KAUNAS UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATICS

Ąžuolas Krušna

MUSIC GENERATION USING MACHINE LEARNING

Master's Degree Final Project

Supervisor
dr. Mantas Lukoševičius

KAUNAS, 2018

KAUNAS UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATICS

MUSIC GENERATION USING MACHINE LEARNING

Master's Degree Final Project
Informatics (code 6211BX007)

Supervisor
(signature) dr. Mantas Lukoševičius
(date)
Reviewer
(signature) assoc. prof. dr. Tomas Blažauskas
(date)

Project made by
(signature) Ąžuolas Krušna
(date)

KAUNAS, 2018

KAUNAS UNIVERSITY OF TECHNOLOGY

Faculty of Informatics

(Faculty)

Ąžuolas Krušna

(Student's name, surname)

Informatics, 6211BX007

(Title and code of study programme)

"Music generation using Machine Learning"

DECLARATION OF ACADEMIC INTEGRITY

20 ___ _____ ___

Kaunas

I confirm that the final project of mine, **Ąžuolas Krušna**, on the subject "Music generation using Machine Learning" is written completely by myself; all the provided data and research results are correct and have been obtained honestly. None of the parts of this thesis have been plagiarized from any printed, Internet-based or otherwise recorded sources. All direct and indirect quotations from external resources are indicated in the list of references. No monetary funds (unless required by law) have been paid to anyone for any contribution to this thesis.

I fully and completely understand that any discovery of any manifestations/case/facts of dishonesty inevitably results in me incurring a penalty according to the procedure(s) effective at Kaunas University of Technology.

_____ _____

(name and surname filled in by hand) *(signature)*

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

Krušna, Ąžuolas. Muzikos generavimas panaudojant mašininį mokymąsi. Magistro baigiamasis projektas / vadovas dr. Mantas Lukoševičius; Kauno technologijos universitetas, Informatikos fakultetas.

Mokslo kryptis ir sritis: fiziniai mokslai, informatika

Reikšminiai žodžiai: *algoritminė kompozicija, aido būsenų tinklas (ESN), MIDI, rekurentinis neuroninis tinklas, ilgos trumpalaikės atminties tinklas (LSTM)*

Kaunas, 2018. 61 p.

SUMMARY

Muzika visuomet buvo labai svarbi mūsų gyvenimuose. Iš tiesų, tai ne tik mūsų. Keletas iš geriau žinomų banginių yra pastebėti dainuojant taip pat. Muzika visąlaik mus lydėjo mūsų karuose, džiaugsmuose ir varguose. Šis straipsnis apžvelgia žmonių bandymus kurti muzikos kompoziciją pasitelkiant skaičiavimus. Nors ir algoritmiškai kuriama muzika egzistuoja nuo senų dienų, ji niekada nebuvo pritraukus tiek daug tyrėjų kaip pastaraisiais metais. Mūsų žiniomis tai buvo daroma viduramžiais Irane bei per Švietimo amžių Europoje. Nors ir algoritminės muzikos kompocijos forma pasikeitė užsiaugino sudėtingus sluoksnius, patys fundamentai, kurie generuoja muziką nepasikeitė, t. y. dauguma jų yra paremti tikimybinėmis sistemomis. Atsiradus šalia modeliams, kurie sugeba mokintis, mum tai suteikė galimybę imituoti visų nepaprastų muzikantų muziką per istoriją. Pati ši mintis atrodo neįtikėtina bei iš mokslinės fantastikos. Šis raštas atlieka išsamią technologijų, naudojamų muzikos generavimui, analizę. Toliau yra atliekamas tyrimas, bandantis rasti patį geriausią aido būsenos tinklo modelį, kuris panašiausiai nuspėtų legendinio Volfgango Amadėjaus Mocarto muziką. Iš tyrimo išplaukia, kad geriausi modeliai tie, kurie turi ilgalaikę atmintį. Pabaigoje yra pasiūlomos įžvalgos generuojant muziką su aidų būsenos tinklu.

Krušna, Ąžuolas. *Music Generation Using Machine Learning*: Master‘s thesis in Informatics / supervisor dr. Mantas Lukoševičius. The Faculty of Informatics, Kaunas University of Technology.

Research area and field: Physical Sciences, Informatics

Key words: *algorithmic composition, echo state network, MIDI, recurrent neural network, long short-term memory (LSTM) network*

Kaunas, 2018. 61 p.

SUMMARY

Music has always had a special role among humans. In fact, not only humans. Several of the popular baleen whale species have been observed to be singing. Music has been a great companion to our wars, joys and troubles. An overview on the attempts to compose music with computations is given

in this article. Even though algorithmic music has been around the world since the old days, it has never attracted as many researchers as in the recent years. To our knowledge it existed in Iran back in the Middle Ages and in Europe during the Age of Enlightenment. Though the form has changed and it has grown layers of complexity, the very foundations of the algorithm that generates musical compositions have not changed, i.e. most of them are based on structures of fortuity. Additionally, models that are able to learn have been discovered allowing us to imitate the music of the incredible artists throughout history. The thought alone is crazy to think of and seems to be from the sci-fi. In this paper a thorough analysis of the technologies for music generation is carried out. Moreover, a research trying to find the best model of an echo state network in order to mimic the music of the legendary Wolfgang Amadeus Mozart has been realized. As it turns out, the best models are the ones that rely on long-term dependencies. In the end, insights of using an echo state network for music generation are given.

1. INTRODUCTION

Music has always had a special role among humans. In fact, not only humans. Several of the popular baleen whale species such as the humpback whale and the blue whale have been observed to be singing. These marine mammals are able to produce loud melodic notes and sounds that mimic the sounds created by humans. They are commonly refered as whale songs. Unfortunately, researchers do not have a clear understanding of what these sounds mean and why they are sung. We are not able to interpret what exactly is being communicated between the male and the female whales but the fact that these songs are often heard during mating season [1] suggests us that singing has probably played a vital role for these fascinating mammals.

Music has always been a major attraction for the kings and royalties throughout centuries. At some points in history it has even been an indisputable activity for the aristocracy here in Europe, in Lithuania, all the way to Hawaii, through New Zealand to Iceland. It has not only been for aristocracy. Music has been used in wars to boost the morale of soldiers and to scare the enemies away. Vikings are known for their war chants, Maoris are known for their war cry *haka*, Winged Hussars are known to scare the enemy with the terrible sound produced by the feathers on their wings when they charge. Some armies even replaced some of the soldiers in the battlefield to drummers or bagpipers. Taking a look back to music as an entertainment and an uplifting activity for the crowds, we see that some of the classical musicians left a footprint so deep, that people create movies and go crazy about them to this day.

Singing brought us *Odyssey* and *Iliad*. Singing has led us through work and trouble. People sang during the Soviet deportations to Siberia. The partisans sang and had created a lot of songs during the resistance years. Singing has accompanied not only Lithuania, but other Baltic States to the independence as well [2].

Music has sparkled peaceful revolutions and movements around the world. Early in 1964 to demonstrate the power of music to people, *Life* magazine put a quote like this: “In [1776] England lost her American colonies. Last week the *Beatles* took them back.” [3]

We believe that music is very important to us as humans. It is a lot of fun as well. These are the reasons we chose music as our field of study. Getting to know that we might be able to create new music by mimicking the great musicians of the past, made us very curious and interested in this topic.

Algorithmic music is by no means a new trend in our techy world. Even Pythagoras believed that music and mathematics were meant to be studied together [3]. In fact, three Iranian brothers collectively known as Banu Musa were successfully devising automatic and even programmable musical instruments back in 850 AD [5]. They were most likely invited to the best parties in the city back then. Moreover, an algorithmic music game circulated around Europe since the Enlightenment

Age, i.e. the 18th century in a form of *Musikalishes Würfelspiel*. It has been attributed to Mozart in a form of myth, yet never proven to be true. This game took small fragments of music and combined them in a random order by chance, often tossing a dice. Since then, the scope of the algorithmic music has augmented layers of complexity, but the underlying foundations have not changed. The main difference is that now we do not toss a dice, but rather run a random number generation function in our favourite programming language.

The very first uses of computers to compose music dates back to the mid 1950s. It is roughly at the same time as the concept of Artificial Intelligence (AI) was coined at the Darmouth Conference although the two fields did not converge until sometime later. Computers were slow and expensive. They were also difficult to use as they operated in batch mode [6].

One of the most commonly cited examples is the *Illiac Suite* by Hiller and Issacson [7]. A composition that was generated using random number generators, rule systems and Markov chains in late 1956 [3, 6]. It was designed as a series of experiments on composition of formal music. During the following decade, Hiller's work inspired colleagues within the same university to further experiment with algorithmic composition. They were also using a library MUSICOMP for algorithmic composition written by Baker, also a collaborator of Hiller [8]. This library provided researchers a standard implementation of the various methods used by Hiller and others [6].

A renowned *avant-garde* composer, Iannis Xenakis, profusely used stochastic algorithms to generate raw material for his compositions using machines since the early 1960s to automate these methods [8]. While Keonig was not as a well-known composer as Xenakis was, he implemented and algorithm PROJECT1 using serial composition (a musical theory) and other techniques as Markov chains to automate the generation of music in 1964 [8]. And there were many others burning with the passion to *teach* machines compose music in the early days of computational algorithmic music generation [8].

Musical composition has been one of the long term goals of artificial intelligence [9]. It brings together machine learning researchers that aim to capture the complex structure of music as well as music professionals and enthusiasts that wish to see how far a computer got to be a real composer [10].

Artificially generated music is being experimented in healthcare as a pain and stress remedy. Real time music generation is also explored for gaming environments [11].

Broadly speaking, music generation by AI is based on the principle that musical styles are in effect complex systems of probabilistic relationships, as defined by the musicologist Leonard B. Meyer.

Recently, AI music composition has gotten quite some fame as the *composer's* Iamus 4 works have been performed by the London Symphony Orchestra. Actually, Iamus is a computer. No wonder its first music is called *Hello, world!* Indeed, the performance of Iamus 4 works by the London

Symphony Orchestra created the album *Iamus*. No previous attempts to make music by computer dating back to the early days of computation have been afforded such severe attention [12]. "New Scientist" reported it as the first complete album to be composed by a computer and recorded by humans.

Artificially composing and generating music has attracted even such big industry players as *Google* and *Sony*. *Google Brain* team has launched their project *Google Magenta* with Douglas Eck in the lead to answer the question of whether we can use machine learning to create compelling art and music [13]. *Sony* has dedicated this task to its Computer Science Laboratory (CSL). Their team has developed an artificial intelligence to create the first ever full pop song composed by AI [14]. However, it must be noticed that the French composer Benoit Carre arranged the song and wrote the lyrics. Recently, *Sony* has also developed a model that creates chorales in the style of J. S. Bach [15]. Around 50 percent of the listeners believed it was authentic Bach music.

Having said that, music composition is not a trivial task and researchers find it intimidating. Even such big companies as *Google* and *Sony* face a bunch of obstacles and struggle to create an algorithm that produces great music.

We may also see that AI music composition is on the rise and is quite a *hot* topic. Maybe even at the breakpoint of innovation. It has never attracted as many scientists as in the recent years.

As the models that are able to learn the patterns of the music have been discovered, the thought that we might be able to imitate the music of the incredible artists throughout history has driven us crazy. It even seems to be from the sci-fi.

In this paper we are trying to train a system to generate classical piano music. For a quantitative rather than qualitative analysis only one composer was chosen. Mozart has been opted for his indisputable genius and some haphazardness.

Rather than working with sound signals, we chose to work with notes for several reasons. Firstly, it is a lot less complex. Therefore, it is a lot easier for us to understand and analyse it as well as it is for the algorithm in the means of computational resources and dependency on previous notes. In the note level we are also able to compare it with the musical theory. And it helps us stay in the realm of classical music as well.

Furthermore, we are lucky enough to have the MIDI (musical instrument digital interface) protocol for a *.mid* is a musical file format that captures the notes, the times piano keys were pressed and released, how strong they were pressed etc. MIDI supports 128 notes whereas general pianos usually provide 88 keys.

1.1. Aim of the research

The main aim of the research is a development of an algorithm that predicts classical music by using machine learning which could be later used for music generation.

1.2. Objectives

1. Choice of music
2. Selection of tools
3. Research of existing algorithms used for music composition
4. Choice of an algorithm for music composition
5. Choice of evaluation of the algorithm
6. Research on the algorithm
7. Development of the algorithm
8. Insights for music generation

1.3. Novelty of research in science

Echo state networks have almost not been observed to be used for algorithmic music composition.

1.4. Structure of the document

Chapter 2 overviews pitches, the rise of electronic music generation and the emergence of the MIDI music protocol. It also provides an analysis of the overall advancements and cutting edge approaches to music generation by artificial intelligence. Chapter 3 provides reasoning for my choice of music, data, tools, machine learning algorithms and the general approach to data preparation, analysis, evaluation and the overall scheme of the experiment. Chapter 4 gives away results of the research and their analysis. It provides insights based on the results as well. Chapter 5 summarizes all this data. Chapter 6 presents all of the references used to write this thesis.

2. LITERATURE OVERVIEW

2.1. Pitch

Let us first define what a pitch is since we are dealing with music. What makes the notes different? Is it just the notation or is there something else? Yes, there is something very unique about every note – their pitch. Pitch is the common quality of sound that makes it possible to judge the sounds in a sense as being "higher" or "lower". It is a perceptual property of sound such that allows their ordering on a frequency related scale. Pitches can only be determined in sounds that have a frequency that is clear and stable enough to distinguish from noise. Pitch is one of the major auditory attributes of musical tones. One may say that pitch could be quantified as a frequency yet pitch is not a purely objective physical property. It is more of a psychoacoustical attribute of sound. Frequency is an objective, scientific attribute that can be measured whereas pitch is a personal subjective perception of a sound wave and it cannot be measured directly. Anyhow, this does not necessarily conclude that most people will not agree on the highness or lowness of a particular note.

Historically, the studies of pitch and pitch perception have been a central problem in psychoacoustics. It has been extremely handy in forming and testing various theories of sounds representation and processing as well as perception in the auditory systems.

There is another definition of pitch by the American National Standards dispenses with the musical reference: "Pitch [is] that attribute of auditory sensation in terms of which sounds may be ordered on a scale extending from low to high. Pitch depends primarily on the frequency content of the sound stimulus, but it also depends on the sound pressure and the waveform of the stimulus" (ANSI 1994). It appears to be a quite fairly broad definition, requiring the words "low" and "high" to be associated with pitch or frequency, rather than with volume of the sounds, for example. Also, this definition seems to include what some would regard as timbral effects, such as the increase in the "brightness" of a sound as the level of its high-frequency aspects increase [63].

A pitch standard is the conventional pitch reference group of different musical instruments that are tuned for a performance. Concert pitches may vary from an ensemble to another ensemble. Over the period of history, the standard pitches have widely varied.

Standard pitch is the more commonly accepted convention than concert pitch. The A above middle C is usually set to 440 Hz although other frequencies such as 442 Hz are also often and possible variations. It may be written as "A = 440 Hz" or sometimes as "A440". Another standard pitch has been set in the 20th century as "A = 415 Hz" – approximately an equal-tempered semitone lower than A440 to facilitate transposition.

Transposing instruments have their origin in the variety of pitch standards. In modern times they conventionally have their parts transposed into different keys from voices and other instruments or

even from each other. Consequently, musicians need a path to refer to a particular pitch in an unambiguous manner when communicating with one another.

As a great example, let us take the most common type of trumpet or clarinet. While playing a note written in their part as C, it sounds as that pitch is Bb on a non-transposing instrument like a violin. This means that at one of these wind based instruments the pitch is a tone lower than the violin pitch. In order to refer to that pitch unambiguously a musician calls that pitch a "concert" Bb. To no surprise the human perception of musical intervals is not linear. It is approximately logarithmic with respect to fundamental frequency. Humans perceive the difference of the pitches "A220" and "A440" in the same way they perceive an interval between the pitches of "A440" and "A880". Encouraged by such a logarithmic perception, music theorists sometimes represent pitches using a numerical scale based on the logarithm of the fundamental frequency, for instance, one may adopt the widely used MIDI standard to map fundamental frequencies f to real numbers p as follows:

$$p = 69 \times \log_2 \frac{f}{440Hz} \quad (1)$$

It creates a linear space of pitches in which the size of octaves is 12 semitones. Semitone has a size of 1. "A440" is assigned as the number 69. Distance in this space corresponds to all the musical intervals as understood by artists and composers. An equally tempered semitone can be subdivided into 100 cents. Such a system is flexible enough to even include microtones not found on standard piano keyboards. Thus, the pitch halfway between C (60) and C-sharp (61) can be labelled 60.5.

Below is the Table 1 giving the frequencies in Hertz of musical pitches, covering the full range of most musical instruments. It uses an even tempered scale with A equal to 440 Hz. Semitone has a frequency ratio of a twelfth root of 2 which is approximately equal to 1.059.

Table 1. Frequencies of notes

	C	C#	D	Eb	E	F	F#	G	G#	A	Bb	B
0	16.35	17.32	18.35	19.45	20.6	21.83	23.12	24.5	25.96	27.5	29.14	30.87
1	32.7	34.65	36.71	38.89	41.2	43.65	46.25	49	51.91	55	58.27	61.74
2	65.41	69.3	73.42	77.78	82.41	87.31	92.5	98	103.8	110	116.5	123.5
3	130.8	138.6	146.8	155.6	164.8	174.6	185	196	207.7	220	233.1	246.9
4	261.6	277.2	293.7	311.1	329.6	349.2	370	392	415.3	440	466.2	493.9
5	523.3	554.4	587.3	622.3	659.3	698.5	740	784	830.6	880	932.3	987.8
6	1047	1109	1175	1245	1319	1397	1480	1568	1661	1760	1865	1976
7	2093	2217	2349	2489	2637	2794	2960	3136	3322	3520	3729	3951
8	4186	4435	4699	4978	5274	5588	5920	6272	6645	7040	7569	7902

In order to properly address this table, the octave number is in the left column. Thus, to find the frequency of middle C which is C4, look down the "C" column till you get to the 4th row. That is that

the middle C is 261.6 Hz. This table provides frequencies in relation to the even tempered scale – an octave is a frequency change between notes of exactly two times. Piano range usually goes from A0 to C8, i.e. from 27.5 to 4186 Hz.

2.2. Electronic music

Reaching back to grab the grooves of 1970s, electronica became a whole new entity in and of itself, spinning new sounds and subgenres without an end in sight of two decades down the spike. Grooves began with the disco and funk music and all the gadgets for electronic composition.

Its development came in the post-disco environment of Chicago and New York that spawned house music during the 1980s. Detroit, of course, too. It was the motherland for techno music.

Electronic music became more dominant in pop music in the 1980s with the greater reliance on synthesizers. Since the emergence of digital technologies and digital synthesizers, a group of musicians and music merchants developed digital interface for music. This allowed electronic artists to communicate easily.

Later that decade, clubbers in Great Britain latched onto the fusion of sensual and mechanical sounds and returned the favour to hungry Americans with brand new styles like jungle, drum'n'bass and trip-hop. Although most of all early electronica was danceable, by the beginning of the 1990s, producers were also making music for the headphones and chill areas as well. This resulted in dozens of stylistic fusions like ambient-house, experimental techno, tech-house, electro-house, etc. It was typical for all of the styles gathered under the umbrella to bear a focus on danceable grooves, highly loose structure if any at all and for some a relentless desire to find new sounds no matter how tepid the result [16].

2.3. MIDI protocol

And it is due to these music artists and enthusiasts that we have the MIDI (Musical Instrument Digital Interface) protocol. MIDI is an industry standard music technology protocol. Though it was developed in the 1980s, it still on today connects products from many different companies including digital musical instruments, computers, tablets and smartphones. MIDI allows electronic instruments and other digital musical tools to communicate with each other. MIDI is used around the world every day by musicians, DJs, producers, artists, educators and hobbyists to create, perform, teach, learn and share their talent [17, 18].

MIDI itself does not produce sound. It is a series of messages as "note on", "note off", "pitch/note", "pitch bend", etc. It is these messages that are interpreted by a MIDI instrument to generate sound. A MIDI instrument could either be a piece of hardware such as electronic keyboard

or a synthesizer or a part of a software environment such as “Ableton”, “GarageBand”, “FL Studio”, “Digital Performer”, etc. [17].

One of the reasons for the popularity of MIDI files is that it does not to capture and store actual sound unlike audio files (.mp3, .wav, .aiff, etc.) or even compact discs or cassettes. Instead, the MIDI file can be just a list of events that describe particular steps for either the soundcard or other playback device to generate certain sounds. Thus, MIDI files are very much smaller than digital audio files. In addition, those events are also adjustable allowing the music to be rearranged, edited or even composed interactively. [17] Moreover, the same melody of a MIDI file can be played by various instruments since it only describes which notes to play. This changes the overall sound of the composition.

To sum up, advantages of a MIDI file are the following:

1. It is extremely compact. An entire song could be stored within a few hundred MIDI messages. In comparison, usual audio data is sampled thousands of times per second.
2. Easily adjustable notes. One is able to change either the pitch or the duration of the note without having to rerecord them once again.
3. Interchangeable instruments. The very same melody can be played by a different instrument giving it a completely different timbre [17].

Furthermore, the messages of the MIDI files carry important information. A “note on” message consists of two pieces of information [17]:

1. Note/pitch – which note is to be played
2. Velocity – how fast it should be pressed

Note in the message describes the pitch as a value between 0 and 127. It lists all the MIDI notes and their standard musical notation equivalents. For instance, MIDI note 60 is the middle C (C4), whereas middle A (A4) is represented as a number 69. Just as introduced before by the MIDI standard and shown in the Table 2.

Table 2. MIDI standard

Note	…	C4	C4#	D	Eb	E	F	F#	G	G#	A	Bb	B	…
#	…	60	61	62	63	64	65	66	67	68	69	70	71	…

Likewise, velocity is also a number between the values 0 and 127. These values basically describe the volume of a MIDI note. Higher velocity means that the note is played louder. Occasionally, different timbres create different sounds in an instrument. For instance, a MIDI flute might sound more frictional at a higher velocity as if one was blowing into it heavily. At lower velocities it sounds more sinusoidal and cleaner [17].

“Note on” message represents the start of the note. A “note off” message is sent when the note is supposed to end.

In the end, MIDI carries event messages that specify notation, pitch and velocity, control signals for parameters such as volume, vibrato, audio panning, cues, and clock signals that set and synchronize tempo between multiple devices. These messages are sent via a MIDI cable to other devices, where they control sound generation and other features, or a digital cable to a digital synthesizer.

Standard MIDI file (SMF) format is different from native MIDI protocol for the events are time stamped for playback in the proper sequence. Standard MIDI files come basically in two varieties. A Type 1 and a Type 0. There is also a Type 2 but is not so common. In a Type 1 file individual parts are saved on different tracks within the sequence, whereas in a Type 0 file everything is merged onto a single track [19].

MIDI music files are also libraries of musical pieces because they contain the sequences of notes of the pieces and even some additional information.

2.4. Algorithmic music

2.4.1. The process of algorithmic composition

Algorithmic music composition is basically music composition based on a set of certain algorithms. Algorithms themselves, or at least some formal set of rules, have been used for the purpose of music composition for centuries. A very simple example of using a procedure to generate a piece of music is to use a 12-sided die (numbered from 1 to 12) to determine the order of pitches in the composition. An association, or mapping is made to correlate each pitch in a twelve-tone equal tempered scale with each number of the rolling die [21]. Such mapping of pitches to numbers is demonstrated in Fig. 1.

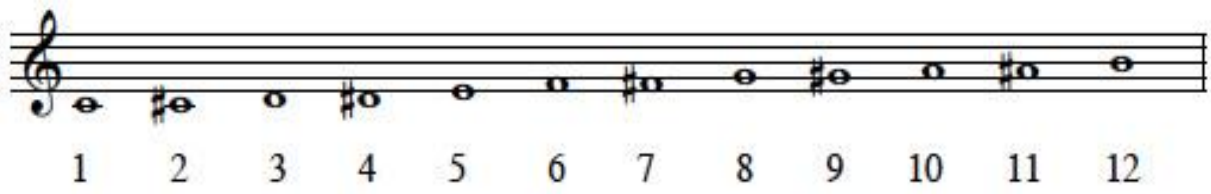

Fig. 1. 12 notes mapping to numbers [21]

Let's say that we decide to roll the die six times. Our six tosses return the numbers 2, 5, 3, 9, 3, and 12. The music that results from our roll of the die is found in Fig. 2.

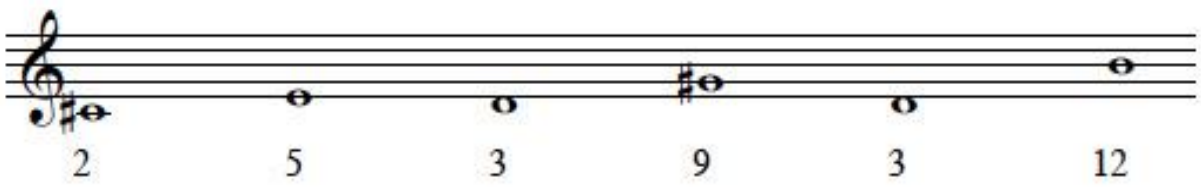

Fig. 2. 12-sided die mapping of numbers to pitches [21]

Then this simple algorithm confronts us with the question if it can produce interesting music. How do we determine the other aspects of the composition as rhythm, timbre, loudness, register, etc.? These questions have been explored by various composers throughout history [21].

The Greek philosopher, mathematician and music theorist Pythagoras documented the relationships between music and mathematics around 500 BC. These relationships laid foundations for our modern study of music theory and acoustics. The Greeks believed that the understanding of numbers was key to understanding the whole universe. The quadrivium, their educational system, was based on the study of music, arithmetic, geometry, and astronomy. Although we have numerous treatises on music theory dating from Greek antiquity, the Greeks left no clues if mathematical procedures were applied to the composition of music [21].

Over a thousand years later, the hard work of music theorists such as Guido d'Arezzo established the framework for our conventional system of music notation. His system employed a staff accompanied by a clef making it possible for a composer to notate a score so that it could be performed by someone other than the composer himself. Prior to the development of the score, music was learned by rote and generally improvised and embellished by the performing artist. By the thirteenth century, formalized music composition began to replace improvisation and the role of composer and performer became increasingly distinct because it has not been as distinct before [21].

The music theorist Franco of Cologne established rules for the time values of single notes, ligatures, and rests in his treatise *Arts canus mensurabilis* around 1250. By the early fourteenth century, composers began to treat rhythm independently of pitch and text [21].

Algorithmic composition used to plot voice leading in Western counter points, for instance, could be simply reduced to algorithmic determinacy sometimes.

Some of the algorithms or the arrangements that have no immediate musical relevance might be used by the composers as a creative inspiration for their own music. Such is the outsider music. Outsider music is not by any means part of the commercial music industry and is written in a way that ignores standard musical or lyrical conventions. This is due to the fact that either they have no formal training at all or just simply because they disobey with the conventional rules. It lacks typical structure and may even incorporate bizarre lyrics and melodies. This type of music has a few outlets. Recordings or performers are usually heard through various fan chat sites or just by the name of theirs among

music experts such as music collectors and connoisseurs. Outsider musicians have much more control over the final product either because of their inability or objection to cooperate with producers. Only very few of the outsiders attain any of the mainstream popularity. There is, however, a niche market for outsider music [20].

There is a variety of algorithms that have been used as source materials – fractals, statistical models, L-systems and even arbitrary data, e.g. census figures, GIS coordinates, magnetic field measurements.

Music is considered as composed by computers if the algorithm is able to make choices of its own during the process of creation. Even though there is no universal method to list and sort various compositional algorithms into categories, one may look at the way the algorithm takes part in the process of composition. Differentiating upon this, we can see that it can be divided into:

1. Music composed by the computer
2. Music composed with the aid of the computer

Another way to divide the compositional algorithms is to examine the results of such compositional actions. Algorithms can do the following:

1. Provide notations for other instruments as notes or MIDI
2. Provide an independent sounds synthesis
3. Do both of the above

One may as well categorize compositional algorithms by their structure and the way they are processing the data. Thus, one shall be able to distinguish these partly overlapping types:

1. Mathematical models
2. Knowledge-based systems
3. Grammars
4. Evolutionary methods
5. Systems that learn
6. Hybrid systems

This type of division in to categories is not straightforward since many of the AI methods can be considered as equivalent. For instance, Markov chains are similar to the type-3 grammars. Furthermore, some of the systems have more than one prominent feature. In such cases, the method that is more responsible for the generation of musical output is chosen. Most of the research has been done in evolutionary methods and in systems that learn [4].

2.4.2. Mathematical models

Mathematical models are based on mathematical equations and random events. The most common way to create compositions through mathematics is by using stochastic processes. In

stochastic models a piece of music is composed as a result of non-deterministic methods. The compositional process is only partially controlled by the composer by weighting the possibilities of random events. Prominent examples of stochastic algorithms are Markov chains and various uses of Gaussian distributions. Stochastic algorithms are often used together with other algorithms in various decision-making processes [4].

Music may be composed from the inspiration of natural phenomena as well. Such chaotic models create compositions from the harmonic and inharmonic phenomena of nature. For example, since the 1970s fractals have been studied as models for algorithmic composition as well.

2.4.3. Knowledge based systems

In some sense, most of AI systems are knowledge based systems (KBS). Here, we mean systems which are symbolic and use a set of rules or constraints. The use of KBS in music seems to be a natural choice especially when we try to model well defined domains or we want to introduce explicit structures and rules. Their main advantage is that they have explicit reasoning. Their choice of actions can be explained [4]. One way to create compositions is to isolate the aesthetic code of a certain musical genre and use this code to create new similar compositions. Knowledge based systems are based on a pre-made set of arguments that can be used to compose new works of the same style or genre.

For example, Ebcioglu [25] implemented his own Backtracking Specification Language (BSL) and used this to implement CHORAL. CHORAL is a rule-based expert system for the harmonization of chorales in the style of J. S. Bach.

Even though KBS seem to be the most appropriate choice for algorithmic composition as a stand-alone method, they still exhibit some key problems:

1. Knowledge elicitation is difficult and time consuming, particularly in such subjective domains as music
2. Since they do what we program them to do, they depend on the ability of the *expert.* The expert in many cases is not the same as the programmer to clarify the concepts and find a flexible representation.
3. These systems become too complicated if we add all of the exceptions to the rule and their preconditions, something that is necessary in this domain

2.4.4. Grammars

Music can also be examined as a language with a distinctive grammar set. Compositions are created by first constructing a musical grammar, which is then used to create comprehensible musical

pieces. Grammars often include rules for macro-level composing, for instance, harmonies and rhythm rather than single notes.

2.4.5. Evolutionary methods

Evolutionary methods of composing music are based on genetic algorithms. The composition is being built by the means of evolutionary process. Through mutation and natural selection, different solutions evolve towards a suitable musical piece. Iterative action of the algorithm cuts out bad solutions and creates new ones from those surviving the process [22]. Genetic algorithms (GAs) have proven to be very efficient search methods, particularly when dealing with problems with very large search spaces. Coupling this with their ability to provide multiple solutions, which is what is often needed in creative domains, makes them good candidate for a search engine in a musical application. Based on the implementation of the fitness function, we shall divide the algorithms into two categories [4].

1. Use an objective fitness function

In this case the chromosomes are evaluated based on the formally stated and computable functions [4].

2. Use of a human as a fitness function

Usually we refer to this type of GA as an interactive-GA (IGA). In this case a human replaces the fitness function in order to judge and evaluate the chromosomes. IGA suffer from subjectivity and efficiency. The *bottleneck* is that the user has to hear all of the potential solutions to evaluate a population. In the end, this approach tells us little about the mental processes involved during music composition since all the reasoning is encoded inaccessibly in the mind of the user [4].

2.4.6. Systems that learn

In the category of learning systems are systems which, in general, do not have a priori knowledge (e.g. production rules, constraints) of the domain. Instead, they learn its features from the examples supplied by the user or programmer. These systems can be further classified, based on the way they store information. Either to subsymbolic/distributive systems or symbolic systems [4]. This method of algorithmic composition is strongly linked to algorithmic modelling of style, machine improvisation, and such studies as cognitive science and the study of neural networks. In other words, it is called machine learning.

2.4.7. Evaluation of the systems

We cannot help to notice a two-fold lack of experimental methodology in many research reports in this area. Firstly, there is usually no evaluation of the output by real experts, e.g. professional musicians in most of the systems. Secondly, the evaluation of the system is not given enough attention and work in consideration with respect to the length of the reports [4].

Then there are unanswered musical questions for systems which only generate melodies. Most melodies will sound acceptable in a given context. How can we expect to evaluate the generated music if we do not have a defined harmonic context for it [4]?

2.4.8. Knowledge representation

The two almost ubiquitous issues in AI are the representation of knowledge and the search method. Our categorisation above mostly reflects the search method. This, however, constrains the possible representations of knowledge [4].

In many AI systems, particularly symbolic systems, the choice of knowledge representation is an important factor in reducing the search space. In the articles [23, 24] researchers used a more abstract representation, representing the degrees of the scale rather than the absolute pitches. By doing this, the search space was immensely reduced since the representation did not allow generation of non-scale notes that are considered dissonant as well as inter-key equivalence was abstracted out [4].

Most of these reviewed systems exhibit a single and fixed representation of the musical structures. On the other hand, some systems viewpoints as described in [4, 25, 26] and the authors believe that it simulates human musical thinking more closely.

2.4.9. Computational creativity

Most probably the most difficult task is to integrate the concept of creativity into such systems. This is challenging since we do not have a clear idea of what creativity is [27].

Luckily, Rowe and Partridge [28] proposed some characteristics of computational creativity.

1. Knowledge representation is organised in such a way that it maximises the number of possible associations. In other words, a flexible knowledge representation scheme. Similarly, Boden [27] claims that representation ought to allow to explore and transform the conceptual space.
2. Tolerates ambiguity in representations
3. The usefulness of new musical combinations should be assessable
4. Allows multiple representations in order to avoid the problem of *functional fixity*.
5. New combinations need to be elaboratable so that their consequences could be found out

Papadopoulos and Wiggins [4] propose one question that AI researchers should aim to answer:

Do we want to simulate human creativity itself or the results of it? (Is DEEP BLUE creative, even if it does not simulate the human mind?) This is more or less similar to the, subtle in many cases, distinction between cognitive modelling and knowledge engineering.

As for a more recent example, we could ask the same question about AlphaGo instead of DEEP BLUE. Which had beaten the world champions in Go game more recently.

Papadopoulos and Wiggins [4] then ask: "Even after all these, will computers be able to emulate our musical thinking?"

In [29] Kugel expands on what Myhill [30] seems to have first proposed that in musical thinking there is more than computing. He then proposes that we should in addition implement incomputable processes *to our conceptual palette*. These have also been called limiting-computable processes by Gold [31] and trial-and-error processes by Putnam [32].

2.5. Machine learning

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can teach themselves to grow and change when exposed to new data. [40] It evolved from the studies of recognition of various patterns as well as computational learning theory in artificial intelligence. [41] Machine learning explores the study and construction of algorithms that can learn from and provide predictions on data – such algorithms overcome following strictly static program instructions by making data driven predictions or decisions through fitting a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms is unfeasible. Such applications include spam filtering, detection of network intruders or malicious insiders working towards a data breach [42], optical character recognition, search engines, computer vision and, of course, art and music generation.

The process of machine learning is similar to that of data mining. Both systems search through data to look for patterns. However, instead of extracting data for human comprehension, machine learning uses that data to detect patterns in data and adjust program actions accordingly. Machine learning algorithms are often categorized as being supervised or unsupervised. Supervised algorithms can apply what has been learned in the past to new data. Unsupervised algorithms can draw inferences from datasets.

Facebook's News Feed uses machine learning to personalize each member's feed. If a member frequently stops scrolling in order to read or "like" a particular friend's post, the News Feed will start to show more of that friend's activity earlier in the feed. Behind the scenes, the software is simply using statistical analysis and predictive analytics to identify patterns in the user's data and use patterns

to populate the News Feed. Should the member no longer stop to read, like or comment on the friend's posts, that new data will be included in the data set and the News Feed will adjust accordingly.

Resurging interest in machine learning is due to the same factors that have made data mining and Bayesian analysis more popular than ever. These things are like growing volumes and varieties of available data, computational processing that is cheaper and more powerful, affordable data storages.

All of them combined means that it is possible to quickly and automatically produce models that can analyse bigger and more complex data, deliver faster and more accurate results – even on a very large scale. Also, by building precise models, an organization has a better chance of identifying profitable opportunities – or avoiding unknown risks.

Let us elaborate on how is machine learning different from statistical models. The main difference with machine learning is that just like statistical models, the goal is to understand the structure of the data. That is to fit theoretical distributions to the data that are well understood. Firstly, with statistical models there is a theory behind the model that is mathematically proven, but this requires that data meets certain strong assumptions too. Machine learning has developed based on the ability to use computers to probe the data for the structure, even if we do not have a theory behind of what that structure looks like. The test for a machine learning model is a validation error on new data, not a theoretical test that proves a null hypothesis. In the end, since machine learning often uses an iterative approach to learn from input data, the learning can be easily automated. Passes are run through the data until a robust pattern is found [42].

Moreover, a type of machine learning with many hidden layers of algorithmic computations make it Deep Learning. This refers neural networks. By having multiple layers, neural networks are able to learn more complex prediction functions but this hides away the thinking of the network even more. Makes it even harder for us to reason about its reasoning. Deep learning combines advances in computing power and special types of neural networks to learn complex patterns in huge amounts of data. Deep learning techniques are currently state of the art for identifying objects in images and words in sounds. Researchers are now looking to apply these successes in pattern recognition to more complex tasks such as automatic language translation, medical diagnoses and numerous other important social, business problems and art.

Finally, machine learning tasks are typically classified in three main broad categories depending on the nature of the learning system:

1. Supervised learning

In this case the computer is presented with example inputs and their outputs. The goal of this system if to learn a general rule that maps inputs to outputs as closely as possible without wasting too many computational resources.

2. Unsupervised learning

Here the algorithm does not receive any labels. It has to find patterns of data for the given data. Unsupervised learning can be a goal in itself discovering hidden patterns among data, e.g. grouping news in a news website or grouping customers of a retailer.

3. Reinforcement learning

This algorithm is used to interact with dynamic environment in which it has to perform a certain goal. Goals could be like driving a vehicle or flying a drone without telling it explicitly whether it has come close to its goal.

And, of course, there can be various combinations of these three types of machine learning systems.

Some great recent examples of reinforcement learning are learning to play game against an opponent just as DEEPBLUE played and won chess against then world chess champion Gary Kasparov and when AlphaGo beat the world Go champion Lee Sedol.

2.5.1. Neural networks

Artificial neural networks are computing systems inspired by biological neural networks that constitute animal brains. A single node in a simple neural network takes some number of inputs and performs a weighted sum of these inputs by multiplying each of them by some weight before adding them all together. In addition, there is a constant added that the sum that is called bias. The overall sum is then squashed into a small range using a nonlinear activation function such as a sigmoid function. This range usually geos from -1 to 1 or from 0 to 1. We can visualize this node by drawing its inputs and outputs as arrows denoting the weighted sum and activation by a circle. Then we can take multiple nodes and feed them all the same inputs, yet allow them to have different weights and biases. This is known as a layer (Fig. 3).

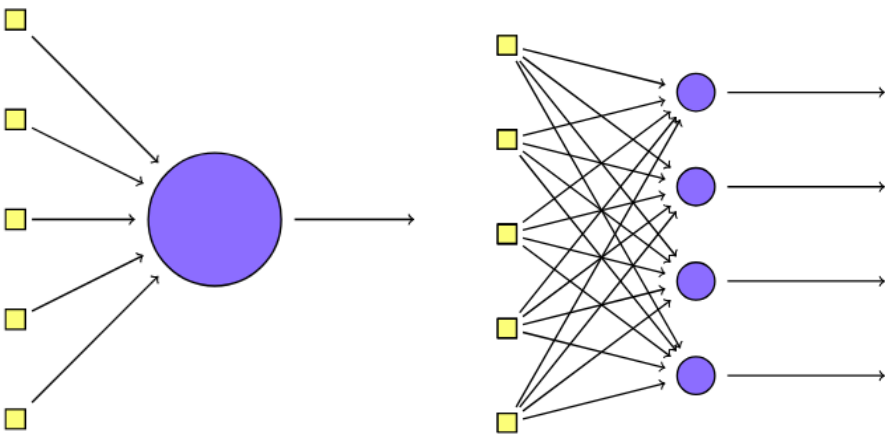

Fig. 3. Single neural network node (perceptron) on the left and a layer of 4 nodes on the right [60]

Note that because each node in the layer performs a weighted sum, yet they all share the same inputs, we shall calculate the outputs using a matrix multiplication followed by elementwise activation. This is one of the reasons why neural networks can be trained so effectively [51]. Multiple layers can be connected together as in Fig. 4.

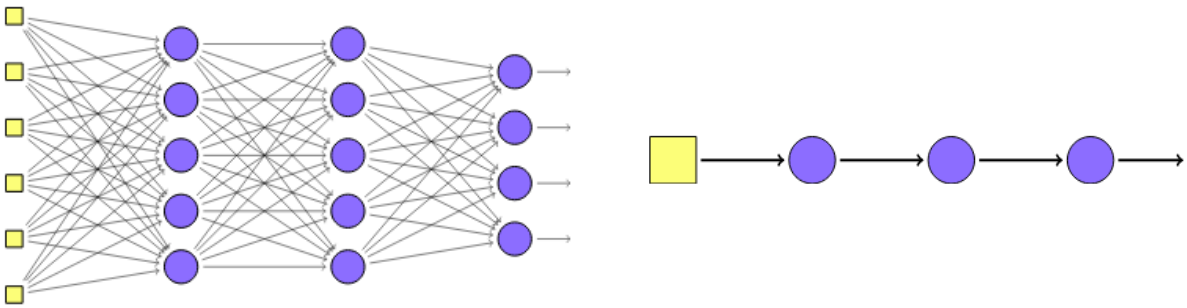

Fig. 4. Neural network and its visual simplification on the right [52]

This gives us a neural network. Only to be on the same page, the set of inputs is called the input layer whereas the last layer in called the output layer. All the other layers in between are called hidden layers. Each arrow carries the same value since each node has a single output value. For simplicity, layers can be visualised as single objects Fig. 4. This is how they are implemented most of the time.

2.5.2. Recurrent neural networks

Notice that in the basic feedforward network there is a single direction in which the information flows: from input to output. In a recurrent neural network, this direction constraint is absent. There are a lot of possible networks that can be classified as recurrent, but we will focus on one of the simplest and most practical. Basically, what we can do is take the output of each hidden layer, and feed it back to itself as an additional input. Each node of the hidden layer receives both the list of inputs from the previous layer and the list of outputs of the current layer in the last time step. (So if the input layer has 6 values, and the hidden layer has 3 nodes, each hidden node receives as input a total of 6+3=9 values.) [52] This is visualised in Fig. 5.

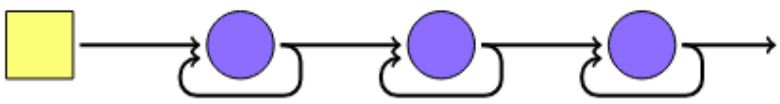

Fig. 5. Recurrent neural network [52]

This can be shown more clearly by unwrapping the recurrent network along the time axis (Fig. 6).

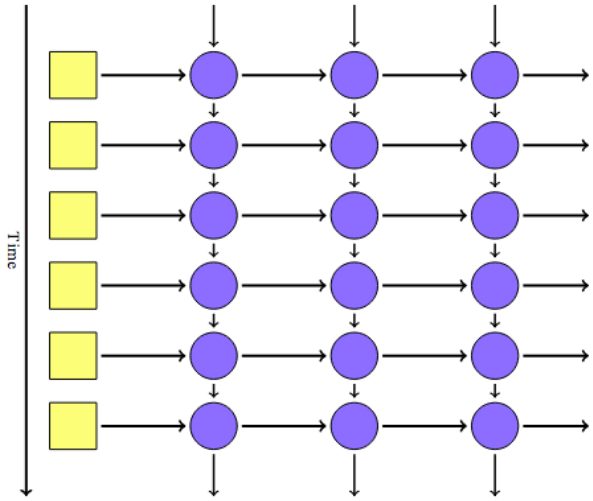

Fig. 6. Recurrent neural network unwrapped along the time axis [52]

In this representation, each horizontal line of layers is the network running at a single time step. Each hidden layer receives both input from the previous layer and input from itself one time-step in the past. The power of this is that it enables the network to have a simple version of memory, with very minimal overhead. This opens up the possibility of variable-length input and output: we can feed in inputs one-at-a-time, and let the network combine them using the state passed from each time step [60].

In general, Adrej Karpathy, now director of AI at Tesla, then a PhD student at Stanford, in his blog post [58] puts that there is something magical about recurrent neural networks.

The major problem with traditional recurrent neural networks is that their inner memory is very short-term. Any value that is output in one time-step becomes input in the next, but unless that same value is output again, it is lost at the next tick. As put in the paper [54] by Hochreiter and Schmidhuber, RNNs have been notoriously hard to train because of *vanishing gradients*. It is a problem commonly seen in RNNs when training with gradient based methods. Gradient methods such as back-propagation through time (BPTT) [55] and real-time recurrent learning (RTRL) [56], as well as their combinations, update the neural network by flowing the errors *back in time*. Whilst the error propagates from layer to layer, it has a tendency to either explode or shrink exponentially depending on the magnitude of the weights. Therefore, the network fails to learn long-term dependency between inputs and outputs [57]. Tasks with time lags that are greater than 5-10 are already difficult to learn [54], not having to mention that dependency of music usually spans across from tens to hundreds of notes in time, which contributes to the unique phrase structures of music.

Long short term memory (LSTM) [54] algorithm was designed to tackle the error-flow problem by enforcing constant error flow through the *constant error carousels* in its internal states [57]. LSTM learns quickly and efficiently. It also proved to be effective in multiple recognition tasks.

Using LSTMs introduces a *memory cell* value that is passed down for multiple time steps (Fig. 7), and which can be added to or subtracted from at each tick [52].

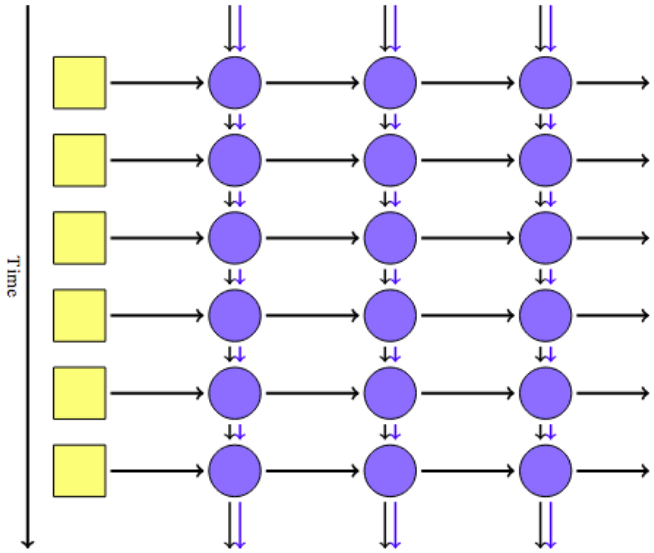

Fig. 7. LSTM unwrapped along the time axis [52]

For comparison, traditional recurrent and LSTM neural networks are pictured in Fig. 8.

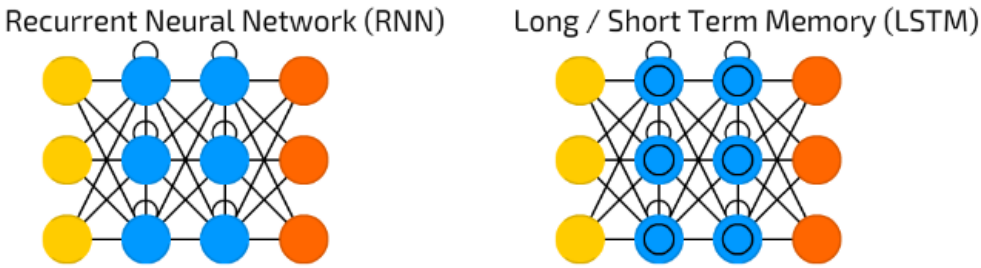

Fig. 8. Recurrent and LSTM neural networks [53]

In the figure yellow (left) circle is an input cell. Blue (middle) circle is a memory cell. Cell with a black circle inside is a memory cell. Orange (right) is an output cell.

2.6. Overview of the advancements in music generation

In the early days, symbolic AI methods and specific grammars describing a set of rules had driven the composition [33, 34]. Then these methods were significantly improved by evolutionary algorithms in a variety of ways [35] as represented by the famous EMI project [36]. More recently, statistics in the form of Markov chains and hidden Markov models (HMM) played a major part in algorithmic composition [37]. Next to this development was the rapid rise of neural networks (NN) due to the growing capacity of computational powers. It has made a remarkable process not only in the AI world but also in music composition [38].

As music is a sequence of notes, a sequential model was chosen to train on Mozart's music. Markov models are not very suitable for this task due to their monophony (although it is possible to design a system for polyphonic music as well). Currently, the cutting-edge approach to generative music modelling is based on recurrent networks [9, 10, 39, 15, 61, 57] like the long short-term memory (LSTM) network.

2.6.1. Overview of advancements in music generation using RNN

One of the earliest papers that used a RNN music generation was by Todd [43]. It generated a note-by-note music with Jordan recurrent neural network. Jordan recurrent network is a simple RNN that has a recurrent connection from the output layer to the input layer and a recurrent link at the input link. The network is trained with back-propagation through time. Recurrence is managed by teacher-forcing. In the training phase, Todd had trained monophonic melodies with this network. The trained network could then be used to generate music. This could be achieved by either mixing and varying the original training data, or by introducing new *seed melody* as the input. The rest of the network output are recorded as the generated music.

In 1994 Mozer's [44] fully connected RNN was trained by minimizing the log-likelihood of the L2 norm of the predicted and actual output using BPTT. The outputs of the final layer are considered as a probability of whether the note should be on or off. In addition, to enhance model harmonic relationship of musical notes, Mozer proposed a grey-code like representation that encodes notes based on their location on chromatic circle, circle of fifths and pitch height, a psychologically based representation derived from Shepard [45]. To compensate via backpropagation through time trained RNN's inefficacy of learning long-term dependencies, a similar encoding scheme is used to represent durations based on three fraction scales.

Todd's network was adopted by Franklin [46]. Additionally, he then added a second training phase whereas the network was further trained via reinforcement training. In the reinforcement learning phase a scalar value was calculated by a set of *music rules* to determine the quality of the output. This scalar value was then used to replace the explicit information of error.

To deal properly with the vanishing gradient problem, Eck and Schmidhuber [47] used two long short-term memory networks. One for learning memory and the other for learning chords. The output of the chord network system was connected to the input of the melody network. This kind of system was able to learn the standard 12-bar blues chord sequences and it was able to generate music notes that follow the chords. LSTM networks were used by Franklin to learn Jazz music [48]. These networks developed a representation scheme of pitch based on major and minor thirds. Also, circles-of-thirds representation and inspired by Mozer [44], circle-of-fifths pitch representation. In the research,

duration representation was expanded by dividing note durations into 96 subdivisions each representing a *tick* in the MIDI protocol.

In order to describe correlated musical pattern among multiple notes, Boulanger-Lewandowski et al. [50] developed and RNN based model by using restricted Boltzmann machine (RBM) [59] and a recurrent temporal RBM (RTRBM) [60]. System of type RNN-RBM allows freedom in describing the temporal dependencies of notes. It is also believed to be able to model unconstrained polyphonic music in a piano-roll representation without any reduction of dimension.

More recently, Huang and Wu proposed a 2-layer LSTM that like Boulanger-Lewandowski et. al. produces music that is more complex than a single note sequence. It is able to produce chords as well [39]. The novelty of the work by Huang and Wu was that they integrated knowledge from music theory to build neural architecture and produced multi-track pop music – melody, chords and drums. Furthermore, researchers feel that more work could be done in the development of a better evaluation metric for the quality of a piece. They believe that only then we will able to train models able to compose truly original music.

Zhen Sun et al. as well used LSTM models to compose music [9]. In order to produce more music according to music theory they augmented the dataset by adding the generated music by their LSTM model. Johnson [60] used a biaxial RNN one axis for time and the other for notes drawing inspiration from the RNN-RBM model.

There is also a BachBot [61]. It is a LSTM based approach specially designed to deal with the chorales by Bach. This kind of approach relies on little musical knowledge – all of chorales are transposed in a common key. It is able to produce quite high quality harmonizations.

The most recent LSTM approach that produced remarkable results is that of Sony CSL. They tried to create an artificial Bach that created new Bach chorales. The difficulty of this task from a compositional point of view comes from the intricate interplay between harmony and voice movements. Moreover, each voice has its own *style* and its own coherence. Finding a chorale-like reharmonization which combines Back-like harmonic progressions with musically interesting melodic movements is a problem which usually takes years of practice for musicians. In the paper [15] they introduce DeepBach as an LSTM based model capable of reproducing musically-appealing four-part chorales in the style of Bach. Contrary to other models based on RNNs, they did not sample from left to right and modelled each voice separately. This allowed them to enforce user-defined constraints such as rhythm, notes, parts, chords and cadences. DeepBach is able to produce coherent musical phrases and provides, for instance, varied reharmonizations of the melodies without plagiarism. Its core features are its reliance upon no knowledge, its speed, the possible interaction with users and the richness of harmonic ideas that it proposes.

Google Magenta uses a number of machine learning algorithms for the creation of artificial music. Next to LSTMs, computer vison and others are used, yet the LSTM approach seems to be a leading technology at the moment.

2.6.2. Echo state networks usage for music composition

Echo state networks (ESN) seem promising since they can have quite a lot of memory and a lot of neurons since it is computationally cheap to train them. It has almost never been used for music composition. There is only one article found [64]. In the article ESN is used as a rhythm generator and the authors conclude that ESN is a good choice to learn rhythm patterns.

2.6.3. Choice of the network

Summarizing, long-term dependency in the neural networks is needed to generate quality music. Otherwise, neural networks are able to generate melody yet no harmony, i.e. the music gets stuck at some point or turns out to be repetitive. LSTMs are better than traditional RNNS in this case since they have a stronger long-term dependency. Though fine-tuned LSTM algorithms are able to overcome the obstacles that traditional RNN algorithms confront, they still face the same problems in a way that the music lacks the theme, i.e. the *big picture*. Long short-term memory algorithms have been extensively studied in the recent years. Besides, LSTM algorithms are also heavy and require a lot of resources. To add an advantage for ESNs, they can have a huge reservoir of nodes. Having a bigger reservoir provides them more memory since the reservoir nodes are interconnected. ESNs also have a leaking rate parameter which regulates the update speed of the state [65]. We have also been looking for a light-weight solution.

For these reasons, we chose to work with a type of recurrent neural networks – echo state network (ESN) – that have barely been researched for musical composition.

3. EXPERIMENT

3.1. Choice of music

As it was mentioned in the introduction, we chose to work with notes for it is a lot less complex. It is a lot easier for us to understand and analyse it as well as it is for the algorithm in the means of computational resources and dependency on previous notes. We are also able to compare it with the musical theory and it helps us stay in the realm of classical music as well. It was also relatively easy to find on the internet, especially piano, while other types of music were not easy to find.

3.2. Echo state network

Echo state networks supply an architecture and principles of supervised learning for recurrent neural networks. The idea behind an ESN is to drive a large, random and fixed reservoir of neurons with the input signal (Fig. 11). Thence, inducing each neuron within it with a nonlinear response signal. After, combine the desirable output data by a trainable linear combination of all of these response signals [66]. In practice, it is important to keep in mind that the reservoir acts not only as a nonlinear expansion, but also as a memory input at the same time [65].

Echo state networks use a clever trick to make it much easier to learn a recurrent neural network. They initialise connections in the network in such a way that it has a huge reservoir of coupled oscillators. Thus, it converts input into the states of the oscillators. Then you can predict the outcome you want from the sense of these oscillators. The only thing it has to learn is how to couple the output to the oscillators – learn the W^{out}. This entirely gets rid of the issue of learning hidden to hidden connections or even input to hidden connections. However, to get these networks to be good at tasks, one needs a very big hidden state [68].

The main advantage of an ESN computationally is that only the last layer has to be learned which is a linear model that uses the transformed inputs to predict the outputs. It is much faster to learn a linear model. Therefore, we can afford an ESN with a lot of nodes.

3.2.1. Systems equations

Systems equations for the echo state network are described as following. Here the basic discrete-time sigmoid-unit echo state network with *N* reservoir units, *K* inputs *L* outputs is governed by state update equations of this kind (2):

$$\boldsymbol{x}(n+1) = f(\boldsymbol{W}\boldsymbol{x}(n) + \boldsymbol{W}^{in}\boldsymbol{u}(n+1) + \boldsymbol{W}^{fb}\boldsymbol{y}(n)) \qquad (2)$$

Here $\boldsymbol{x}(n)$ is the *N*-dimensional reservoir state, f is a sigmoid function – usually the logistic sigmoid or the hyperbolic tangent function (Fig. 9, Fig. 10). $\boldsymbol{W}$ is the N×N reservoir weight matrix,

$\boldsymbol{u}(n)$ is the K dimensional input signal, $\boldsymbol{W}^{fb}$ is the $N \times L$ output feedback matrix and $\boldsymbol{y}(n)$ is the L-dimensional output signal. In tasks where no output feedback is required, $\boldsymbol{W}^{fb}$ is nulled. The output is obtained from the extended system state the equation (3):

$$\boldsymbol{y}(n) = g(\boldsymbol{W}^{out}\boldsymbol{z}(n)) \tag{3}$$

where g is an output activation function and $\boldsymbol{W}^{out}$ is a $L \times (K+N)$-dimensional matrix of output weights. Extended system state $z(n)=[x(n); u(n)]$ at time n is the concatenation of the reservoir and input states. [66].

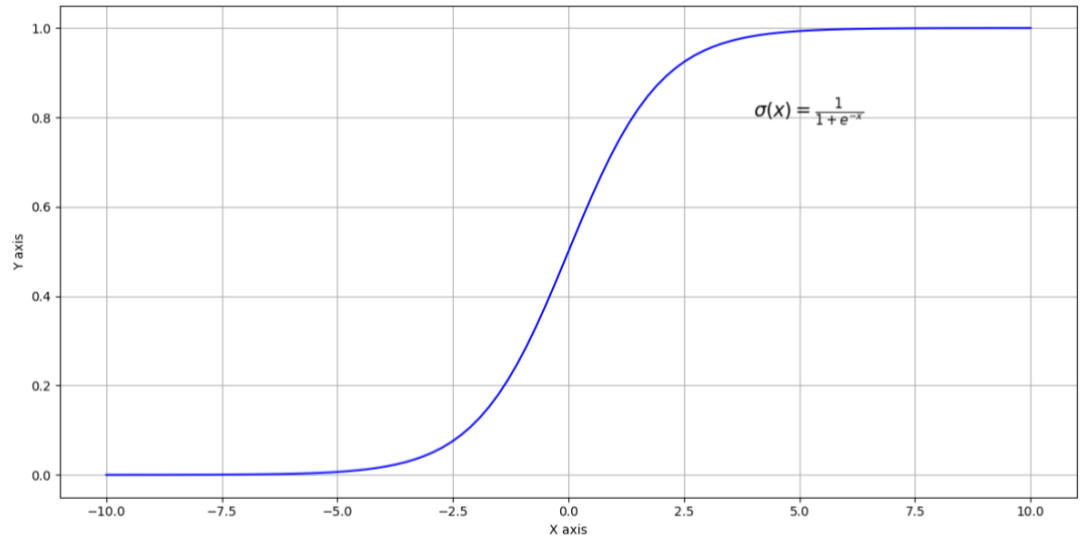

Fig. 9. Logistic sigmoid function

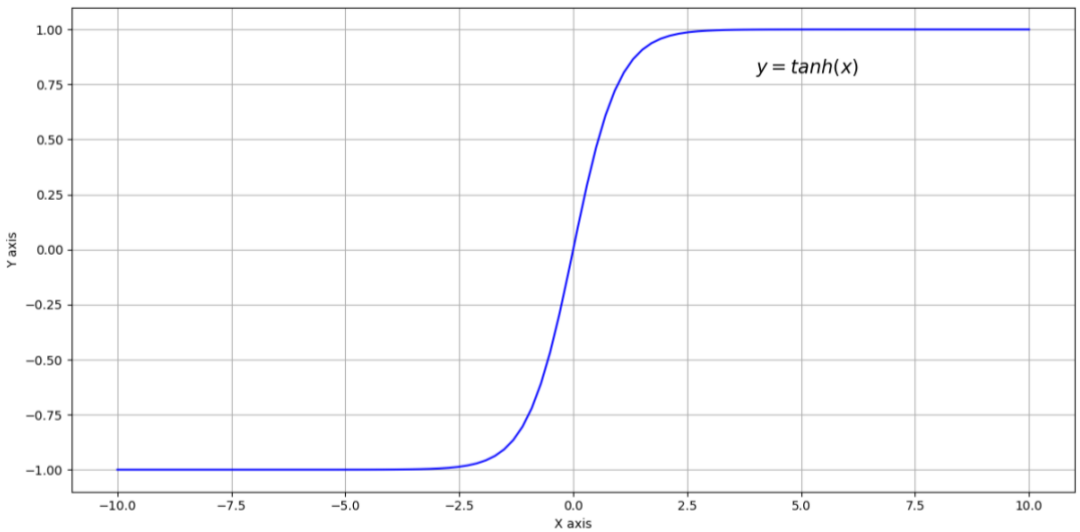

Fig. 10. Hyperbolic tangent function

3.2.2. Ridge Regression

Since readouts from an ESN are typically linear and feed-forward the equation for mapping input to outputs can be written as (4):

$$\boldsymbol{Y} = \boldsymbol{W}^{out}\boldsymbol{X} \tag{4}$$

where $\boldsymbol{Y}$ are all the $\mathbf{y}(n)$ *and* $\boldsymbol{X}$ are all [1; $\boldsymbol{u}(n)$; $\boldsymbol{x}(n)$]. Here we use a single $\boldsymbol{X}$ instead of [1; $\boldsymbol{U}$, $\boldsymbol{X}$] for notation brevity [65].

Probably the most universal and stable solution to (4) in this context is ridge regression. It is also known as regression with Tikhonov regularization [65] (5):

$$\boldsymbol{W}^{out} = \boldsymbol{Y}^{target}\boldsymbol{X}^{T}(\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}} + \beta\boldsymbol{I})^{-1} \tag{5}$$

where β is the regularization coefficient and $\boldsymbol{I}$ is the identity matrix.

3.2.3. Evaluation of the model

An echo state network may be evaluated by comparing the output signals to the actual and printing out the root mean squared error (RMSE) of this difference. It can be described with the formula (6).

$$E(\boldsymbol{y}, \boldsymbol{y}^{target}) = \frac{1}{N_y}\sum_{i=1}^{N_y}\sqrt{\frac{1}{T}\sum_{n=1}^{T}(y_i(n) - y_i^{target}(n))^2} \tag{6}$$

Here y is the output of the model and y^{target} is the actual output signal. Then n=1, …, T is the discrete time discrete time and T is the number of data points in the training dataset.

3.2.4. Echo state network tuning

Echo state network can be tuned by altering the following parameters:

1. leaking rate
2. input scaling
3. spectral radius
4. regularization

Leaking rate of the network can be regarded as the speed of the *reservoir* update dynamics in discrete time.

The update equations of a leaky-integrated ESN look like this (7, 8) [65]:

$$\tilde{\boldsymbol{x}} = f\left(\boldsymbol{W}^{in}[1; \boldsymbol{u}(n)] + \boldsymbol{W}\boldsymbol{x}(n-1)\right) \tag{7}$$

$$\boldsymbol{x} = (1-\alpha)\boldsymbol{x}(n-1) + \alpha\tilde{\boldsymbol{x}}(n) \tag{8}$$

where α is the leaking rate and f is the activation function element-wise.

Another key parameter to optimize an ESN is the input scaling. It multiplies the input weight matrix W^{in} by its value either strengthening the input weights or diminishing them.

Spectral radius is one of the most global parameters of an ESN, i.e. the maximum absolute eigenvalue of the reservoir weights matrix W. It scales the matrix W, or in alternate words, scales the width of the distribution of its nonzero elements [65].

Regularization is not really a parameter that tunes the echo state network model but it is used in order to avoid overfitting of the model.

It has been noticed that having some of the reservoir weights matrix elements equal to zero, shall provide coupled states with some inner knowledge. This can lead to a better prediction model [68].

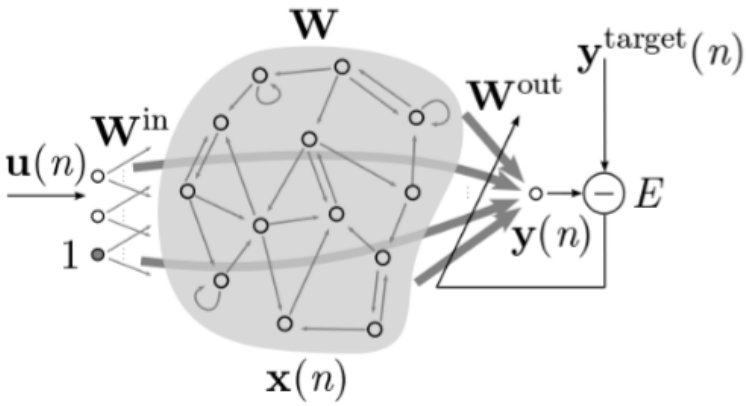

Fig. 11. Design of an echo state network [65]

In Fig. 11 *u* is the input data, W^{in} is the input weights matrix, x is the *reservoir* nodes and their outputs, *W* is the weight matrix of the network cluster, W^{out} is the output weights and y is the output data.

The number of neurons inside the *reservoir* has been opted to be equal 1000.

Programming code for ESN has been adapted from [67] and expanded for multidimensional input data and output as well as other implementations.

3.3. Data and tools

Musical data were downloaded in the format of *.mid* from the website http://www.piano-midi.de. From now on MIDI and *.mid* will be used interchangeably meaning the same, i.e. the file format unless stated otherwise, e.g. MIDI protocol. In total, 21 pieces by Mozart were gathered (all that are found on the website).

MIDI format is a sequence of notes (and commands such as tempo change and sound perturbations) whereas the time difference is represented in ticks. A quarter note is usually 480 or 960 ticks but that also depends on the resolution of the file. Thus, a full note or, in other words, a tact is 1920 or 3840 ticks respectively.

Later on, the data had to be transformed in a format that is easier to read, maintain and process. Hence, it was read and transformed into notes as messages into a *.csv* (comma separated values) format.

Every message consists of information of this type:

1. note

2. pitch
3. on tick
4. off tick
5. length

The length parameter is not in the MIDI file and had been artificially generated for the purpose of data analysis. Table 3 shows the types of information as well as their ranges in a message. Note pitch ranges from 0 to 127, thus a byte is more than enough to store it. The beginning and the end of a note tick is undetermined and can grow to infinity when the data grows. Length parameter is purely the difference between on and off ticks. It may grow to a large number due to software bugs or a divergence of the algorithm, but usually it shall stay in the realm of classical music and get a value up to a full note. We have noticed that our MIDI files have the information of hand included in the messages. 1 stands for the right hand and 0 stands for the left hand. We have later extended our research according to this new finding. It was not noticed before.

Table 3. Information inside a message

Info	Note pitch	On tick	Off tick	Length	Hand
Type	byte	long	long	integer	bit
Range	0-127	0-*infinity*	1-*infinity*	1-*full note*	0-1

A message in MIDI that signifies the event of pressing a note is the note_on message. It represents an event when a note is released as well, only the velocity then is equal to zero. The algorithm that was applied for treatment of raw MIDI files looks as following (Fig. 12).

iterate through the messages:
--check if it is a 'note_on' type of message:
----if velocity > 0:
------take the time of the note that was pressed
----else if velocity equals 0:
------check if the actual note was pressed:
--------release the note
--------measure the length of the note
----------append the note as a message to the CSV

Fig. 12. Algorithm of raw MIDI file treatment into a CSV file

The programming language of choice was Python due to its recognition in data science and machine learning among scientists and developers [69]. Python's incredible growth can be seen in Fig. 13. Also, due to the many data processing as well as machine learning libraries although none of the machine learning libraries were used for this work. For the purpose of .mid processing, Mido library was chosen [64].

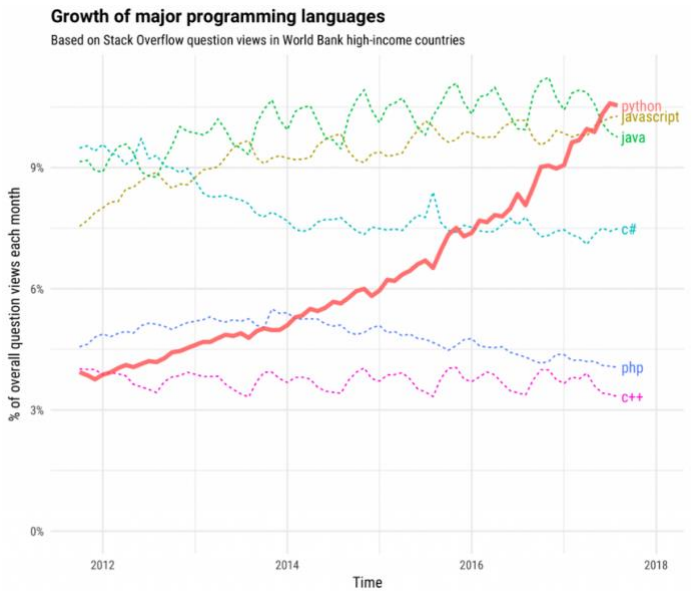

Fig. 13. Growth of major programming languages in high-income countries [69]

Machine learning algorithms perform better under more data. We could have just taken in all of the composers from the website full of classical MIDI files, but we chose only one for the purpose of thorough analysis. Despite the fact that our choice was only Mozart's music and that had given us only 21 pieces of scores, this resulted in around 68 thousand notes.

3.4. Initial data analysis

Prior to the research, an analysis of the data was performed based on the distribution of note pitches as well as their lengths. As we can clearly see in Fig. 14, there are 2 maximums. One is of a higher pitch while the other is of a quite lower pitch. This is most probably due to the fact that piano is played by 2 hands and that the left hand usually wanders in the region of lower pitch notes whilst the right hand sits in the region of higher pitch notes.

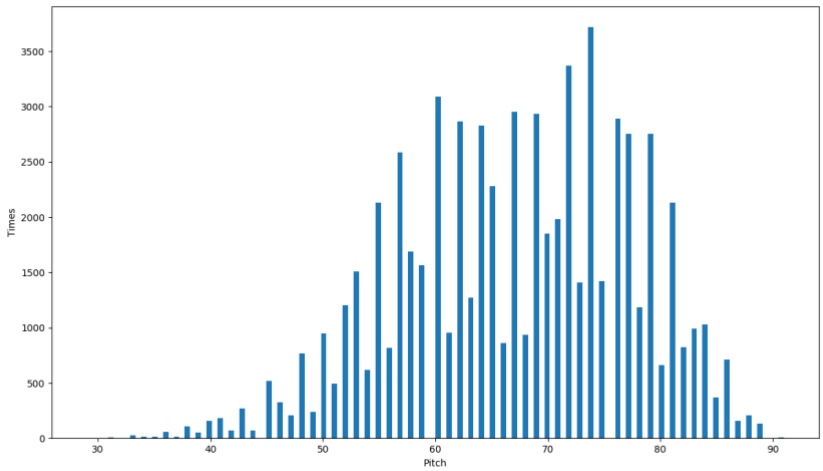

Fig. 14. Distribution of Mozart notes

These data are not so much relevant for our research, but provide us with insights such as it would make perfect sense to study the hands in more detail. We ought to bolster our research either by adding an additional dimension of the hand or by having 2 different outputs for each hand by the network. This analysis is also useful for future comparison and judgment of generated music.

Analysis of lengths (Fig. 15) provide us only one maximum, meaning both hands share the same maximum or that the note lengths of one hand are very dispersed.

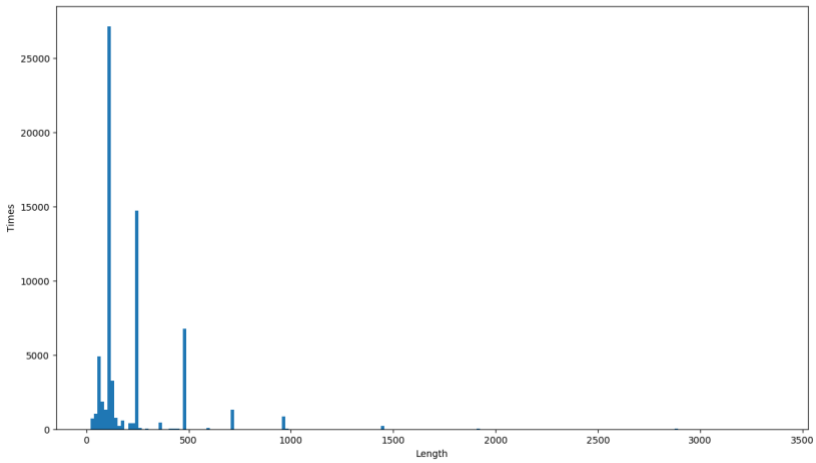

Fig. 15. Distribution of Mozart note lengths

3.5. Experimental setup

3.5.1. Predicting music

Research has been accomplished in a manner that can be seen in Fig. 17. First of all, music was accumulated in *.mid* format (hex code). As stated before, it was processed by Mido library and stored in a *.csv* format in a form of messages that carry the information of notes as the pitch number, on and off ticks and length.

Then the messages were read from the *.csv* file and quantized. Quantization was performed for the beginning and the end of the notes in the following way. A quantization unit of 60 ticks (represents a 32nd of a note) was chosen. Next, if the residual value of the tick was less than half the quantization unit, it was reduced by the residual. If the residual value was equal or higher than half of the quantization unit, i.e. 30 ticks, it was increased by the difference between the quantization unit and the residual. The lengths of notes were recalculated afterwards.

In Fig. 16 we can see the distribution of the notes after quantization. Hereby, the number of notes of the length of the quant (60 ticks) has increased. The most frequent note stayed the same (120 ticks). Also, a tiny part of the very shortest notes was quantized to zero length, thus, eliminated.

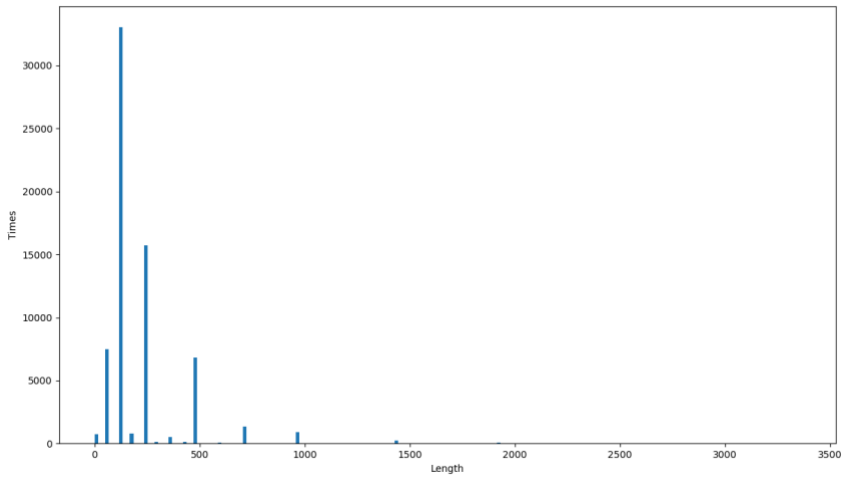

Fig. 16. Lengths of quantized notes whereas the quantization is 60 ticks

As a further step, these quantized music messages were turned into a state matrix of length that is equal to the division of the total length of the pieces by the quantization unit rounded to integer. Another dimension of the state matrix were the note pitches, that is 128 values in total. Then the value at each time step at a certain note represents its state (1 for pressed and 0 for not pressed). 80% of the data were sent to the echo state network whilst 20% were used for validation of the model, thus finding out the error. Error was calculated in the shape of root mean squared error (RMSE).

An ESN was generated according to given parameters. This ESN was then trained on input and predicted music based on its learned weights as a one time-step prediction. The training process was initialized by 300 time steps, that is by 300 quants (60 ticks).

To find out the best parameters for our echo state network, we would repeat the procedure of generating the network according to different parameters and training the new network model on the very same data. Then we predicted next notes based on the newly gained weights and found out the error by comparing with the original Mozart data. Prediction of notes was a sequel of the training process. To be more precise, the model predicted notes as a one time-step prediction. Summarizing, a grid search analysis of 4 parameters of the echo state network has been performed.

Parameters that have been investigated for tuning our network are the following. Leaking rate, input scaling, spectral radius and regularization which are the most important ESN parameters explained in Chapter 3.2. Since their ranges usually go from 0 to 1, 0 to 2, 0 to 2 and almost anything respectively, they have been tested for values in these ranges. An exhaustive grid search analysis had been performed looking for the best parameters. In addition to RMSE, mean and standard deviation were calculated. Original Mozart music had the mean of 0.04238 and standard deviation of 0.0779. Mean represents the probability of a note to played at each time step in the note spectrum. In Mozart's

case note spectrum is from the 29th to the 91st note. Standard deviation represents the mean of standard deviations of the notes in the note spectrum.

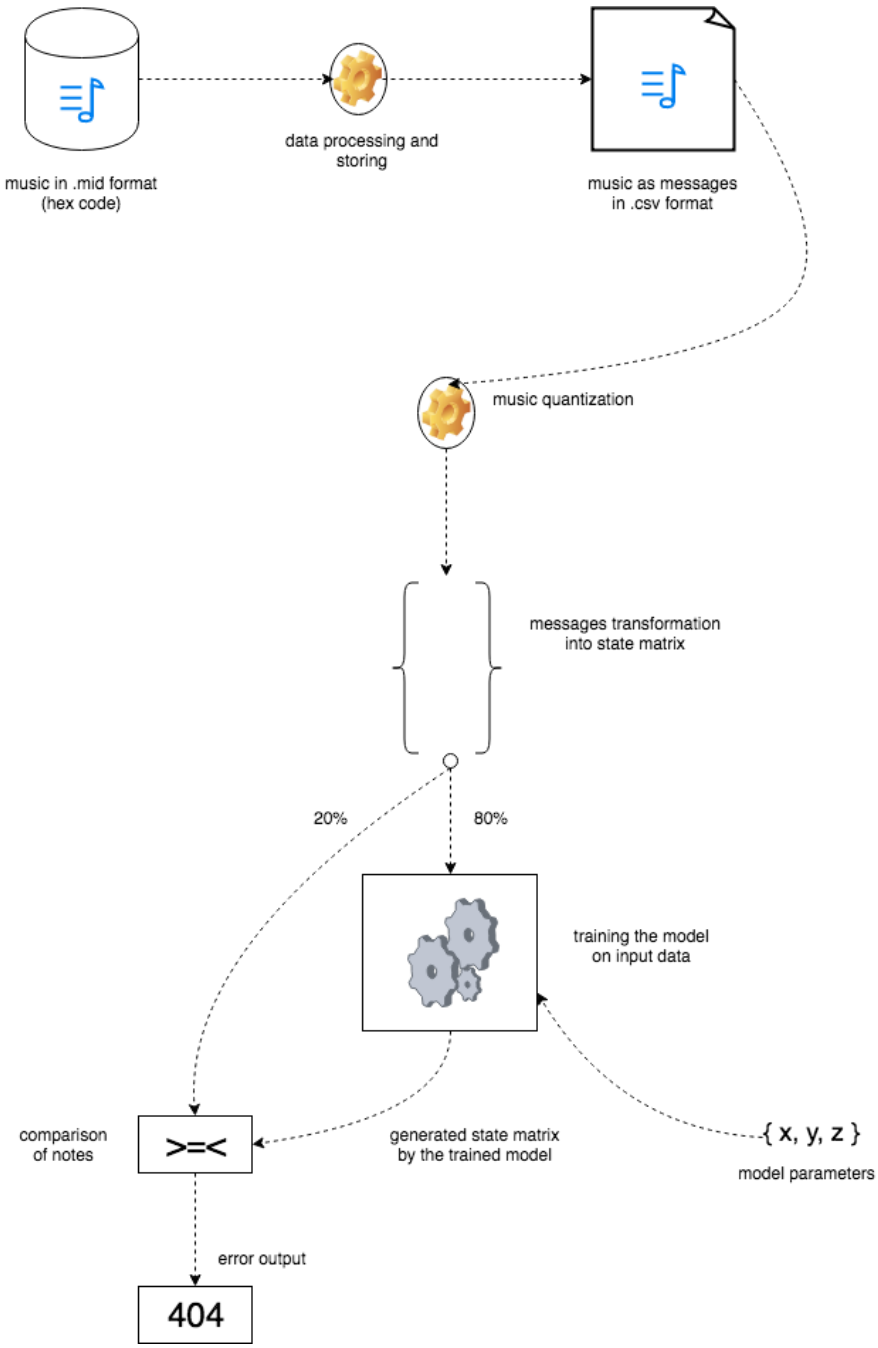

Fig. 17. Scheme of research

Leaking rate has been tested from 0.0025 to 1, spectral radius varied from 0.0015 to 2 in this test, input scaling from $2*10^{-6}$ to 2 and regularization from 10^{-6} to 10^{5}.

Firstly, the activation functions for the echo state network used were hyperbolic tangent (tanh) and the range of the input weights matrix W^{in} was from -0.5 to 0.5 times the input scaling and the of the reservoir nodes matrix W was from -0.5 to 0.5. Later, we came to the idea that since out output is 0 to 1, it may be better to change the activation function into logistic sigmoid and shift the weights up

to 0 to 1 times the input scaling for the W^{in} matrix and 0 to 1 for the W matrix. This observation, as it turns out, actually gives away a little bit better results.

3.5.2. Predicting music including the hand information

We mentioned earlier that after a while we found out about hand information inside the MIDIs, i.e. with which hand was the note played. Usually, the left hand sits in the lower region of pitches compared to the right hand. To add this in our model, we have mapped notes played with right hand as in a state 1 in the state matrix and notes played with left hand as -1. This should give a hint about which hand pressed the key to the network. Surprisingly, this produced us even better results. These were the statistics by using this kind of state matrix. 0.00020608 and 0.07018343 for mean and standard deviation respectively.

3.5.3. Generating music

Generating music using our algorithm from a user's/programmer's point of view would look like in Fig. 18. At the moment the model doesn't not save weights nor get the saved weights since there was no need for this as we were looking for the best parameters of the model. Yet if this algorithm was used and reused, saving of the weights ought to be implemented not to redo all of the computations. For now, it always trains the model from scratch. User only has to choose music and run a command. As an output, he gets a MIDI file generated according to the type of chosen music.

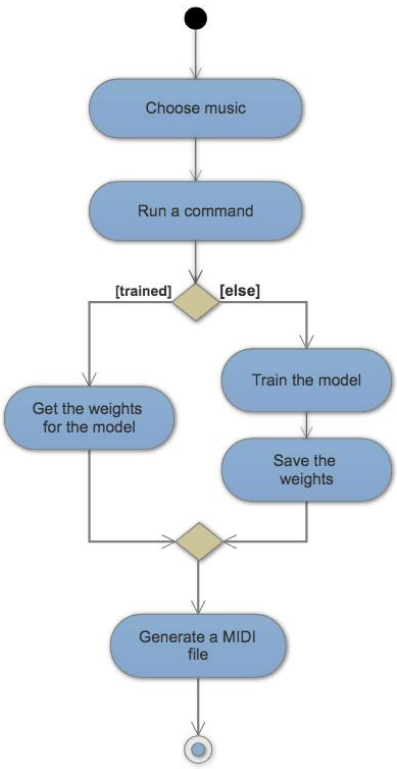

Fig. 18. Generating music from user's/programmer's point of view

No class diagram is given since the code was written using function only. The code is open source and can be found on https://github.com/AzisK/Experiments-in-Music-Creation.

To generate music, we used the very same network except that now we quantized the notes. We transformed the signals into a note or a pause according to rules that we thought could help predict classical music. We added a random sampling and looked for the best results.

To implicitly give the network some kind information about the chords. Only the notes that are away by 3 or 4 half tones away from the strongest and then from the others added are played, since most of the chords have a distance of 3 to 4 half tones from the notes. Although there are some chords that have 2 half tones distance, we did not include this but it could happen even from this rule.

We have also added the maximum number of notes played at the same time by one hand up to 6 since this is the highest number that classical music pieces usually have.

We also increased the probability of the note being pressed at a certain time step if it was pressed before. This was done by trial and error.

4. RESULTS

4.1. Prediction of music

As we can see from the sorted by error (top 10) Table 4, the lowest value of error (RMSE) is a tiny bit above 0.0307. It is clear that the best leaking rate for our model is about 0.025 while the combination of input scaling and spectral radius vary a little bit. Input scaling goes from 0.002 to 0.0002 and spectral radius from 0.01 to 0.1. We can notice that while RMSE is the lowest, the mean of the notes is about the same of the quantized original Mozart music data mean but standard deviation is quite different.

Table 4. Sorted error data (top 10)

leaking rate	input scaling	spectral radius	reg	mean	rmse	std
0.025	0.0002	0.1	0.0001	0.0410	0.030769	0.05696
0.025	0.0005	0.01	0.001	0.0411	0.030771	0.05699
0.025	0.0005	0.01	0.001	0.0411	0.030771	0.05698
0.025	0.0002	0.02	0.0001	0.041	0.030772	0.05697
0.03	0.0005	0.1	0.001	0.0411	0.030772	0.05699
0.025	0.0005	0.05	0.001	0.0411	0.030773	0.05699
0.03	0.0005	0.06	0.001	0.0411	0.030774	0.05699
0.02	0.0006	0.02	0.001	0.0411	0.030775	0.05697
0.015	0.0006	0.02	0.001	0.0411	0.030776	0.05697
0.02125	0.002	0.1	0.01	0.0410	0.030776	0.05697

reg stand for regularization, *rmse* stands for RMSE and std stands for standard deviation in the tables of error (Table 4, Table 5, Table 6).

Having low leaking rate suggests us that the state has a lot of *inertia* and the change of the state is slow. Input scaling scales the W^{in} matrix, thus, the input weights are very low and the model depends on its input just a tiny bit. Since it is lower than the spectral radius, it has a lot of memory, i.e. follows a long-term dependency. Having low spectral radius as well tells us that the models are almost linear. To summarize, the prediction function is not very complex and the model has a lot of memory.

From Table 5 we see that a high regularization value gives us huge errors. It has to be noted that for this particular grid search step, the maximum value of input scaling and spectral radius was 0.2. Thus, we can also deduce that high input scaling values lead to higher error. Though leaking rate is not as important, we can still see that some of its higher values lead to higher errors.

High regularization significantly reduces the mean value and standard deviation of the notes.

Table 5. Sorted error data (worst 10)

leaking rate	input scaling	spectral radius	reg	mean	rmse	std
0.175	0.2	0.2	100000	0.0345	0.064397	0.00959
0.175	0.2	0.14	100000	0.0357	0.064656	0.00952
0.1	0.2	0.02	100000	0.035	0.064968	0.00801
0.1	0.2	0.2	100000	0.0352	0.065097	0.00817
0.1	0.2	0.14	100000	0.0353	0.065158	0.00806
0.1	0.2	0.08	100000	0.0354	0.065258	0.00808
0.025	0.2	0.02	100000	0.035	0.066049	0.00572
0.025	0.2	0.14	100000	0.0352	0.0662	0.00584
0.025	0.2	0.08	100000	0.0353	0.066243	0.00585
0.025	0.2	0.2	100000	0.0354	0.066295	0.00582

In order for us to see tendencies beyond regularization, we filtered the data for regularization below or equals 10. This brought us back to the maximum values of input scaling, spectral radius and leaking rate.

In Table IV we see that high input scaling produces high error once again. Interestingly, leaking rate stays at 0.25 for the highest error. Although spectral radius stays quite high, it is not of the highest value for the highest error. Mean is almost as with the best results. Standard deviation is higher in this case than with the best results. It is even closer to quantized Mozart's music standard deviation than the one provided with the best results.

Table 6. Sorted error data (worst 10) while regularization is smaller or equal to 10

leaking rate	input scaling	spectral radius	reg	mean	rmse	std
0.25	2	0.8	1	0.0405	0.043065	0.06195
0.25	2	0.8	0.1	0.0405	0.043094	0.06198
0.25	2	1.4	0.01	0.0405	0.043098	0.06199
0.25	2	1.4	1	0.0406	0.043128	0.06173
0.25	2	1.4	0.1	0.0406	0.043139	0.06175
0.25	2	1.4	0.01	0.0406	0.043142	0.06175
0.25	2	1.4	10	0.0406	0.043169	0.06171
0.25	0.8	1.4	1	0.0413	0.043178	0.06141
0.25	0.8	1.4	0.1	0.0413	0.043234	0.06145
0.25	0.8	1.4	0.01	0.0413	0.04324	0.0615

Fig. 19 shows us the minimum error dependency on leaking rate. It is worth to note that although leaking rate 0.25 yields worst results when regularization is not high, it may also yield very good results with other values of ESN parameters as can be seen in Fig. 19.

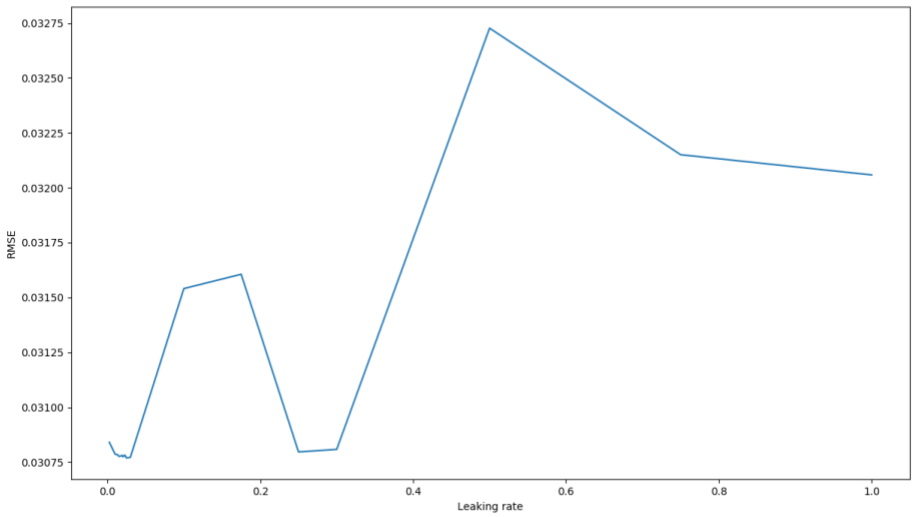

Fig. 19. Minimum RMSE dependency on leaking rate

The errors were grouped by leaking rate and the minimum value of the error was taken to plot the dependency graph. In Fig. 20 we can see the most promising region of leaking rate for our echo state network.

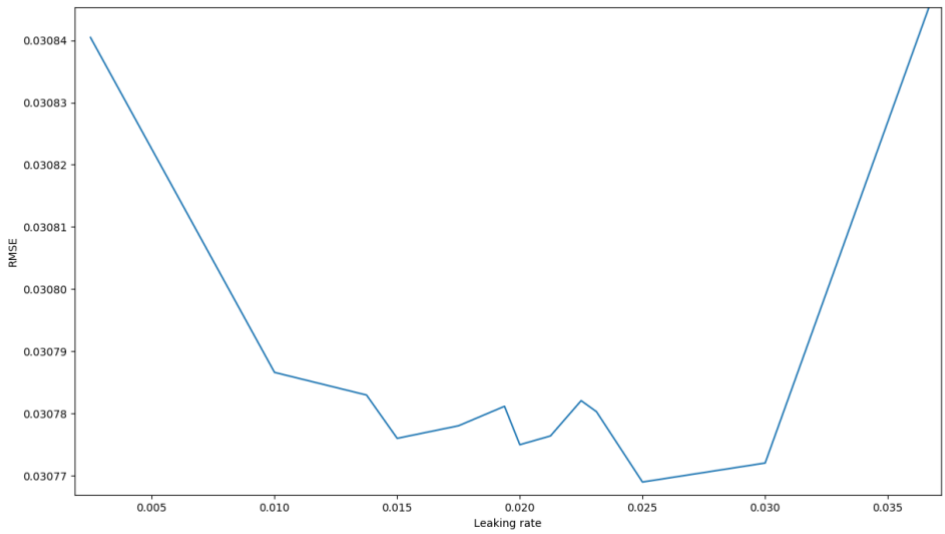

Fig. 20. Zoomed minimum RMSE dependency on leaking rate

Fig. 21shows us the minimum error dependency on input scaling whereas Fig. 22 zooms us to the most promising region of input scaling. The best values of input scaling are 0.0002 and 0.0005. Going even lower, the values increase dramatically.

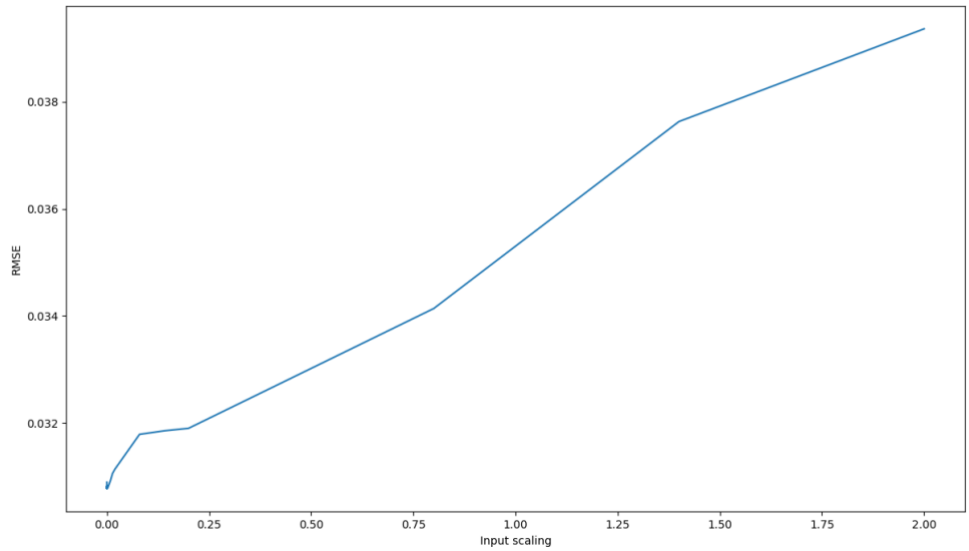

Fig. 21. Minimum RMSE dependency on input scaling

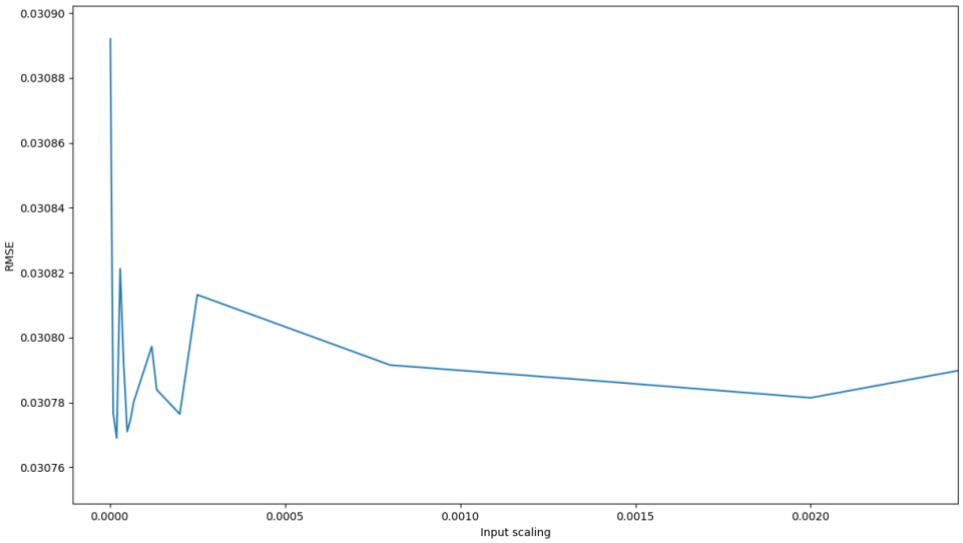

Fig. 22. Zoomed minimum RMSE dependency on input scaling

Fig. 23 shows us the minimum error dependency on spectral radius. From Fig. 24 and Fig. 25 we can see that the minimum RMSE stabilizes and reaches the minimum on spectral radius below 0.1. Then starts growing again above 0.01.

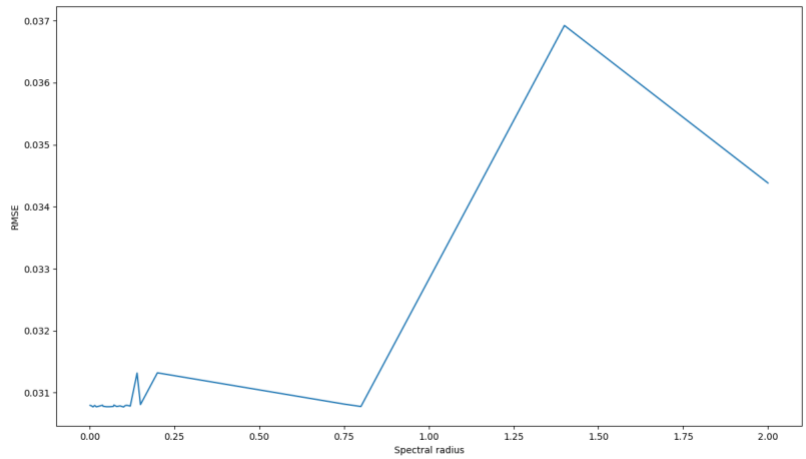

Fig. 23. Minimum RMSE dependency on spectral radius

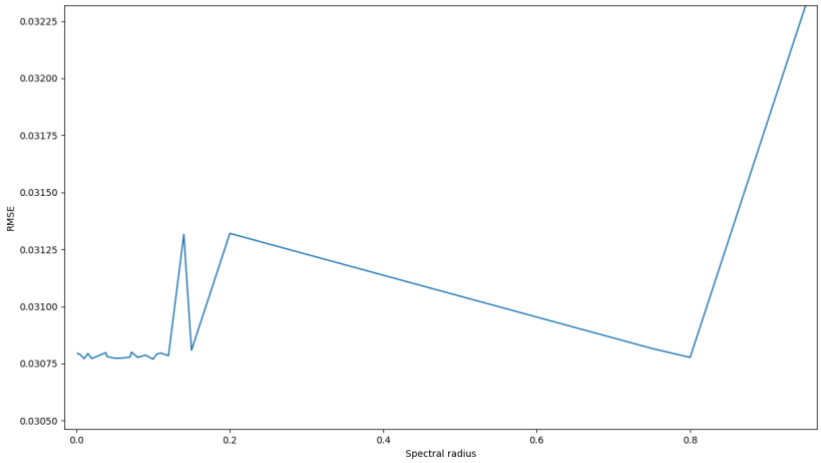

Fig. 24. Zoomed minimum RMSE dependency on spectral radius

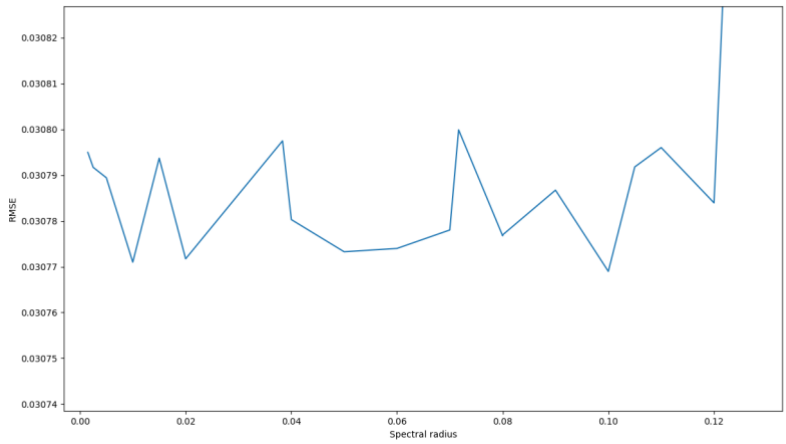

Fig. 25. Zoomed minimum RMSE dependency on spectral radius to the most promising region

Fig. 26 and Fig. 27 implies us that the best regularization values are of the power 10^{-4} to 10^{-2}.

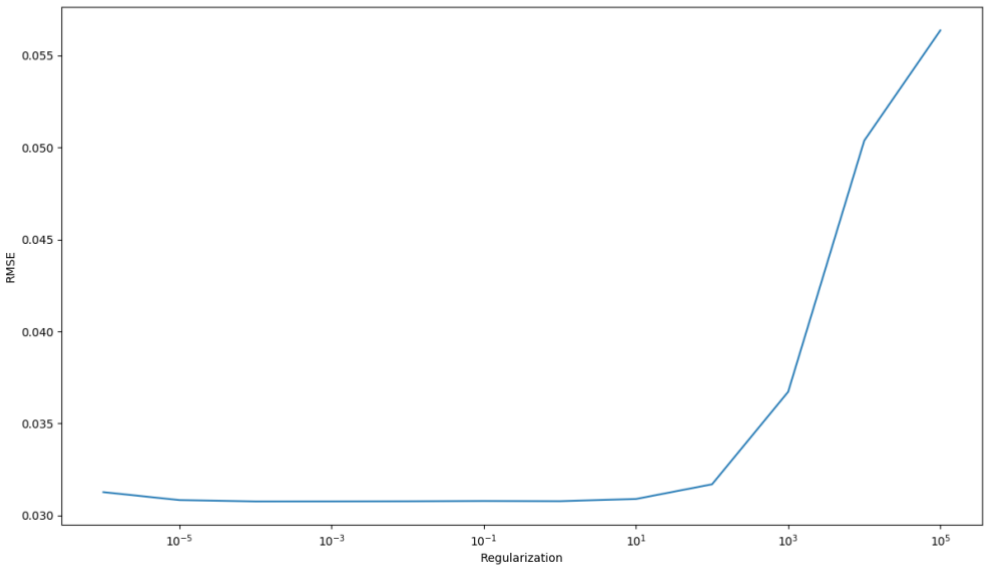

Fig. 26. Minimum RMSE dependency on regularization

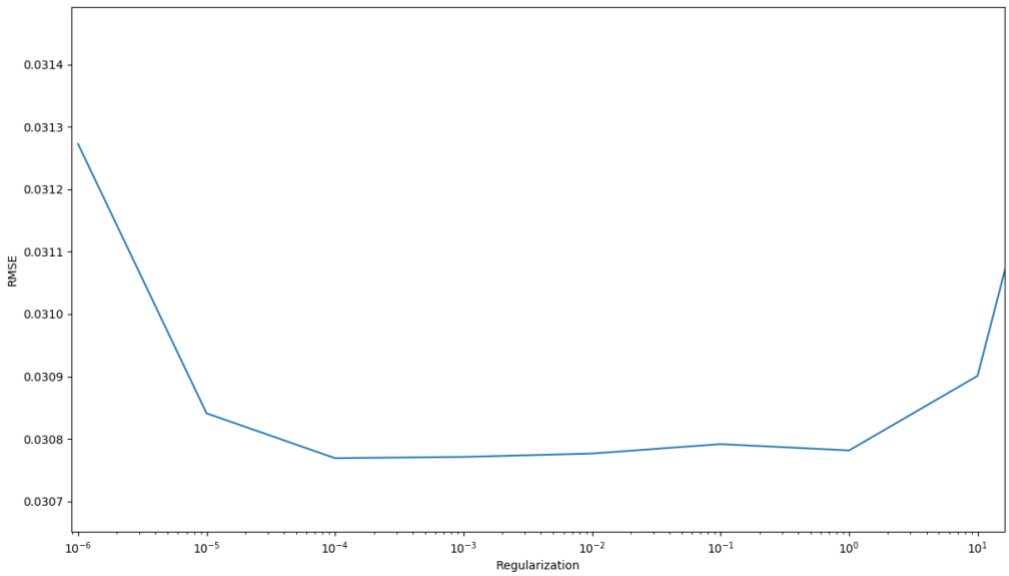

Fig. 27. Zoomed minimum RMSE dependency on regularization

Since it was a lot easier to find the optimal leaking rate than input scaling and spectral radius, we grouped the errors by input scaling and spectral radius taking the minimum RMSE value in Fig. 28. Regularization is an additional parameter that prevents overfitting and we have not grouped by it. It was quite easy to find as well.

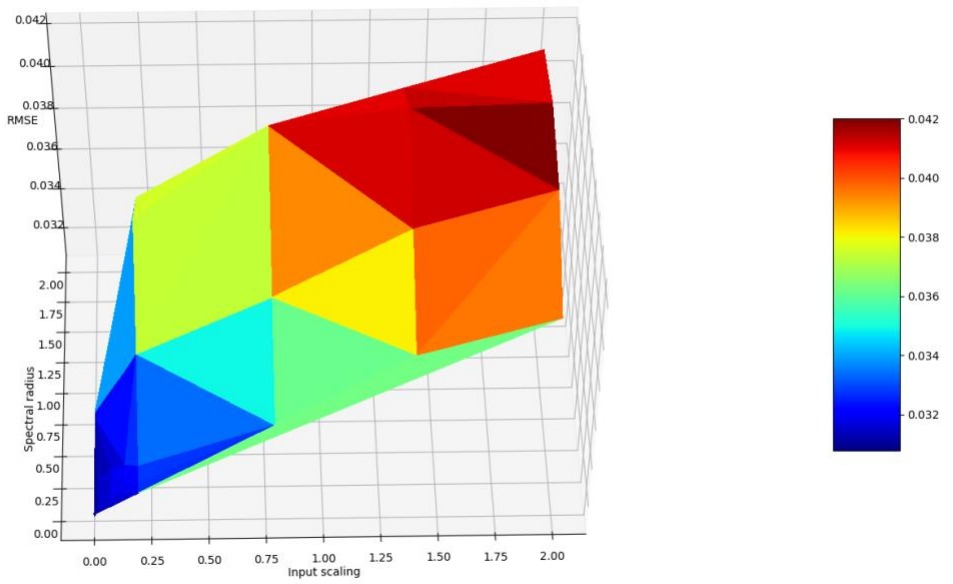

Fig. 28. Grid search minimum error (RMSE) grouped by input scaling and spectral radius

As it was found out that the most optimal leaking rate is 0.025, the errors were grouped by input scaling and spectral radius once again by a set leaking rate. Now they we grouped having leaking rate set to 0.025. In Fig. 29 we can see pointy triangles in the lower region where the errors are the lowest.

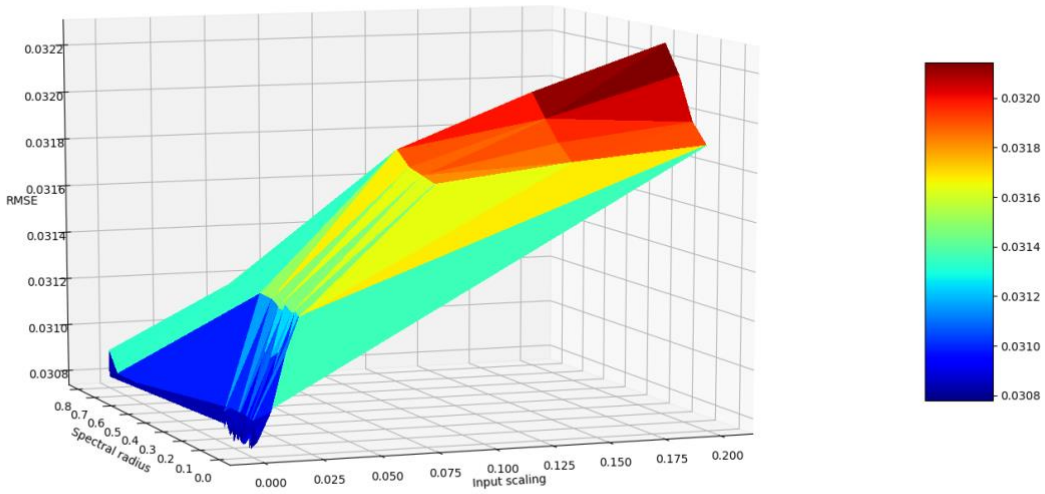

Fig. 29. Grid search minimum error (RMSE) grouped by input scaling and spectral radius while leaking rate equals 0.025.

It has to be taken into account that producing an even finer grid might give us even better results but this takes time. Also, it seems from all this analysed data that the reduction in error would be quite low.

As mentioned in the end of Chapter 3, an observation has been made that since the outputs space is from 0 to 1, it would make sense to use the logistic sigmoid instead of the hyperbolic tangent for the activation function. As it turns out, this works out quite well since the lowest RMSE drops quite a bit (Table 7). A grid search of about the same values has been carried out.

Table 7. Sorted error data (top 10) using a logistic sigmoid activation function

leaking rate	input scaling	spectral radius	reg	mean	rmse	std
0.03	0.000225	0.01	0.001	0.040972	0.030719	0.056976
0.03	0.000225	0.01	0.001	0.040997	0.030733	0.056982
0.05	0.000225	0.01	0.001	0.040960	0.030735	0.056984
0.04	0.000225	0.01	0.001	0.040982	0.030740	0.056976
0.04	0.000200	0.02	0.001	0.041040	0.030741	0.056986
0.04	0.000200	0.08	0.001	0.041039	0.030742	0.056985
0.05	0.000200	0.02	0.001	0.041033	0.030743	0.056991
0.03	0.000300	0.14	0.001	0.040847	0.030744	0.056962
0.04	0.000200	0.08	0.001	0.041034	0.030745	0.056991
0.025	0.000200	0.2	0.001	0.041052	0.030745	0.056983

Here the lowest error (RMSE) value is smaller by 0.00005. This time leaking rate is about the same but a tiny bit higher. Best results are with leaking rate 0.03. Other values stay about the same as well. This difference is very low but is quite significant in this context since we were nowhere near this using the hyperbolic tangent and different ranges of weights for the input weights matrix as well as reservoir weights matrix. We could say that having reduced the output range to the very range of the expected output as well as having restricted the weights matrixes only to non-negative numbers, increases the quality of an ESN model by a little bit.

Having noticed that the quantization mechanism deletes a part of notes as seen in Fig. 16, 721, to be exact, the mechanism was changed to lengthen the very shortest notes to the quant size. This is how the histogram of notes looks like after this kind of change (Fig. 30). Now it does not *null* any of the notes.

Having changed the quantization function and now having more notes, we ran a grid search in about the same range of values again. In Table 8 we may observe that the errors increased by a tiny bit. This seems logical as having more notes gets harder to predict them. However, we will stick with this model for music generation since we do not want to lose notes.

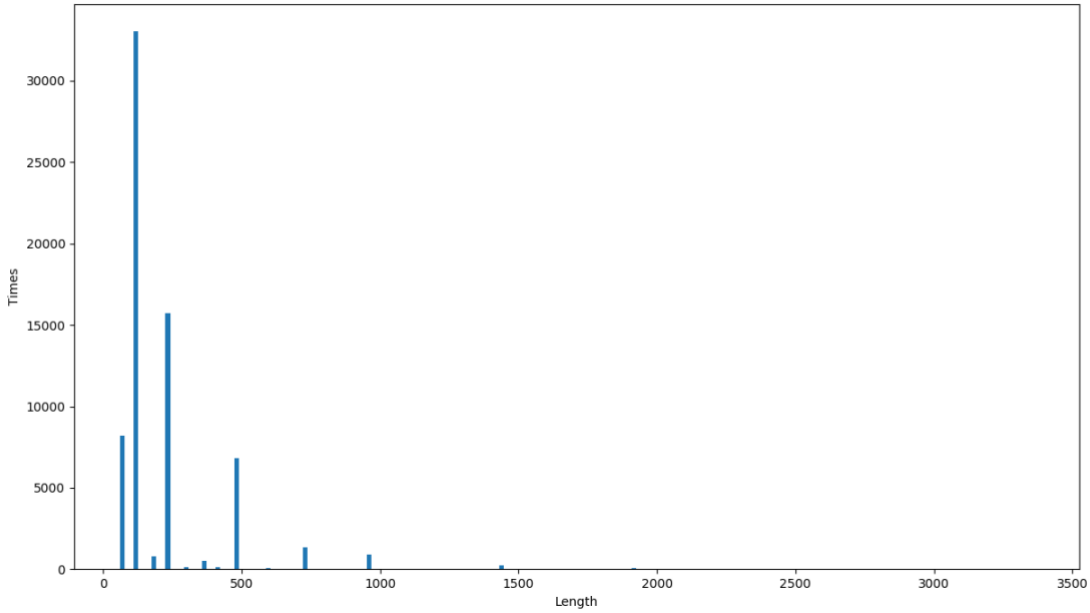

Fig. 30. Lengths of notes after change of quantization

Table 8. Sorted error data (top 5) using a logistic sigmoid and new quantization

leaking rate	input scaling	spectral radius	reg	mean	rmse	std
0.02	0.000225	0.01	0.001	0.040910	0.030737	0.056862
0.03	0.000225	0.01	0.001	0.040905	0.030746	0.056860
0.03	0.0002	0.01	0.001	0.040949	0.030749	0.056870
0.02	0.000225	0.01	0.001	0.040945	0.030757	0.056862
0.02	0.00025	0.01	0.001	0.040895	0.030757	0.056851

By randomly zeroing 10 percent of the reservoir weights matrix *W*, prediction model did not become more accurate as seen in Table 9. Here the best value of error was 0.030761.

Table 9. Sorted error data (top 5) while 10 % of the reservoir weights were reduced to 0

leaking rate	input scaling	spectral radius	reg	mean	rmse	std
0.02	0.00025	0.01	0.001	0.040894	0.030761	0.056851
0.02	0.00025	0.02	0.001	0.040898	0.030762	0.056849
0.02	0.000225	0.02	0.001	0.040944	0.030768	0.056858
0.02	0.000225	0.01	0.001	0.040972	0.030772	0.056864
0.02	0.0002	0.02	0.001	0.040997	0.030773	0.056869

4.2. Prediction of music including the hand information

Having given a hint to the neural network about the hand that pressed the piano key, we get a lot better results according to RMSE (Table 10). This time hyperbolic tangent was used as the activation function since the range of the outputs is from -1 to 1. Parameters of the best ESN model stay the same, except regularization. Regularization is 100 times bigger. Best value of RMSE is 0.029015.

Table 10. Sorted error data (top 8) including hand information

leaking rate	input scaling	spectral radius	reg	mean	rmse	std
0.02	0.00023	0.02	0.1	-0.001642	0.029015	0.052773
0.01	0.00023	0.03	0.1	-0.001643	0.029018	0.052773
0.01	0.000225	0.03	0.1	-0.001641	0.029021	0.052776
0.02	0.00025	0.02	0.1	-0.001654	0.029021	0.052767
0.02	0.000237	0.02	0.1	-0.001652	0.029022	0.052771
0.02	0.000223	0.02	0.1	-0.001648	0.029022	0.052774
0.01	0.000225	0.02	0.1	-0.001643	0.029022	0.052774
0.01	0.00023	0.02	0.1	-0.001642	0.029022	0.052772

4.3. Generation of music

Having sampled the data by different degrees of power, we get results as seen in Table 11. For comparison, the statistics for all zero output compared to original data are -0.00164782509, 0.039822453, 0.052771963 for mean, RMSE and standard deviation respectively. Therefore, even if we get lower RMSE by increasing the degree of sampling, it also comes closer to the RMSE when all states are zero. Also, we shall observe that the mean value and standard deviation decrease significantly as well. Results are getting closer to zero when applying a high degree for the sampling. Sampling works in such a way that the only values played are the ones that their values in a certain degree of power are higher than a random number.

Table 11. Sampling results

mean	rmse	std	deg
0. 00119512	0. 081092295074	0. 074045865994	1
0. 00070094	0. 052798627049	0.028107546679	1.25
0. 00042870	0. 044536801549	0. 010758374139	1.5
0. 000518302	0. 0429901646575	0. 007306312746	1.6
0. 000154387	0. 041229865806	0. 003188746708	1.8
3.9975463e-05	0. 0403228363280	0. 001287406962	2

here *deg* stands for the power of degree of the value, predicted by the ESN.

In Fig. 1 we can see the results of the generation when no sampling and no rules are applied in the first row. The result is kind of a mess. So many notes and most of them are just 60 ticks, so very short. By applying the musical rules we get a much nicer result in the second row. Other rows are when sampling is additionally increased from 1.05 to 1.1 to 1.15 to 1.2 to 1.3. As we can see, sampling does is not as important as the rules are.

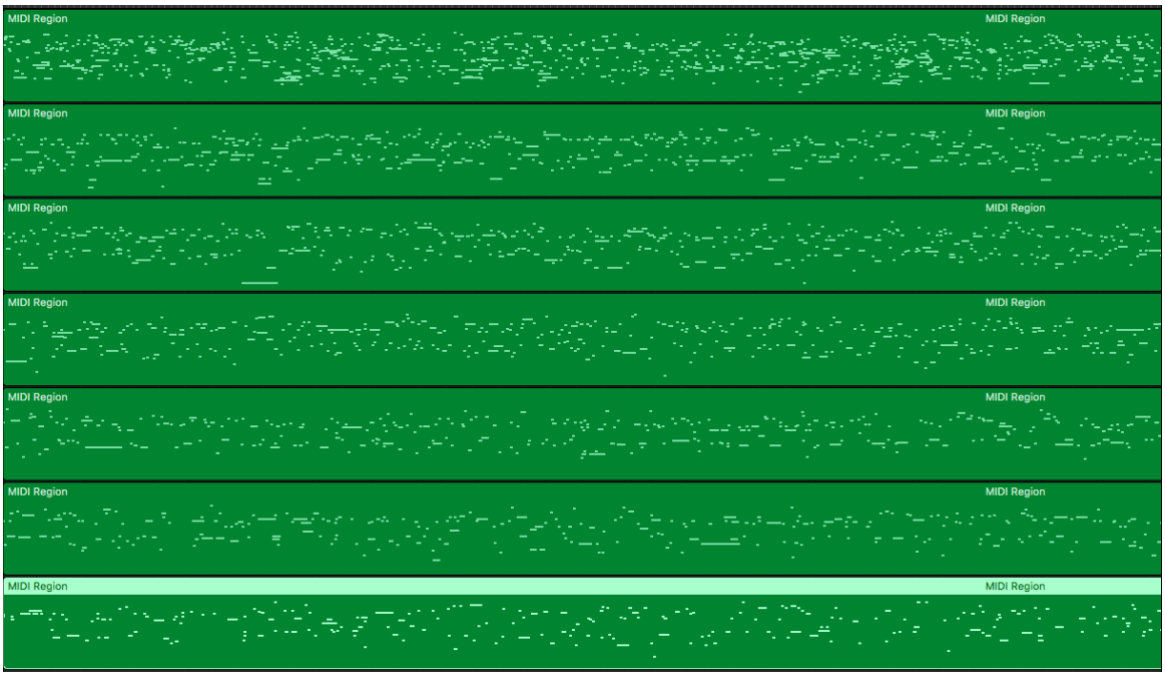

Fig. 31. Music generation results

The rule that took into account the length of the note look liked this. If the note was already pressed, it would change the degree of sampling from its initial value to (9):

$$1 + \frac{degree - 1}{(notelength\ modulus\ 8) + 1} \tag{9}$$

here *notelength* is in steps and it refers to how many steps of the note there were before the step at the moment. And then since we had most notes of ticks 120, the procedure for 2nd step would be repeated again if unsuccessful.

5. CONCLUSIONS

1. Choice of music – classical music and, to be more precise, Mozart

First of all, with classical music we can work on the note level, thus it is easier for us to understand and analyse the data as well as the predicted output, we can compare it with music theory It is also a lot easier for the network since there are less dimensions. Also, classical music, particularly piano, was easy to find on the internet. Mozart was chosen because of his genius and some haphazardness.

2. Programming language of choice – Python

Python was chosen because of its recognition in data science and machine learning among scientists and developers. Also, because of the many data processing as well as machine learning libraries although none of the machine learning libraries were used for this research.

3. Insights from the research of algorithms used for music composition.

In literature overview we have noted that the leading algorithms in music composition are sequential recurrent neural networks. For the moment of writing of this thesis, the cutting edge technology for music composition is the system of LSTM networks. Mostly because they are able to *remember* a lot, have a long term memory whereas usual BPTT RNN networks have only short term memory. As it turns out from all the research, music composition prefers long term memory.

4. Choice of algorithm – ESN (echo state network)

We have chosen ESN for several reasons. First of all, novelty. ESNs have not been used for musical composition. Only one research could be found on rhythm generation. We have chosen ESNs also because they are computationally cheap to train. Therefore, we can have a huge reservoir. A bigger reservoir provides more memory for the model. Additionally, it has a leaking rate parameter which regulates the update speed of the system.

5. Choice of evaluation for the algorithm – RMSE (root mean squared error)

The model was evaluated using RMSE as it is a common practice with neural networks.

6. Outcomes from the research of the algorithm

Having searched for the best parameters of the ESN model, we found out that the ESN has spectral radius much higher than the input scaling. This agrees with the research of algorithms outcome because this means that music is better predicted with longer term dependencies. Also, because both spectral radius and input scaling are relatively low (compared to 1), this means that the model operates in an almost linear style. Thus, the prediction function ought to be fairly simple. In addition, low leaking rate suggests that the update of the system is rather slow as well. In a way, this adds some memory too.

Using the logarithmic sigmoid as an activation function gives away slightly better results. Having to predict more notes is harder for the network in the means of the RMSE. Depending on the

quantization, the best values for the ESN were 0.02, 0.000225, 0.01 for leaking rate, input scaling and spectral radius respectively when the quantization deleted some notes and 0.02, 0.00025 and 0.01 when the quantization did not delete any of the note. Outcomes in the form of RMSE were 0.030737 and 0.030761 respectively.

7. Development of the algorithm for music composition

Giving the ESN a hint about the hand that pays the note on the piano, to be exact, changing back to the hyperbolic tangent activation and mapping right hand to the state of 1 while left hand to the state of -1, produces even better results.

It could also be said that using activation similar in form with the output can increase prediction accuracy as well.

8. Insights for music generation

We can deduct from the process of generation that sampling is not so important in this case, more important are the rules. By implying some of the music rules, the generation followed nicer results according to the music theory. Therefore, for classical music as well. The closer the rules are to the music theory, the better the results ought to be.

6. REFERENCES

1. Why do whales sing? [seen on 2018-05-24] Access on the internet: http://www.whalefacts.org/why-do-whales-sing/
2. MARTINELLI, D. Dainuojanti revoliucija – viena iš geriausiai (deja) saugomų Lietuvos paslapčių. [seen on 2018-05-24] Access on the internet: http://www.mic.lt/lt/diskursai/lietuvos-muzikos-link/nr-18-2015-sausis-gruodis/dainuojanti-revoliucija/
3. The British Invasion: From the Beatles to the Stones, The Sixties Belonged to Britain. [seen on 2018-05-24] Access on the internet: 1. https://www.rollingstone.com/music/news/the-british-invasion-from-the-beatles-to-the-stones-the-sixties-belonged-to-britain-19880714
4. G., PAPADOPOULOS, G., WIGGINS. AI Methods for Algorithmic Composition: A Survey, a Critical View and Future Prospects. *In AISB Symposium on Musical Creativity*, pp. 110-117, 1999.
5. MIDI history: chapter 1 – 850 AD TO 1850 AD. [seen on 2018-05-25] Access on the internet: https://www.midi.org/articles/midi-history-chapter-1
6. J., D., FERDINANDEZ, F., VICO. AI Methods in Algorithmic Composition: A Comprehensive Survey. *Journal of Artificial Intelligence Research*, Volume 48, pp. 513-582, 2013.
7. L., A., HILLER, L., M., ISAACSON. Musical composition with a High-Speed digital computer. *Journal of the Audio Engineering Society*, Volume 6 (3), pp. 154–160, 1958.
8. C., AMES. Automated composition in retrospect: 1956-1986. *Leonardo*, Volume 20 (2), pp. 169– 185, 1987.
9. Z., SUN, et al. Composing music with grammar argumented neural networks and note-level encoding. 2016. Access on the internet: https://arxiv.org/abs/1611.05416
10. H., CHU, R., URTASUN, S., FIDLER. Song from PI: A Musically Plausible Network for Pop Music Generation. 2016. Access on the internet: https://arxiv.org/abs/1611.03477
11. S., ENGELS, F., CHAN, T., TONG, Automatic Real-Time Music Generation for Games. AIIDE Workshop, 2015.
12. Artificial music: the computers that create music. [seen on 2018-05-24] Access on the internet: http://www.bbc.com/future/story/20140808-music-like-never-heard-before
13. Magenta: Make Music and Art Using Machine Learning. [seen on 2018-05-24] Access on the internet: https://magenta.tensorflow.org/

14. Hear the first ever full pop song by artificial intelligence. [seen on 2018-05-24] Access on the internet: http://www.factmag.com/2016/09/22/hear-first-complete-pop-song-composed-artificial-intelligence/
15. G., HADJERES, F., PACHET. DeepBach: a Steerable Model for Bach chorales. *Proceedings of the 34th International Conference on Machine Learning*, Volume 70, pp. 1362-1371, 2017.
16. Electronic. [seen on 2018-05-24] Access on the internet: http://www.allmusic.com/genre/electronic-ma0000002572
17. What is MIDI? [seen on 2018-05-24] Access on the internet: http://www.instructables.com/id/What-is-MIDI/
18. What is MIDI? [seen on 2018-05-24] Access on the internet: https://www.midi.org/
19. About MIDI-part 4: MIDI Files. [seen on 2018-05-24] Access on the internet: https://www.midi.org/articles/about-midi-part-4-midi-files
20. I., CHUSID. Time and Curiosity: Journey to the Outside. [seen on 2018-05-24] Access on the internet: http://www.keyofz.com/after/
21. M., SIMONI. Algorithmic Composition: A Gentle Introduction to Music Composition Using Common LISP and Common Music. Access on doi: https://quod.lib.umich.edu/s/spobooks/bbv9810.0001.001
22. C., FOX. Genetic Hierarchical Music Structures. [seen on 2018-05-24] Access on the internet: http://www.cs.brandeis.edu/~marc/misc/proceedings/flairs-2006/CD/07/FLAIRS06-047.pdf
23. J., A., BILES. Genjam: A Genetic Algorithm for Generating Jazz Solos. *In Proceedings of the International Computer Music Conference*, 1994.
24. G., PAPADOPOULOS, G., WIGGINS. A Genetic Algorithm for the Generation of Jazz Melodies. In *STeP'98*, 1998.
25. K., EBCIOGLU. An Expert System for Harmonizing Four-part Chorales. *Computer Music Journal*, Volume 12 (3), pp. 43-51, 1988.
26. D., CONKLIN, I., H., WITTEN. Multiple Viewpoint Systems for Music Prediction. Journal of New Music Research, Volume 24, pp. 51-73, 1995.
27. M. BODEN. What is Creativity. In M. Boden, editor, *Dimensions of Creativity*, pp. 75–118. MIT Press, 1996.
28. J., ROWE, D., PARTRIDGE. Creativity: A survey of AI approaches. *Artificial Intelligence Review*, Volume 7, pp. 43–70, 1993.
29. P., KUGEL. Myhill's Thesis: There's More than Computing in Musical Thinking. *Computer Music Journal*, Volume 14 (3), pp. 12–25, 1990.
30. J., MYHILL. Some Philosophical Implications of Mathematical Logic: Three Classes of Ideas. *Review of Metaphysics*, Volume 6 (2), pp. 165–198, 1952.

31. E., M., GOLD. Limiting Recursion. *Journal of Symbolic Logic*, Volume 30 (1), pp. 28–48, 1965.
32. H., PUTNAM. Trial-and-Error Predicates and the Solution to a problem of Mostowski. *Journal of Symbolic Logic*, Volume 30 (1), pp. 49–57, 1965.
33. G., M., RADER, A method for composing simple traditional music by computer. *Communications of the ACM*, Volume 17 (11), pp. 631–638, 1974.
34. J., D., FERNANDEZ, F., VICO. AI methods in algorithmic composition: a comprehensive survey. *Journal of Artificial Intelligence Research*, Volume 48 (48), pp. 513–582, 2013.
35. K., THYWISSEN. Genotator: an environment for exploring the application of evolutionary techniques in computer-assisted composition. *Organised Sound*, Volume 4 (2), pp. 127–133, 1999.
36. D., COPE. Computer modeling of musical intelligence in EMI. *Computer Music Journal*, Volume 16, (16), pp. 69–87, 1992.
37. M., ALLAN. Harmonising chorales in the style of Johann Sebastian Bach. Master's Thesis, School of Informatics, University of Edinburgh, 2002.
38. D., SILVER, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, Volume 529 (7587), pp. 484–489, 2016.
39. A., HUANG, R., WU. Deep learning for music, 2016. Access on the internet: https://arxiv.org/abs/1606.04930
40. Machine learning. [seen on 2018-05-24] Access on the internet: http://whatis.techtarget.com/definition/machine-learning
41. Machine learning. [seen on 2018-05-24] Access on the internet: https://www.britannica.com/technology/machine-learning
42. Machine learning. [seen on 2018-05-24] Access on the internet: http://www.sas.com/en_us/insights/analytics/machine-learning.html
43. P., M., TODD. A connectionist approach to algorithmic composition. *Computer Music Journal*, pp. 27–43, 1989.
44. M., C., MOZER. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science*, Volume, 6 (2-3), pp. 247–280, 1994.
45. R., N., SHEPARD. Geometrical approximations to the structure of musical pitch. *Psychological Review*, Volume 89 (4), pp. 305, 1982.
46. J., A., FRANKLIN. Multi-phase learning for jazz improvisation and interaction. In *Proceedings of the Eighth Biennial Symposium for Arts & Technology*, 2001.

47. D., ECK, J., SCHMIDHUBER. A first look at music composition using lstm recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, 2002.
48. J., A., FRANKLIN. Recurrent neural networks for music computation. *INFORMS Journal on Computing*, Volume 18 (3), pp. 321–338, 2006.
49. P., MESSICK. Maximum midi, 1988.
50. N., BOULANGER-LEWANDOWSKI, Y., BENGIO, P., VINCEN. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription, 2012. [seen on 2018-05-24] Access on the internet: https://arxiv.org/abs/1206.6392
51. D., JOHNSON. Generating polyphonic music using tied parallel networks. *EvoMUSART: Computational Intelligence in Music, Sound, Art and Design*, pp. 128-143, 2017.
52. D., JOHNSON. Composing Music with Recurrent Neural Networks, 2015. [seen on 2018-05-24] Access on the internet: http://www.hexahedria.com/2015/08/03/composing-music-with-recurrent-neural-networks/
53. F., V., VEEN. The neural network zoo. [seen on 2018-05-24] Access on the internet: http://www.asimovinstitute.org/neural-network-zoo/
54. S., HOCHREITER, J., SCHMIDHUBER. Long short-term memory. *Neural computation*, Volume 9 (8), pp. 1735–1780, 1997.
55. P., J., WERBOS. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, Volume 78 (10), pp. 1550–1560, 1990.
56. A., J., ROBINSON, F., FALLSIDE. The utility driven dynamic error propagation network. University of Cambridge Department of Engineering, 1987.
57. I.-T., LIU, B., RAMAKRISHNAN. Bach in 2014: Music Composition with Recurrent Neural Network. [seen on 2018-05-24] Access on the internet: https://arxiv.org/abs/1412.3191
58. A., KARPATHY. The Unreasonable Effectiveness of Recurrent Neural Networks, 2015. [seen on 2018-05-24] Access on the internet: https://karpathy.github.io/2015/05/21/rnn-effectiveness/
59. P., SMOLENSKY. Information processing in dynamical systems: Foundations of harmony theory, 1986.
60. I., SUTSKEVER, G., E., HINTON, T., W., GRAHAM. The recurrent temporal restricted Boltzmann machine. *In Advances in Neural Information Processing Systems*, pp. 1601–1608, 2009.
61. Bachbot. [seen on 2018-05-24] Access on the internet: http://bachbot.com/
62. T., MIHAELA, H., JAEGER. Echo State Networks - Rhythm Generator for Music Compositions. Jacobs University Bremen, 2013.

63. C., J., PLACK, A., J., OXENHAM, R., R., FAY. *Pitch: Neural Coding and Perception*, New York: Springer, 2005. ISBN: 0-387-23472-1.
64. Mido – MIDI objects for Python. [seen on 2018-05-24] Access on the internet: https://mido.readthedocs.io/en/latest/
65. M., LUKOŠEVIČIUS. A Practical Guide to Applying Echo State Networks. *Neural Networks Tricks of the Trade, 2nd e.*, Springer, 2012
66. H., JAEGER. Echo state network, Scholarpedia. [seen on 2018-05-24] Access on the internet: http://www.scholarpedia.org/article/Echo_state_network
67. Sample echo state network source codes. [seen on 2018-03-11] Access on the internet: http://minds.jacobs-university.de/mantas/code
68. G., HINTON. Lecture 8.4. – Echo State Networks. [seen on 2018-05-24] Access on the internet: https://www.youtube.com/watch?v=vlRwUV_sGcs
69. D., ROBINSON. The Incredible Growth of Python, 2017. [seen on 2018-05-24] Access on the internet: https://stackoverflow.blog/2017/09/06/incredible-growth-python/