*"I really think this is a good excerpt"*) - a *Likert*-type scale - according to each participant's judgment. The questionnaire was programmed in order to introduce some randomness to it, meaning this that each time a new questionnaire was generated the excerpts were randomly chosen - also avoiding the presence of the same file twice in the same test.

As we considered important that all tests were performed under the same circumstances we chose to conduct them personally instead of making an online-form, providing the same environmental conditions and devices to the participants. This meant assuring that the tests were conducted in a quiet atmosphere and through all tests the exact same sound device was used (portable speaker). Each test session had no time limit, meaning that each participant had no time restriction to perform the tests - also to listen to the excerpts the desired amount of times and in no particular order. Each user-test lasted for about 13 minutes (average estimation). No context regarding the aim of the tests was given to the participants. The participants were only asked to evaluate each excerpt according to what they considered a good or a bad piece and to not to take into account the beginnings and ends of the excerpts as they were part of a greater composition. Additionally, at the beginning of each session, the participants were asked to sign a consent form in which we assure their anonymity in order to use the collected data. File 17 below provides two expressive and two non-expressive excerpts (from artificial and human compositions) as an audible example for comparison purposes. The full set of excerpts used can be found in Appendix C (9).

File 17: User-Test Excerpts Examples (Expressiveness vs Non-expressiveness)
url: https://soundcloud.com/josemariasimoes/sets/user-test-excerpts-examples/s-uvn1j

6.4 Results

After all user-testing sessions, each set of answers was added and combined to evaluate the score of each parameter. As each excerpt could be rated from 1 to 5 and once there were two excerpts per parameter, the maximum score would be 10 in each questionnaire - resulting in a final maximum score of 300 considering all questionnaires. The final scores were converted to percentages. Figure 6.6 below displays a graphical representation of the final scores regarding all parameters.
To ease the reading process and analysis, we list below each parameter acronym and its meaning (note that by 'human composition' we mean *Bach* composition):

- **HOO** - Human composition, no expressiveness.
- **COO** - Computer-generated composition, no expressiveness.
- **HCO** -Human composition, computer-generated velocities and no durations (mean).
- **HHO** -Human composition, live performance velocities and no durations (mean).
- **CCO** - Computer-generated composition and velocities, no durations (mean).
- **HHH** - Human composition, live performance (no modifications).

- **CCC** - Computer-generated composition and expressiveness.
- **HOC** - Human composition, no velocities (mean) and computer-generated durations.
- **HOH** - Human composition, no velocities (mean) and live performance durations.
- **COC** - Computer-generated composition, no velocities (mean) and computer-generated durations.

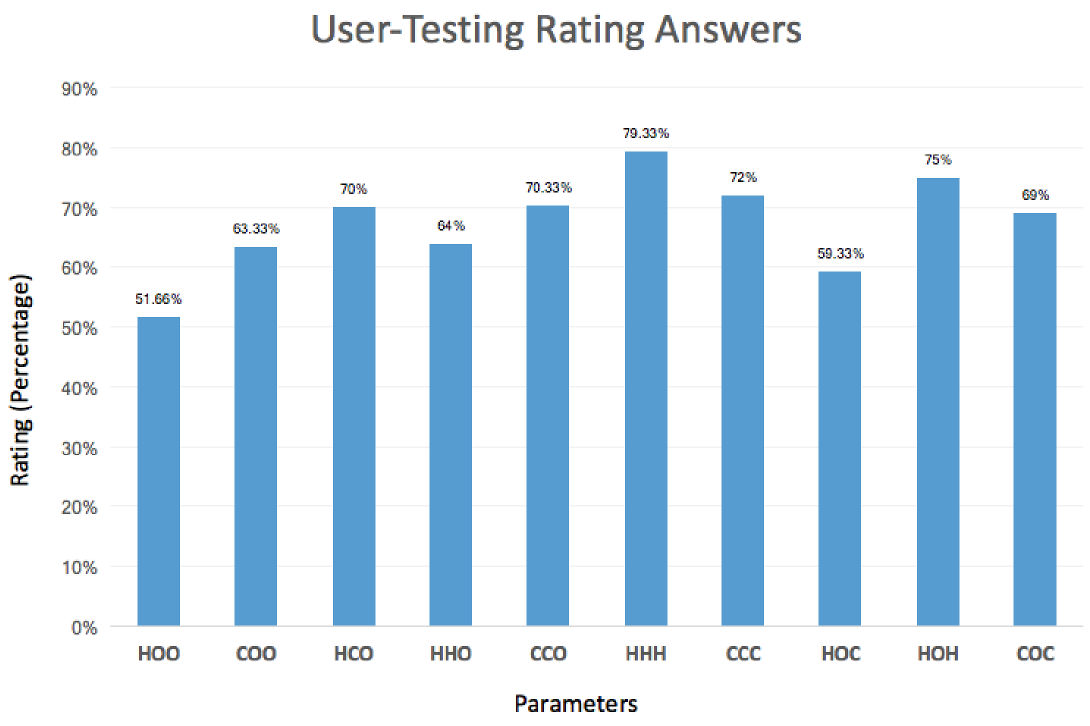


Figure 6.6: User-Testing results

Results presented in Figure 6.6 will be analyzed in more detail through simpler and distributed graphical representations. These examinations will allow us to address each goal separately. We begin by observing how expressive elements influenced the ratings in human compositions.

The histogram below (Figure 6.7) shows that the excerpts with no expressiveness got a total score of 51,66%. By introducing velocities, we identify an increase of 12,34%. In the case of durations, when these assume some variation the score increases by 23,34% - suggesting that durations have more impact on the compositions, at least on this excerpts. When combining both expressive elements, the full expressive excerpts (HHH) assume an increase of 27,67% when compared to the non-expressive parameter, reaching a score of 79,33%. In a global overview, these results suggest that participants favoured the excerpts containing partial or full expressiveness. We proceed to the same comparison, now concerning the excerpts generated by our model.

After verifying that expressiveness holds a positive influence on human compositions, we

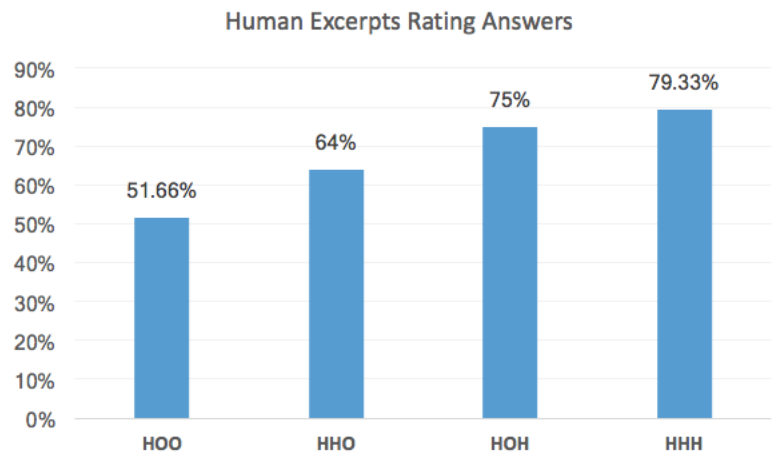


Figure 6.7: Human Excerpts rating answers

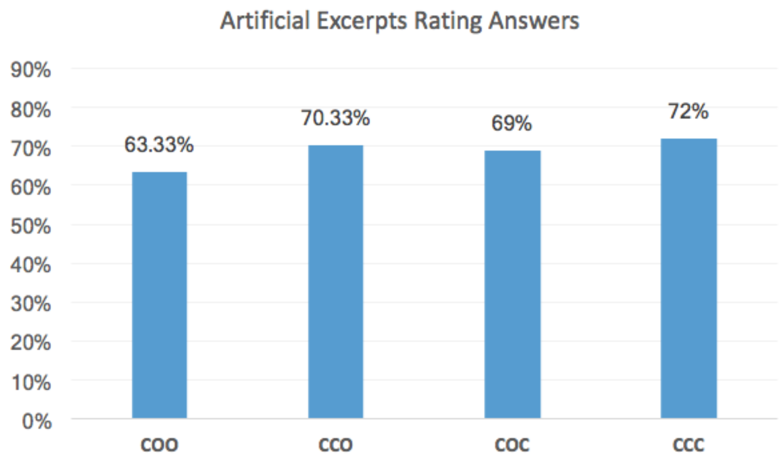


Figure 6.8: Network's Artificial excerpts rating answers

were willing to verify if the same pattern occurs in artificial compositions and expressiveness. Figure 6.8 shows that non-expressive compositions gathered a total score of 63,33%. When velocity expression is added, it provides an increase of 7%. Looking at the third parameter (COC), we observe that in this case the introduction of different durations caused an increase of 5,67%. It is curious to note that in these artificially generated compositions the intensity seems to have more influence than note duration variation - although by a small difference -, the opposite of what was observed on the last histogram. Comparing the final parameter (CCC) to the non-expressive one (COO), we note that full expressiveness has promoted an improvement of 8,67%. In general - and similarly to what we verified in human compositions -, this results also suggest that expressiveness has improved the excerpts.

The last two graphs display indicators of improvement promoted by expressiveness. We compare both non-expressive and expressive ratings regarding human and artificial compositions (Figure 6.9). As we can observe, the artificial non-expressive compositions (COO) had higher ratings than the human ones (HOO). This doesn't imply that the model gen-

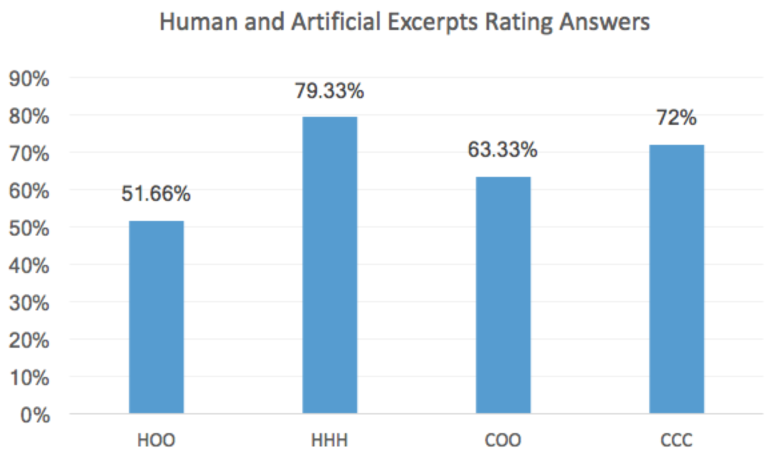


Figure 6.9: Human and Artificial Excerpts rating answers (non-expressiveness and expressiveness comparison)

erates better melodies than *Johann Bach*, but instead it suggests that it has learned to create reasonably good melodies. However, the scenario changes when expressiveness is applied, indicating a solid improvement regarding human compositions and surpassing artificial expressive excerpts. Although the generation of compositions as good as humans isn't the purpose of this work, it was important to have melodic sequences as good as training examples to fairly evaluate the impact of expressive elements.

Lastly, we analyze the experimental setup of combining human compositions with artificial expressive elements. Figure 6.10 presents the partial ratings concerning these parameters compared to human expressive elements. Regarding velocity, we notice that the artificially generated sequences provided good results in comparison to the live performed version. However, it is important to note that in a previous analysis (Figure 6.7) durations have shown to be more influential than note intensity. The same isn't verified when it comes to artificial durations applied to human compositions, where the participants usually preferred the original durations applied.

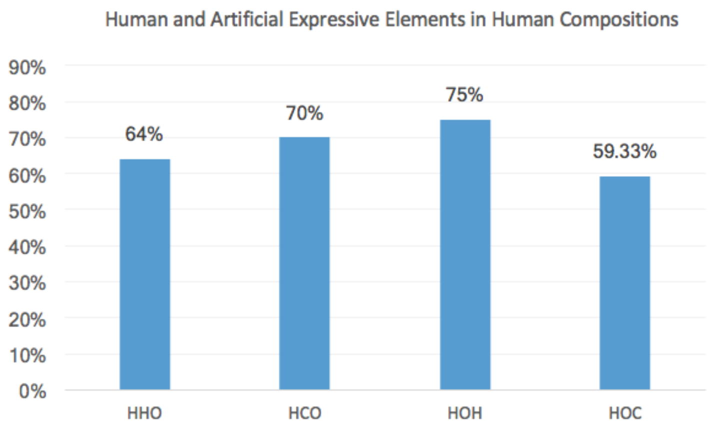
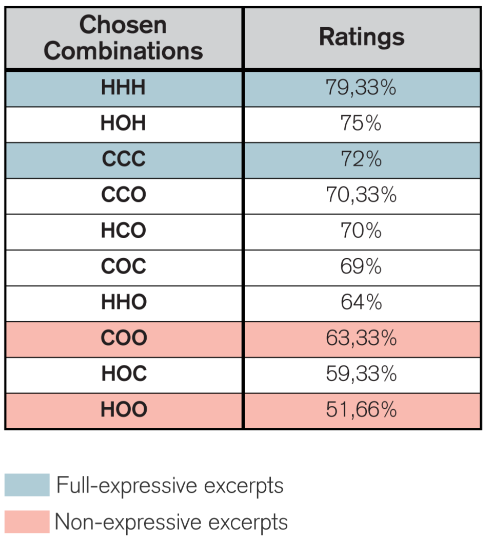


Figure 6.10: Human and Artificial expressive elements in human compositions



Chosen Combinations	Ratings
HHH	79,33%
HOH	75%
CCC	72%
CCO	70,33%
HCO	70%
COC	69%
HHO	64%
COO	63,33%
HOC	59,33%
HOO	51,66%

🟦 Full-expressive excerpts
🟥 Non-expressive excerpts

Figure 6.11: All parameters ratings (ranked)

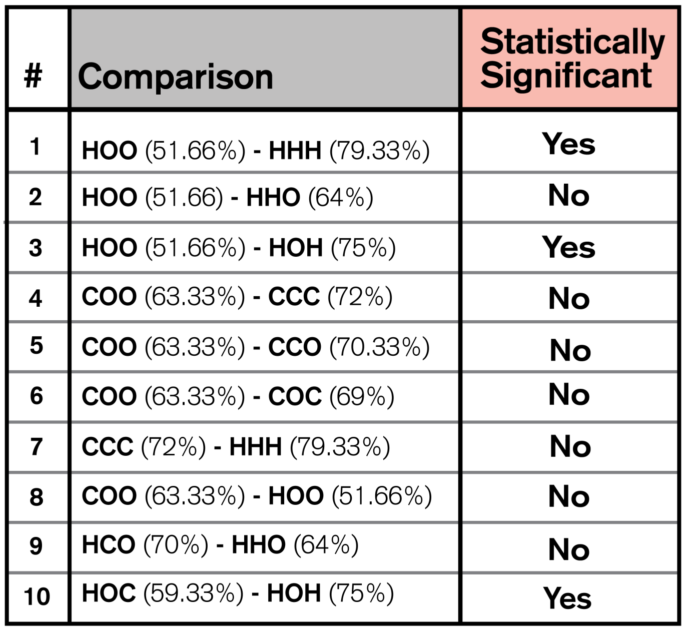
Observing all the given scores to each parameter, there's a suggestion that different combinations have an influence over the excerpts - and also that adding expressive elements (alone or simultaneously) tend to increase the value concerning the participant's preferences. Figure 6.11 presents the ratings of all the parameters, ranked from the highest score to the lowest one. The presented differences and comparisons in this section will be discussed in further detail in the next chapter, as well as global thoughts on the conducted work and subsequent results.

6.5 Statistical Analysis

The observed ratings provide good indicators regarding the impact expressiveness has on either artificial or human compositions. Taking into account that these results were obtained from a representative population sample (30 participants), we decided to perform statistical analysis on the gathered data (i.e. inferential statistics) in order to study possible further interpretations of the results.

We first performed a *Kolmogorov–Smirnov* test for each variable (i.e. parameter) to assess the distribution of our data. For that, we used our raw data (the actual scores and not percentages). A significance value of 5% (i.e. $\alpha = 0.05$) was used to test all variables. One variable differed significantly from that which is normally distributed, meaning that we couldn't assume that all data followed a normal distribution. With this, we had to perform non-parametric tests. To examine if there were statistically significant differences among the variables we performed a *Kruskal-Wallis* test. Using the same significance value, we obtained a critical value of 16,92 and finally a H_{value} of 42,83. Once the H_{value} is higher than the critical value, we were able to reject the null hypothesis, meaning that we determined that there were statistically significant differences among the different variables.

Acknowledging this, we then compared the variables of our interest in groups of two to study whether their differences could be said to be statistically significant. For that, we used the *Mann-Whitney U test* to compare the variables. Again, we used the same significance value of $\alpha = 0.05$ but applying a *Bonferroni* correction - once we performed 10 comparisons, we ended up with $\alpha = 0.005$. The figure below depicts the outcome of all *Mann-Whitney* comparisons (we included the percentages of the different parameters to ease the interpretation).

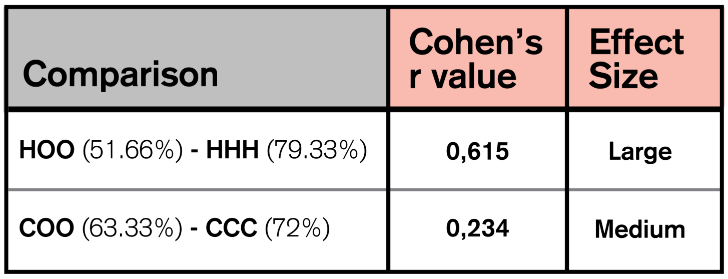


#	Comparison	Statistically Significant
1	**HOO** (51.66%) **- HHH** (79.33%)	**Yes**
2	**HOO** (51.66) **- HHO** (64%)	**No**
3	**HOO** (51.66%) **- HOH** (75%)	**Yes**
4	**COO** (63.33%) **- CCC** (72%)	**No**
5	**COO** (63.33%) **- CCO** (70.33%)	**No**
6	**COO** (63.33%) **- COC** (69%)	**No**
7	**CCC** (72%) **- HHH** (79.33%)	**No**
8	**COO** (63.33%) **- HOO** (51.66%)	**No**
9	**HCO** (70%) **- HHO** (64%)	**No**
10	**HOC** (59.33%) **- HOH** (75%)	**Yes**

Figure 6.12: *Mann-Whitney U test* results

We acknowledge that only three differences were pointed as statistically significant. However, not all comparisons described as not statistically significant were negative premises. For instance, looking at comparison 7, once we were comparing live human performances composed by *Bach* with compositions generated by our model the results point that there is no significant difference between those ratings, suggesting that although the participants preferred human performances the artificial expressive excerpts were statistically close to those ratings. On the other hand, *Mann-Whitney* tests shows that there are statistically significant differences between expressive and non-expressive artificial excerpts. Observing the same circumstance regarding human excerpts, we conclude that adding velocities (HOO - HHO) had no significant difference whereas adding durations and both expressive elements had.

Although statistical significance determines if an effect exists probabilistically (in our case, whether the difference between the scores was significant) and also if the observed difference was due to chance, this doesn't reveal the size of the effect itself [59]. Comprehending that results may vary depending on the population sample size, we considered important to test the effect size on variables with expressiveness compared to non-expressive ones. Considering that we were dealing with non-parametric data - and performed *Mann-Whitney* tests - we calculated the effect size between variables using *Cohen's r* value. Then, we analyzed the effect size regarding full expressiveness in human and artificial excerpts. As we observed, the difference between non-expressive and expressive human excerpts was determined as statistically significant, but the same wasn't verified considering artificial excerpts. Figure 6.13 below presents the effect size on expressiveness comparison. The effect measure was obtained by following *Cohen's* guidelines for *r* [30].



Comparison	Cohen's r value	Effect Size
HOO (51.66%) **- HHH** (79.33%)	**0,615**	**Large**
COO (63.33%) **- CCC** (72%)	**0,234**	**Medium**

Figure 6.13: *Cohen's r* value results (Effect Size)

We observe that adding all expressive elements to human excerpts had a large effect size, meaning that there's a substantial impact when full expressiveness is introduced. Regarding artificial excerpts, we identified a medium effect size by adding expressiveness to the excerpts. The effect size measurements provide some interesting insights, especially in the case of artificial excerpts (COO-CCC). These results suggest that although the difference between these two variables isn't statistically significant, the effect size shows that there's a considerable improvement when full expressiveness is added. This indicates that there is an impact differentiating these two variables, although a deeper and more detailed study is required to assess this fact at a different scale, maybe with a larger population sample.

Chapter 7

Discussion and Future Work

Examining the ratings obtained on the user-testing phase we realized that the gathered results provided some useful insights regarding the study of artificial music expressiveness. The ratings exhibit a tendency to increase the preference as expressive elements are added to a music composition - whether these elements are added individually or collectively. This is verified not only in human performances and compositions - where the full expressive compositions provided an improvement of 27,67% over the participants' preferences - but also in the excerpts generated by our model with a global increase of 8,67%. These results come to reinforce findings and suggest that expressiveness is an important aspect of a music composition as it has the capability to enhance the musical output at a perceptual level, allowing participants to distinguish the multiple expressions that music can take in a natural and effortless way. Comparing the artificial non-expressive compositions with expressive ones we observe that in some cases - especially when the generated piece holds a sequence of stacking notes - expressiveness plays a major role in making an artificial composition to sound more natural, closer to a human one.

Another key aspect to evaluate concerns the generated melodies (i.e. notes only). To properly study expressiveness the model must be capable of generating sufficiently good note sequences (i.e. good enough so it won't compromise the ratings due to inconsistency our severe dissonance). The validation of this point was done by comparing the artificial excerpts to human ones (i.e. *Johann Bach's*), where the absence of expressiveness (HOO *vs* COO) indicates that the network has learned to generate coherent melodies. Although the non-expressive artificial melodies got a higher rating comparing to non-expressive human ones, the same doesn't apply when all expressiveness is introduced. It is important to note that this doesn't necessarily mean that the chosen artificial melodies are better than the human ones. In fact, it is likely to happen due to the exact opposite: *Bach* pieces can be more complex in its structure than the compositions generated by our model. The human excerpts chosen contain some variation regarding note positions, which creates some moments of silence between notes or passages where a lot of notes are played in a short amount of time. Our model wasn't trained to learn this note 'allocations', meaning that in each generated composition the notes are positioned according to a generic value, making all notes equally distanced. The variations in *Bach* pieces regarding this factor may have contributed to a cruder sounding in non-expressive compositions than the artificial excerpts where all notes were metrically and equally sequenced.

Looking at the parameters covering artificial expression applied to human compositions (HCO and HOC) with the original ones (HHO and HOH), we state that artificial velocities surpassed the human's live recorded velocities. The same isn't verified in artificial durations, where the original ones were preferred - with a difference of 15,67%. Joining this with the results observed in Figure 6.7, it may suggest that durations are more likely to hold major influence over the composition, not only by providing variation but also how specific values affect the composition (artificial durations haven't surpassed original ones). Considering statistical analysis results, there are some interesting points to address. The improvements on the ratings when comparing non-expressive human compositions with live performances were identified as statistically significant, having a large effect size. This indicates that expressiveness may hold an important role on the listener's preferences. On the other hand, the difference verified in artificial pieces (COO *vs* CCC) wasn't statistically significant according to the same terms. However, the effect size indicated that this difference had impact to a certain level that should be taken into consideration. Finally, the differences between expressive human pieces and artificial ones resulted in no statistical significance, which suggests that the expressive pieces produced by our model were as preferable as human live performances.

Nonetheless, there are some points that we consider important to address in future work. First, the quality of the full generated compositions is an aspect that needs improvements. As we presented, the pieces generated by our model hold quite interesting melodies and sequences, but when considering the full composition we observed some lack of coherence and some sporadic monotony. Being a topic that has already been pointed out by several authors, we agree that it is probably one of the main issues to overcome regarding artificial music. In addition, we estimate that another important factor to improve these musical pieces is to train the network (or model) to learn how to begin and end a composition. The absence of this notion - as seen in our generated compositions - makes the composition less natural to our ears by suddenly ending without any kind of change of 'announcement', which may sound more mechanical and less aware of the full structure - and thus far from human capacities. Nevertheless, it would be interesting in future work to train the model to learn how to position notes with some degree of variance and not equally distanced and test how this factor influences the piece. Another feature that would be interesting to study is making the networks correlated in their learning and generation process instead of having a model of independent networks like ours. We consider that this could be a good way to experiment whether artificial networks find compositional patterns by being dependent - for instance, generating durations based on the pre-generated note sequences. Establishing a good 'plagiarism' or similarity metric regarding artificial compositions could also be a good approach in a way that it might ease the process of identifying possible overfitting cases - like earlier referred, although we want the model to learn musical structures and composition patterns from *Bach*, we still want it to be able to produce its own pieces. This similarity identification is not as straightforward as it might seem, as it logically differs from composition to composition, and defining a minimum number of note sequences might not be trivial, once it is hard to tell - at least in a global or universal manner - to which extent a piece is actually similar to another. It's not hard to tell whether a composition is an exact copy of another or if it is entirely different but identifying if eventual similarities influence the piece can be challenging.

Given that all the architectures used haven't reached a clear state of stabilization, more training time would possibly produce better results. Although we consider having good results when looking at the available computational resources and time-frame, having a larger dataset or more powerful architecture would theoretically improve the compositions. On the other hand, it is also important to note that regardless of the good gathered results we have no proof that the model used is the best one to address this problem. It could be valuable to test our process with different models by exploring other architectures. Lastly, we must keep in mind the size of the user-testing sample. As we described, we preferred having local control of the procedures. An online questionnaire would probably increase the number of participants, but that way we would have no control over the environment. Having that said, it could be beneficial to test the different parameters on a larger scale.

As we referred earlier in this document, only a few attempts of generating expressive artificial music has been performed. The belief in the importance of expressiveness in compositions led us to examine this specific question to assess its relevance. Results suggest that this is indeed a factor that should be explored in order to improve computer-generated music. In sum, we believe that the present work has positively contributed to the field of artificial music generation by studying how a factor like expressiveness - that is usually despised - may perceptually influence musical compositions, providing some good indicators of how much a music piece is affected by it.

Chapter 8

Work Plan

The present work is intended to be complete within an academic year (from September 2018 to July 2019). The required tasks were divided into two work plans - first and second semester. The addressed tasks and effort regarding the work plan for this first semester is illustrated by Figure 5.1. The first step of our work consisted of gathering and analyzing research on the following topics: artificial musical composition, neurofeedback data extraction and music psychology (i.e. the impact of music in humans). As we faced some constraints regarding the neurofeedback device we've decided to prioritize the third task in order to evaluate the feasibility of this approach, which finally led us to restructure this thesis' goals (see Appendix A). In a further stage, we've studied the architecture of some of the most common DL networks for music generation (mainly LSTMs), as well as frameworks and libraries to be used (e.g. Keras [50]). The first tests on the neural network were performed while studying research papers on this topic in order to draw some conclusions on what might be the best approaches concerning some relevant components (e.g. data representation). Both tasks were performed simultaneously once each test case (i.e. network training phase) took at least two days under a relatively small size data set (not more than 30 files). The last task of the mentioned semester was to write part of this document, beginning in December.

The work plan for the second semester was divided into six tasks, presented by the Gantt chart in Figure 5.2, in which we describe the estimated time required for each task and the actual time we needed (in weeks) to perform them. During the first half of the second semester we developed our prototype according to the defined goals and based on notes taken from preliminary results (Chapter 4). Although this was an estimate - due to the uncertainty of the training time needed - we have concluded the conception of the prototype in 12 weeks (one week more than the initially predicted). The user-testing phase took three weeks and it involved all of the procedures: from the excerpts preparation and test design to the tests in person. Part of the results analysis were made during the user-testing phase as the results were gradually added to a single file - in which the graphs of the results were generated. We observed all the results to assess whether they met the defined goals. As we have explained in the last chapters of this document, we have observed good indicators regarding the tests' results. The previously defined 'Refinement' phase was left aside not only because our model have met the desired goals but also because the dissertation writing was a priority. By the end of May, we have decided to submit a paper based in our work to ARTECH 2019 – *Digital Media Art Ecosystems. 9th International Conference on Digital and Interactive Arts.*

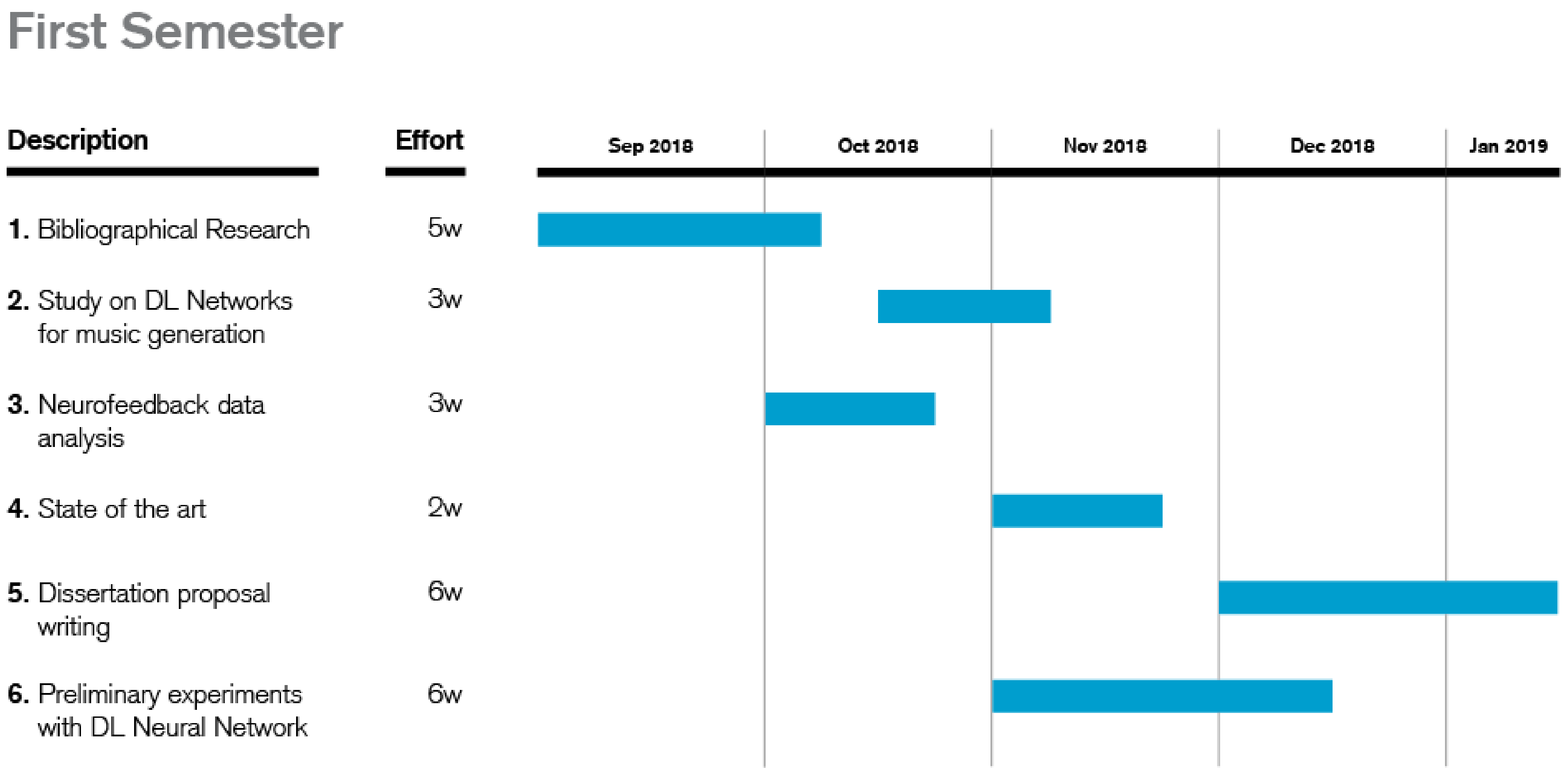


Figure 8.1: Gantt Chart: First Semester.

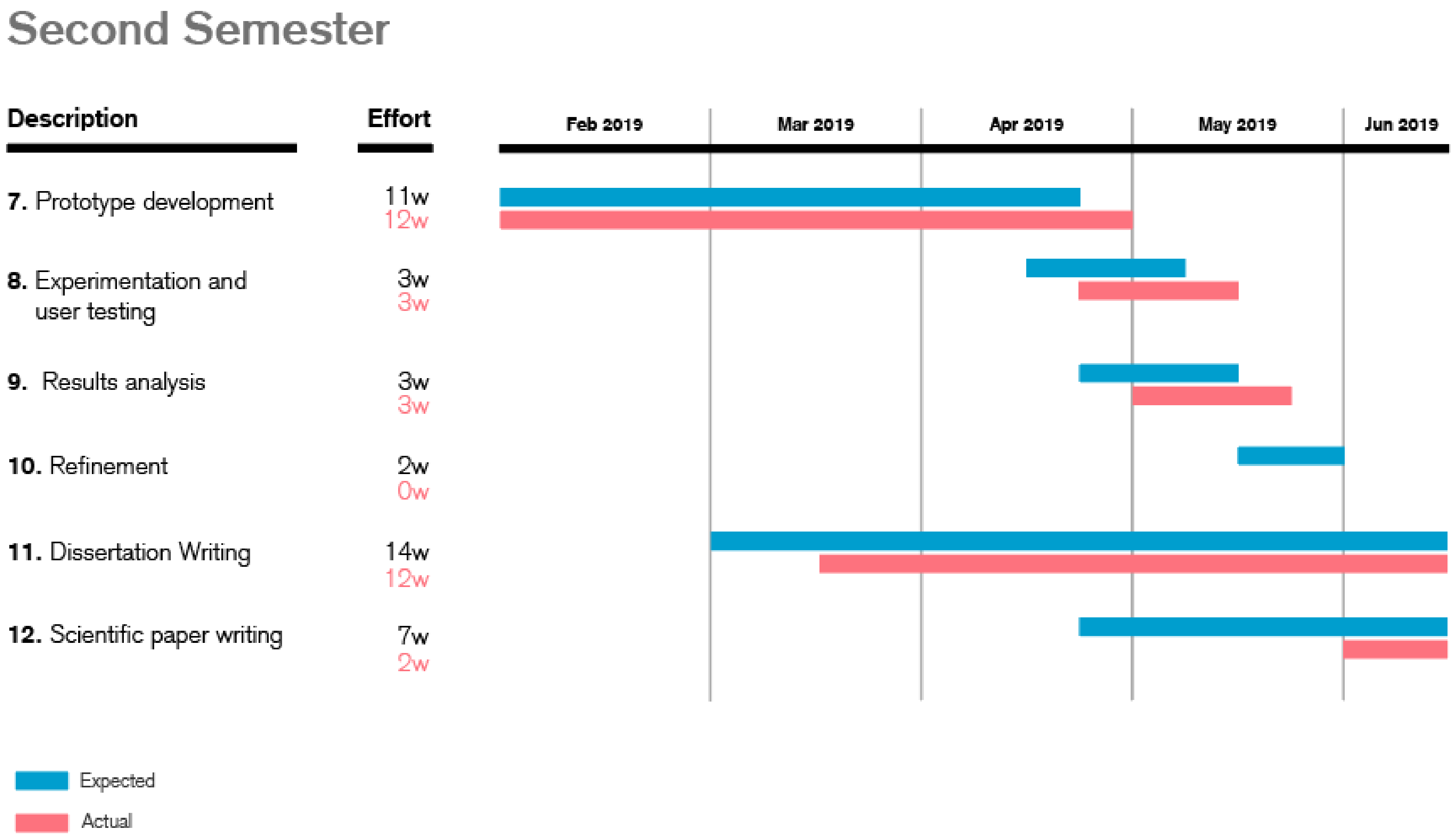


Figure 8.2: Gantt Chart: Second Semester (Expected vs Actual).

Chapter 9

Conclusion

In the last decade, DL algorithms have been proving their efficiency in some unconventional fields such as content generation. This extension reaches different art forms, particularly images and music. The application of these algorithms in image processing and generation has achieved remarkably good results, whereas artificial music generation has been evolving at a much lower rate. Most of the experiments performed reported good results but failed to generate musical compositions close to human expressiveness. Furthermore, there were only a few experiments considering more musical elements than notes. Not despising the undeniable importance of notes in a music composition, we intended to explore more factors that could eventually influence a composition. With that, our work was based on exploring expressive variations - regarding velocity and duration -, with the main goal of creating a model capable of generating expressive musical compositions. In addition, we also had the goal of testing their influence over human and artificial compositions. Our contributions to the field of artificial music generation comprise a model capable of generating melodies adding expressive elements to them and a set of statistical indicators suggesting that expressiveness improves music pieces.

The work was conducted during two semesters, being the first mainly reserved for research purposes and developing the state of the art, where we had the need for restructuring the initial scope. This study was important for us to better understand different approaches and experiments concerning artificial music generation. It became clear that there wasn't, in fact, an indicator of how much expressiveness can influence artificial compositions. Also, this initial research was determinant when choosing the architecture - LSTM - and also to assume the need for building our own model. Although knowing this beforehand, this initial research reinforced that training ANN is a time-consuming process, which led us to anticipate some tasks. Preliminary work results have shown promising results considering what we were aiming for with this project, providing good baselines regarding future model - not only the architecture itself but also important features such as representation and dataset details.

On the second semester, we introduced expressive elements to our practical work and had the need of handling both separately (i.e. notes, velocities and durations), defining a prototype model consisting of three independent networks. After a two-month training phase and adjustments, the first artificial expressive compositions were generated and evaluated, having identified a good element variance within the compositions. We then developed a user-testing questionnaire as a validation process, through which we could examine several parameters combinations. Results demonstrate that expressive pieces were preferred when compared to non-expressive ones, either in human compositions or artificial generated by our model. Although statistical analysis indicated that some differences couldn't be seen as statistically significant, it showed that expressiveness had, in fact, an effect over the pieces and that expressive excerpts generated by our model were rated as good as human live performances with no statistically significant difference.

Reflecting upon all the work performed during this academic year, all the defined goals were achieved. We built a model consisting of three independent artificial LSTM networks capable of generating coherent and original melodies. In addition, our model generates expressive elements that can be added to the melody in order to provide expression to it. We also attended our secondary goals by replacing human expressiveness with artificial ones. All of these factors were tested not only to assess their value - particularly the artificial compositions - but also to assess the aim of this work and observing if expressiveness has any kind of impact on the compositions (human or artificial). The obtained results through thorough comparisons suggest that expressive elements have influential impact over the compositions (either human or artificial), providing better improvements when combined.

With this, we consider to have conducted a good work and gathered positive results. In addition, we recognize that different models and architectures should be tested in order to get deeper understandings on this matter. Nevertheless, our model has proved to be efficient enough to test and accomplish the desired goals. More importantly, we consider that the results presented in this document are useful indicators to conclude that expressiveness must be taken into account when generating artificial music in a plausible way for improving pieces. We hope our work promotes future discussions on this and inspires further research on the topic.

References

[1] Google brain magenta. `https://magenta.tensorflow.org`. [Online; accessed 04-Dec-2018].

[2] International e-piano competition. `http://www.piano-e-competition.com/`. [Online; accessed 24-May-2019].

[3] Keras documentation on lstms. `https://keras.io/layers/recurrent/`. [Online; accessed 17-Nov-2018].

[4] Magenta's melody rnn model. `https://github.com/tensorflow/magenta/tree/master/magenta/models/melody_rnn`. [Online; accessed 23-May-2019].

[5] Magenta's performance rnn model. `https://github.com/tensorflow/magenta/tree/master/magenta/models/performance_rnn`. [Online; accessed 23-May-2019].

[6] Magenta's polyphony rnn model. `https://github.com/tensorflow/magenta/tree/master/magenta/models/polyphony_rnn`. [Online; accessed 23-May-2019].

[7] Midi instrument map. `https://www.midi.org/specifications-old/item/gm-level-1-sound-set`. [Online; accessed 27-Dec-2018].

[8] Music21 library. `http://web.mit.edu/music21/`. [Online; accessed 18-Nov-2018].

[9] Open source magenta models. `https://github.com/tensorflow/magenta/tree/master/magenta/models`. [Online; accessed 12-Dec-2018].

[10] Performance rnn: Generating music with expressive timing and dynamics. magenta model. `https://magenta.tensorflow.org/performance-rnn`. [Online; accessed 04-Dec-2018].

[11] Eniola Alese. The curious case of the vanishing & exploding gradient. `https://medium.com/learn-love-ai/the-curious-case-of-the-vanishing-exploding-gradient-bf58ec6822eb`, 2018. [Online; accessed 20-Dec-2018].

[12] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. In *IEEE Transactions of Neural Networks, Vol. 5, No. 2*, 1994.

[13] Axel Berndt and Tilo Hähnel. Expressive musical timing. In *Proceedings of Audio Mostly 2009 - A Conference on Interaction with Sound*, 2009.

[14] Filippo Maria Bianchi, Enrico Maiorino, Michael Kampffmeyer, Antonello Rizzi, and Robert Jenssen. An overview and comparative analysis of recurrent neural networks for short term load forecasting. 2017.

[15] Margaret A. Boden. What is creativity? In *Dimensions of Creativity*, page 75–118. MIT Press, 1996.

[16] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. Deep learning techniques for music generation. Sorbonne Université, UPMC Univ Paris 06, CNRS, LIP6, Paris, France, 2017.

[17] Vitaly Bushaev. Understanding rmsprop — faster neural network learning. `https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a`, 2018. [Online; accessed 03-May-2019].

[18] Filippo Carnovalini and Antonio Roda. A multilayered approach to automatic music generation and expressive performance. In *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*, page 41–48, 2019.

[19] Keunwoo Choi, George Fazekas, and Mark Sandler. Text-based lstm networks for automatic music composition. 2016.

[20] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Y Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. 2014.

[21] Uniqtech Co. Understand the softmax function in minutes. `https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d`, 2018. [Online; accessed 03-Jan-2018].

[22] Yann Le Cun. A theoretical framework for back-propagation. In *Connectionist Models Summer School*, pages 21–28, 1988.

[23] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. *Association for the Advancement of Artificial Intelligence (www.aaai.org)*, 2018.

[24] Hao-Wen Dong and Yi-Hsuan Yang. Convolutional generative adversarial networks with binary neurons for polyphonic music generation. *19th International Society for Music Information Retrieval Conference, Paris, France*, 2018.

[25] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. In *Neural Computation, 18(7):1527–1554*, Cambridge, MA, USA, 2006.

[26] Douglas Eck and Jasmin Lapalme. Learning musical structure directly from sequences of music. 2008.

[27] Douglas Eck and Jürgen Schmidhuber. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *Neural Computation, 18(7):1527–1554*, IDSIA Istituto Dalle Molle di Studi sull'Intelligenza Artificiale Galleria 2, 6928 Manno, Switzerland., 2002.

[28] Ashkan Fakhr Tabatabaie, Mohammad Azadehfar, Negin Mirian, Maryam Noroozian, Ali Yoonessi, and Mohammad Saebipour. Neural correlates of boredom in music perception. In *Basic and Clinical Neuroscience. Vol 5.*, pages 259–266, 2014.

[29] Pere Ferrera and Josep Puyol-Gruart. Ontomusic: from scores to expressive music performances. In *Frontiers in Artificial Intelligence and Applications, 131*, pages 141–148, 2005.

[30] Catherine Fritz, Peter E Morris, and Jennifer J Richler. Effect size estimates: Current use, calculations, and interpretation. In *Journal of experimental psychology. General. 141*, pages 2–18, 2011.

[31] Richard H. Hall. The neuron. 1998.

[32] Waldie E. Hanser and Ruth E. Mark. Music influences ratings of the affect of visual stimuli. In *Psychological Topics 22*, pages 305–324, 2013.

[33] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[34] Allen Huang and Raymond Wu. Deep learning for music. 2016.

[35] Stanley Jacobson and Elliott M. Marcus. *Neuroanatomy for the Neuroscientist (Chapter 1 - Introduction to the Central Nervous System).* Springer, Boston, MA, 2008.

[36] Anil K Jain, Jianchang Mao, and K.M Mohiuddin. Artificial neural networks: A tutorial. *Michigan State university*, 1996.

[37] Patrik Juslin. From everyday emotions to aesthetic emotions: Toward a unified theory of musical emotions. *Physics of life reviews*, 10, 2013.

[38] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks. `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`, 2015. [Online; accessed 15-Oct-2018].

[39] Nikhil Kotecha and Paul Young. Generating music using an lstm network. *arXiv preprint arXiv:1804.07300*, 2018.

[40] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. In *Nature, Vol 521*, 2015.

[41] Feynman Liang, Mark Gotham, Matthew Johnson, and Jamie Shotton. Automatic stylistic composition of bach chorales with deep lstm. 2017.

[42] Zachary C. Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. 2015.

[43] Kyle McDonald. Neural nets for generating music. `https://medium.com/artists-and-machine-intelligence/neural-nets-for-generating-music-f46dffac21c0`, 2017. [Online; accessed 04-Oct-2018].

[44] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, 2010.

[45] Tianya Mingyue. Tensorflow application combat -7-tools and network model. `https://www.jianshu.com/p/6c5927e28e36`, 2018. [Online; accessed 03-Nov-2018].

[46] Aran Nayebi and Matt Vitelli. Gruv: Algorithmic music generation using recurrent neural networks. 2015.

[47] Gerhard Nierhaus. *Algorithmic Composition - Paradigms of Automated Music Generation.* Springer, 2009.

[48] Christopher Olah. Understanding lstm networks. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2017. [Online; accessed 21-Dec-2018].

[49] George Papadopoulos and Geraint Wiggins. Ai methods for algorithmic composition: A critical survey. *School of Artificial Intelligence, Division of Informatics, University of Edinburgh 80 South Bridge, Edinburgh, EH1 1HN, Scotland*, 1999.

[50] Plunkett and Elman. Exercises in rethinking innateness, mit press, 1997. (excerpt). `http://www.cse.unsw.edu.au/~billw/cs9444/crossentropy.html`. [Online; accessed 18-Dec-2018].

[51] Andy M. Sarroff and Michael Casey. Musical audio synthesis using autoencoding neural nets. 2014.

[52] Jürgen Schmidhuber, Daan Wierstra, Matteo Gagliolo, and Faustino Gomez. Training recurrent networks by evolino. *Neural Comput.*, 19(3):757–779, March 2007.

[53] Catherine Schmidt-Jones and Russell Jones. *Understanding Basic Music Theory.* Connexions, Rice University, Houson, Texas, 2007.

[54] Sagar Sharma. Activation functions: Neural networks. `https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6`, 2017. [Online; accessed 28-Dec-2018].

[55] Vidushi Sharma, Sachin Rai, and Anurag Dev. A comprehensive study of artificial neural networks. In *International Journal of Advanced Research in Computer Science and Software Engineering*, pages 278–284, 2012.

[56] Sirena. Optimizers for training neural networks. https://medium.com/datadriveninvestor/optimizers-for-training-neural-networks-e0196662e21e, 2018. [Online; accessed 03-May-2019].

[57] Sigurður Skúli. How to generate music using a lstm neural network in keras. https://towardsdatascience.com/how-to-generate-music-using-a-lstm-neural-network-in-keras-68786834d4c5, 2017. [Online; accessed 03-Nov-2018].

[58] David Stutz. *Understanding Convolutional Neural Networks.* Fakultät für Mathematik, Informatik und Naturwissenschaften - Seminar Report (Prof. Dr. Bastian Leibe), 2014.

[59] Gail M Sullivan and Richard Feinn. Using effect size-or why the p value is not enough. In *Journal of graduate medical education, 4*, pages 279–82, 2012.

[60] William Thompson and Lena Quinto. Music and emotion: Psychological considerations. In *The Aesthetic Mind: Philosophy and Psychology. 10*, pages 3–22, 2012.

[61] Tonalsoft. Midi note number, frequency table. http://www.tonalsoft.com/pub/news/pitch-bend.aspx, 2005. [Online; accessed 05-Jan-2018].

[62] Avinash Sharma V. Understanding activation functions in neural networks. https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0, 2017. [Online; accessed 09-Dec-2018].

[63] Iannis Xenakis. *Formalized Music: Thought and Mathematics in Composition.* Pendragon Press, Stuyvesant, New York, 1995.

[64] Guoqiang Zhong, Li-Na Wang, Xiao Ling, and Junyu Dong. An overview on data representation learning: From traditional feature learning to recent deep learning. In *The Journal of Finance and Data Science - Volume 2, Issue 4*, pages 265–278, 2016.

Appendices

Appendix A

Scope Redefinition

The initially defined aim of the present thesis was to generate multimedia artefacts through an evolutionary algorithm guided by neurofeedback data - this generated content was intended to induce a certain emotion in a certain user. As we mentioned in the document, this scope and goals were redefined due to some verified constraints. In this appendix, we'll provide a more detailed explanation on why this decision was made regarding the obtained results. Looking at this first plan, we can divide it into two major topics: Neurofeedback data and multimedia artefacts generation.

A few weeks before the beginning of this first semester, our attention focused on the neurofeedback device to be used. After brief research and discussion, we've decided to use the Emotiv Epoc device once it was available for our use and we had indications of how it would work (based on professors experience that had used it before).

The Emotiv Epoc is a Headset containing pairs of electrodes (electrical activity conductors) designed to amplify signals received through brain activity. The headset would be wirelessly connected to a computer, sending this gathered data through time.

To test the device, we've run some experiments using Emotiv BCI [37], a free software provided by its manufacturers. It allows us to create user profiles and access some training sessions so that it can learn (i.e. calibrate) the received impulses according to each persona. This software also provides a graphical representation of the emotional state of the user through time. Unfortunately, the raw neurofeedback data is no longer available in these free software versions, so we had no tangible (i.e. numerical) data to work around with, particularly to use it to train our algorithm.

Some attempts were made in order to gain access to those values, but no one seemed to work efficiently. Either way, we've decided to look for an alternative once this methods weren't officially provided by the manufacturers, which logically compromises the reliability of the collected data.

After that, while searching for alternatives, some time was also spent on gathering information on artefacts (music or images) generated by evolutionary or learning algorithms. This led us to two conclusions that led us to redefine this project's objectives.

Firstly, we've tried a new approach to measure a person's emotion: through facial analysis. The results were good in the sense that a smile was logically (and efficiently) correlated to happiness or any good mood. Still, the issue with this type of analysis is that although non-verbal expressions could be reliable, it can't be compared to neurofeedback impulses. For instance, one probably would show physically disgust or anger when listening to a song he or she doesn't like. On the other hand, one doesn't always smile when listens to an *enjoyable* song. So, although facial recognition could be a plausible alternative it clearly wasn't as straightforward as neurofeedback data.

Second, while researching for work on algorithms that generate music or images we found that there wasn't much deeper analysis on this, especially for musical composition. As we mentioned in the document, most musical pieces generated by algorithms were 'just' piano tracks, and not surprisingly good. Regarding our needs, to induce an emotional state we believe that a musical piece must be close to flawless and that's due to diverse reasons: For instance, an algorithm can learn how to induce anger by simple producing random and unpleasant sounds, which can not be considered music (according to a common pop/rock/jazz song). Another example is that, as we've seen, artificially musical compositions can easily fall into cycles, repeating the same sequence over and over again which would logically saturate the listener.

With this, we stated that musical generation not only wasn't a simplistic domain but also that the generated content must be almost similar to what a human can do in order to induce defined emotions. To this, we add the fact that we're left with no reliable software to measure emotions in an inerrable way. The plan was restructured in a way that we can now focus only on artificial musical generation.

We would like to thank Professor Marco Simões not only for providing us the Emotiv Epoc device but also for sharing his previous work on neurofeedback data extraction. In addition, we express our gratitude to Professor César Teixeira for his availability on helping us trying to overcome these issues.

Appendix B

MIDI files datasets used

The Strokes Dataset

- The Strokes - 12:51
- The Strokes - Alone, Together
- The Strokes - Automatic Stop
- The Strokes - Barely Legal
- The Strokes - Between Love and Hate
- The Strokes - Elephant Song
- The Strokes - Hard to Explain
- The Strokes - I Can't Win
- The Strokes - Is this it?
- The Strokes - Juicebox
- The Strokes - Last Nite
- The Strokes - Meet me in the Bathroom
- The Strokes - New York City Cops
- The Strokes - Soma
- The Strokes - Someday
- The Strokes - Take it or leave it
- The Strokes - The end has no end
- The Strokes - The Modern Age
- The Strokes - Trying you luck
- The Strokes - Under Control
- The Strokes - What ever happened?
- The Strokes - When it started
- The Strokes - You only live once

- The Strokes - You Talk way too much

The Beatles Dataset

- The Beatles - Across the Universe
- The Beatles - A hard day's night
- The Beatles - All I've got to do
- The Beatles - All my Loving
- The Beatles - All you need is Love
- The Beatles - And I Love Her
- The Beatles - Anna
- The Beatles - Ask me why
- The Beatles - Chains
- The Beatles - Cry Baby Cry
- The Beatles - Dizzy Miss Lizzy
- The Beatles - Eleanor
- The Beatles - Golden Slumbers
- The Beatles - Help
- The Beatles - I'm Happy just to Dance with you
- The Beatles - I am the Walrus
- The Beatles - I Want You (She's so heavy)
- The Beatles - Let it Be
- The Beatles - Love me Do
- The Beatles - Misery
- The Beatles - Oh Darling
- The Beatles - Please Please Me
- The Beatles - P.S. I Love you
- The Beatles - Sexy Sadie
- The Beatles - Something

- The Beatles - Strawberry Fields Forever
- The Beatles - Sunking
- The Beatles - Tell Me what you see
- The Beatles - Tell me Why
- The Beatles - Ticket to Ride
- The Beatles - Tomorrow Never Knows
- The Beatles - Twist and Shout
- The Beatles - Why Don't We
- The Beatles - Yesterday
- The Beatles - You've got to hide your love away

Jazz Various Artists Dataset

- A Foggy Day - Gershwin
- A Night in Tunisia - Dizzy Gillespie
- African Flower - Duke Ellington
- Afro Blue - Mongo Santamaria
- After you've gone - Simone
- Afternoon in Paris - Lewis
- All Blues - Miles Davis
- All of me - Simone
- Black Narcissus - Joe Henderson
- Black Orpheus - Louis Bonfa
- Blue in Green - Miles Davis
- Blue Monk - Thelonious Monk
- Blue Moon - Hart & Rodgers
- Blue Trane - Coltrane
- Bluesette - Toots Tielemans
- C jam blues - Duke Ellington

- Captain Marvel - Chick Corea
- Corcovado - Jobim
- Desafinado - Jobim
- Don't Mean a Thing - Ellington
- Five Hundred Miles High - Chick Corea
- Gentle Rain - Louis Bonfi
- Georgia on My Mind - Hoagy Carmichael
- Giant Steps - John Coltrane
- Girl Talk - Neil Hefti
- Gloria's Step - Bill Evans
- Grooving - Buddy Rich
- Grooving High - Gillespie

Johann Bach Dataset

- Aus meines Herzens Grunde (Chorale 1)
- Ich dank' dir, Heber Herre (Chorale 2)
- Ach Gott, vom Himmel seih' darein (Chorale 3)
- Es ist das Heil uns kommen her (Chorale 4)
- An Wasserflussen Babylon (Chorale 5)
- Christus, der ist mein Leben (Chorale 6)
- Nun lob', mein Seel', den Herren (Chorale 7)
- Freuet euch, ihr Christen alle (Chorale 8)
- Ermuntre dich, mein schwacher Geist (Chorale 9)
- Aus tiefer Not schrei' ich zu dir (Chorale 10)
- Jesu, nun sei gepreiset (Chorale 11)
- Puer natus in Bethlehem (Chorale 12)
- Allein zu dir, Herr Jesu Christ (Chorale 13)
- O Herre Gott, dein gottlich Wort (Chorale 14)

- Christ lag in Todesbanden (Chorale 15)
- Bourrée (Cello Suite Number Four in Eb)
- Allemande (Cello Suite Number Four in Eb)
- Courante (Cello Suite Number One in G)
- Gigue (Cello Suite Number One in G)
- Minuetto 1 (Cello Suite Number One in G)
- Prelude (Cello Suite Number One in G)
- Allemande (Cello Suite Number One in G)
- Sarabande (Cello Suite Number One in G)
- Allemande (1st Partita in B flat major)
- Courante (1st Partita in B flat major)
- Gigue (1st Partita in B flat major)
- Prelude (1st Partita in B flat major)
- Sarabante (1st Partita in B flat major)
- Allemande (2nd Partita in C minor)
- Aria (Goldberg Variations, Individual Variations)
- Brandenburg Concerto Number One
- Brandenburg Concerto Number Two
- Brandenburg Concerto Number Three
- Brandenburg Concerto Number Four, First Movement
- Brandenburg Concerto Number Four, Second Movement
- Brandenburg Concerto Number Four, Third Movement
- Fugue in C minor
- Fugue in G minor
- Sinfonia 1 (Three-part inventions)

Live Performances Dataset (Compositions by Johann Bach)

- Partita No. 4 in D Major, BWV 828 *performed by* Ka Jeng Wong

- Prelude and Fugue in F Major, WTC II, BWV 880 *performed by* Ryan Leung
- Prelude and Fugue in C Major, WTC II, BWV 846 *performed by* Ilya Maximov
- Partita in C Minor, BWV 826 *performed by* Timur Mustakimov
- Prelude and Fugue in D-flat Major, WTC I, BWV 848 *performed by* Shuan Hern Lee
- Prelude Fugue No. 06 in D Minor, WTC II *performed by* Carmen Knoll
- Partita No. 1 in B-Flat Major, BWV 825 (Complete) *performed by* Maria Kharsel
- English Suite in A Minor, BWV 807 (Complete) *performed by* Syuzanna Kaszo
- Prelude and Fugue in B Major, WTC II, BWV 892 *performed by* Leyla Kabuli
- Prelude Fugue No. 17 in A-Flat Major, WTC I *performed by* Dongxu Jin
- Toccata from Partita No. 6 in E Minor *performed by* Shih Wei Huang
- Prelude Fugue No. 14 in F-Sharp Minor, WTC II *performed by* Melanie Hebert
- Prelude and Fugue in F-sharp Minor, WTC II, BWV 883 *performed by* Eric Duo
- Prelude and Fugue in B Major, WTC I, BWV 868 *performed by* Josue Gonzalez
- Prelude and Fugue in C-sharp Minor, Book I, BWV 849 *performed by* Nicolas Giacomelli
- Toccata in C Minor, BWV 911 *performed by* Misha Galant
- Prelude Fugue No. 08 in D-Sharp Minor, WTC II *performed by* Daniel Eras
- Prelude and Fugue in D Minor, WTC II, BWV 875 *performed by* John Cao
- Prelude and Fugue in E-flat Minor, WTC I, BWV 853 *performed by* Jun Li Bui
- Prelude and Fugue in D Major, Book II *performed by* Feng Bian

Appendix C

Materials

List of material and files produced during the work.

- Model (Python files)
  - Notes Network
  - Velocities Network
  - Durations Network
  - Generate Notes
  - Generate Velocities
  - Generate Durations
- User-Testing
  - Questionnaire
  - Excerpts
  - Answers
  - Consent forms

All material and files can be found by clicking the following link (Dropbox shared folder): link: https://www.dropbox.com/sh/81kth86vgeyj78d/AAD0fzLV5pgNJvUFBX78xxWva?dl=0j