Towards a Style-driven Music Generator

Helder David Fonseca Duarte

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor: Doctor David Manuel Martins de Matos

Examination Committee

Chairperson: Doctor Teresa Maria Sá Ferreira Vazão Vasques
Supervisor: Doctor David Manuel Martins de Matos
Member of the Committee: Doctor Helena Sofia Andrade Nunes Pereira Pinto

January 2021

Acknowledgements

I would like to thank my parents for their encouragement, support and love, for always backing me up and for helping me along this journey.

To my brother João, for his amazing help, feedback and support while doing this work. I can't thank him enough for being my number one test subject throughout this thesis.

To my second family, Tuna Mista do Instituto Superior Técnico (TMIST), a "home" with people I can always look up to, for help and inspiration. Thanks for the growth you have sprouted in me. To more never-ending "Noites de Folia"!

To all my friends, specially Nuno Miguel Macara and Pedro Rocha for their help and support in reviewing and supporting me throughout this work. To all of them for the inspiration and constant words of encouragement.

To my supervisor David Martins de Matos, for his incredible help and his words of wisdom throughout this thesis, for all the trust and encouragement, who also helped me to grow a lot academically.

Lisboa, January 26, 2021
Helder David Fonseca Duarte

Resumo

A composição musical sempre esteve sujeita a diferentes abordagens inovadoras ao longo da história. Uma abordagem possível seria tentar recriar um dado estilo à nossa escolha de um compositor que já tenha falecido ou de uma banda que já não existe. Queremos com este trabalho criar um agente que consiga gerar músicas ou peças que se assemelhem às de um dado estilo.

Com a ajuda de alguns dos modelos do estado da arte, um modelo foi selecionado e otimizado, com o seu output gerado inquirido a uma população para perceber a sua performance.

Apesar de este processo de seleção e otimização ter demonstrado que o modelo obtido funciona para alguns estilos, são necessários mais desenvolvimentos e pesquisa.

Esperamos com este trabalho estabelecer uma base para ajudar futuros trabalhos neste assunto, quer para criar aplicações relacionadas ou para realizar mais otimizações no modelo proposto.

Abstract

Music composition has always been subject to several innovative approaches throughout history. A possible approach would be to try to resemble a given style that we like from a deceased composer or an old band. We aim with this work to create an agent that is capable of creating songs or pieces that would resemble those specific styles.

With the aid of some state of the art models, a model was selected and optimized, with its generated output being surveyed to a set of respondents to perceive how well it performed.

Although this process of selection and optimization has shown that the obtained model works for certain styles, further research and development needs to be done.

We hope with this work to establish a baseline to help future works on this matter, either for creating related applications or further improving the proposed model.

Palavras Chave
Keywords

Palavras Chave

Composição Musical

Estilo Musical

Agentes Inteligentes

Aprendizagem Automática

MIDI

Keywords

Music Composition

Musical Style

Intelligent Agents

Deep Learning

MIDI

Index

1 Introduction **1**

1.1 Motivation . . . 1

1.2 Objectives . . . 2

1.3 Document structure . . . 3

2 Background and Related Work **5**

2.1 Musical considerations . . . 5

2.1.1 Style . . . 5

2.1.2 Some notions of music theory . . . 6

2.1.3 Considerations on Western Music . . . 7

2.1.4 MIDI Data and Pianorolls . . . 8

2.1.5 Dataset choice . . . 10

2.2 General concepts: Deep Learning . . . 10

2.3 State of the Art . . . 13

2.3.1 AIVA . . . 13

2.3.2 OpenAI's Jukebox . . . 14

2.3.3 Google Brain's Magenta . . . 16

2.3.3.1 Polyphony RNN . . . 16

2.3.4 DeepBach . . . 17

2.3.5 Composer . . . 18

3 Methodology **21**

3.1 Setup and approach . . . 21

3.1.1 Dataset collection . . . 21

3.1.2 Model training . . . 22

3.1.3 Model optimizations to the interface . . . 23

3.1.4 Output generation . . . 23

3.1.5 Experimental data collection 24
3.2 Initial model testing and experiments 25

4 Results and Discussion 27
4.1 Expected results and experimental observation 27
4.1.1 Expected results 27
4.1.2 Experimental observations from the author 28
4.2 Results obtained from form answers 29
4.3 Critique and discussion 31
4.3.1 Discussion of Form Results 31
4.3.2 Critique to the model and posed problems 32

5 Conclusions and Future Work 35
5.1 Conclusions 35
5.2 Future Work 35

I Appendices 41

A Composer Interface commands 43

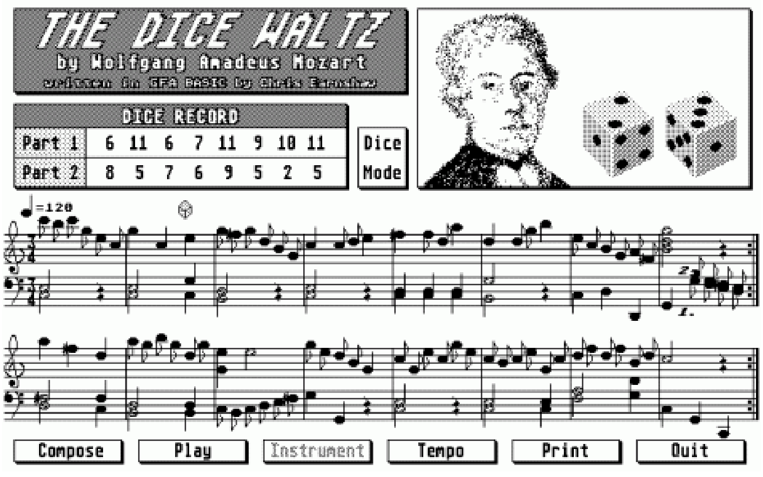
B Google Form 45

1 Introduction

1.1 Motivation

From immemorial times, humans have been creating art to express their different fields of mind in the most various ways; and being one of these ways music itself, one could argue that one of the most complex tasks associated with it would exactly be composing music.

Some composers, throughout history, have found numerous ways of composing and even created some sort of algorithms, like using cards with fragments of music written on and rearranging them allowing thousands of different pieces, as in John Clinton's Combinatorial Music Machine, used to generate Quadrilles (Braguinski, Nikita, 2019) or using dice, as with Mozart's dice game, shown at figure 1.1.


THE DICE WALTZ
by Wolfgang Amadeus Mozart
written in GFA BASIC by Chris Earnshaw
Dice Mode
♩=120
Compose
Play
Instrument
Tempo
Print
Quit

DICE RECORD								
Part 1	6	11	6	7	11	9	10	11
Part 2	8	5	7	6	9	5	2	5

Figure 1.1: Mozart's dice game (Dice Game, 2012)

Since music can be composed, although not necessarily, with the aid of precise mathematical tools, based on sets of durations and frequencies (pitches), giving birth to motifs with chord progressions, melodies, and harmonic structures using as more varied rhythms as possible, we obtain a primary basis for writing music according to what is expected in a western culture style frame.

Another point of view that we need to account for, is that in order to recreate the works of a composer that has died or is hard to get access to, we need some way of being able to mimic their style. Therefore, and given the possibility, we should aim for a solution that stylistically resembles their work.

1.2 Objectives

In order to attain some sort of resemblance with the works we are going to get inspiration on, Artificial Intelligence and Machine Learning may help us reaching our goal of trying to recreate, up to some extent, the human capability of music composition, by replacing the human agent with a computational entity, more specifically, a trained neural network (Graves, Alex, 2014).

There are publicly available trained models and products, some using MIDI and others using raw audio files, but since MIDI is much easier to classify and structure, more specifically in terms of computation, its usage is encouraged for the sake of creating simpler models for training, as raw audio input also requires considerations regarding signal processing.

Figure 1.2: A MIDI keyboard, an important element of modern music composition (ClewsOctober, 2020)

We can then set up a set of experiments to be able to test our approach and try to achieve satisfying results. The results may be tested by human listeners, preferably with different musical backgrounds. The main focus of this work is to recreate and mimic the original style by feeding a given model a set of musical references in MIDI format. This task can be summed up as:

- Capturing the style of the reference datasets with the help of a set of models.
- Picking the best model capable of consistently producing an output that can best suit the given references.
- Checking if the resulting output (or alternatively a derived hand-picked set) can be recognized by the general public as being of those given styles.

We hope with this work to formulate a deep learning model that can comprehend the style of the dataset that has been trained on to later replicate that style as a means of recreating its respective key features. This model shall be robust as in it can always generate music of a given style, regardless of what that style may be.

It is to be noted that motifs and other musical structures can appear in the output up to some variation, since artists usually reuse some of their own motifs among different themes, as a part of ”sticking to their own style”.

Figure 1.3: Kiss' unique style is also part of their visual (Miranda, 2020)

With this work, optimizing a pre-existing model while reuniting ideas from other models, may lead us to a better outcome than the respective predecessors that inspired this work. Later work may include an application, either for musicians or non-musicians to produce and tinker music with, and some other optimizations that were limited by the time constraint.

1.3 Document structure

This work can be structured into its theoretical Background and Related Work, where we explain both the concepts that we will be exploring and the models that arise from the current state of the art; the Methodology used, where we can prepare what to expect from our results and all the scientific considerations involved; the Results and Discussion, where we evaluate our output and how it fits in the methodology defined; and the Conclusions and Future Work, where we analyze how successful were the results of this work, and where incomplete or unaddressed topics will be given the spotlight.

2 Background and Related Work

2.1 Musical considerations

2.1.1 Style

As defined in (Shlomo Argamon & (Eds.), 2010), musical style can be associated to several different aspects of music, like:

- Historical periods (e.g. Classical, Baroque, etc.);
- Particular composers (e.g. Beethoven, Mozart, Vivaldi, etc.);
- Performers (e.g. Itzhak Perlman (figure 2.1), Miles Davis, Jimi Hendrix, etc.);
- As a synonym for texture, as in arrangements (instrumental choice), melodic patterns, rhythmic patterns, harmonic patterns and chord progressions, and cadences. On a more general aspect: orchestral, big band, marching band and other sorts of musical groups;
- Emotional associations (sad, exciting, soothing, calm, scary, etc.);
- As a synonym for genre, from the association to a given artist to a set of cultural and social influences, such as traditional music, for example (such as Pop, Rock, Jazz, etc.).

Figure 2.1: Itzhak Perlman performing (Dobrin, 2019)

Since we want to isolate specific styles to check how well the model behaves, it may be a good approach to consider particular composers, bands or other sorts of distinguishable music composers/groups as our definers for style.

This choice excludes the emotional and performance connotations, since we are trying to capture style as both the genre and the sub-genre given by the intrinsic characteristics to each composer (as an example, if we consider as our genre erudite Hungarian violin compositions, we can consider Béla Bartók as our sub-genre).

This separation is optimal for creating a dataset of our own, with nuances like these. This may also allow us to cross-influence styles if we join different styles on our datasets.

Other notions that were taken into account are both arrangement and instrumentation. Arrangement, as in how the different melodies that compose the harmony are distributed by the different voices/instruments; and instrumentation as which instrument shall perform a given melody.

To deal with these situations, we decided to focus on tracks that can be reduced to single instruments, as in if the track would work under the conditions of a piano arrangement. Although not always being possible, most of the elements of our chosen datasets (section 2.1.5) can fit into these conditions.

2.1.2 Some notions of music theory

Since our musical goals may seem vague, in hopes of clarifying what we are trying to achieve, we may first define the notions that rule our work, like rhythm, chord progressions and melodic lines. Each of these aspects has its own nuances and shall be treated with its own importance.

First and foremost, notes are associations made to certain pitches, in other words, frequencies of sound, that can have various timbres associated to the instrument that plays it. Timbre is what allows us to distinguish two instruments, like a piano and a guitar, that playing the same base pitch, their differences in harmonics (in musical language, overtones), allow us to identify them uniquely.

Notes usually have associated durations, i.e., different rhythmic values that correspond to different figures in notation, each with a power of 2 associated, as shown in figure 2.2.

Rhythm also provides us a base to comprehend a time duration, assigning a given figure to a certain amount of beats per minute, meaning that the occurrences defined by a certain time signature, such as $\frac{4}{4}$ can account for a duration of four beats, corresponding to quarter notes at a given tempo, as shown in 2.3. Beats per minute can also be considered as the periodicity of beats, as 60 beats per minute correspond to a 1 Hz beat frequency.

We can now look at the more melodic aspects, where the mathematical relationships among different pitches are more interesting than the pitches themselves, creating a basic sensation of tension and release. Although this may arise directly from a single melodic line, with a single note at a time, it becomes easier to convey musical ideas through intervals and chords.

Thus, an interval may be seen as merely a relationship established by two notes (or pitches) that can also be understood as a frequency ratio. Different ratios have different perceptions: simpler ratios, like 3:2 or 2:1 sound more harmonious to the human hearing than 16:15 as the prior ratios are more consonant.

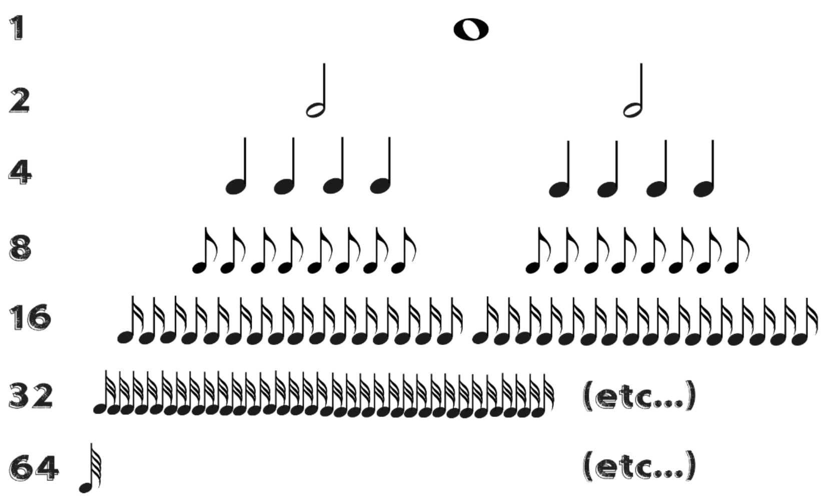


Figure 2.2: Rhythmic figure values (Image courtesy of TMIST)

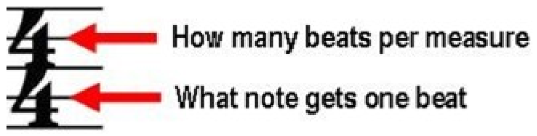


Figure 2.3: Time signature explanation, edited from (*Time Signatures 4/4 - KNILT*, 2018)

Consonance is mathematically related to the ratio as, for instance, two notes related by a 3:2 ratio, played by any instrument, share more overtones than a 16:15 ratio, meaning that the sounds will blend together with each other more harmoniously.

Intervals, however, in our current tuning system are not perfect ratios but factors of $\sqrt[12]{2}$, since by evenly dividing the most important ratio of 2:1, the octave, into its twelve tones used in western music, the constant ratio that we get, close to the value of the smallest used interval, is the one of $\sqrt[12]{2}$. This builds the tuning system called Twelve Tone Equal Temperament (or 12-TET).

Different sets of notes, arranged in the most various intervals can create all sorts of chords, if played simultaneously. However, if notes are arranged in sequences of ascending or descending notes, we obtain scales, one of the primary basis for melody. Different chord progressions provide different intentions and expressions and generate unique structures, playing a crucial role in defining style.

Relating these concepts to our work, the more diverse our set of melodies, chords and harmonic intentions and expressions, the more stress-tested our models can be and the better we can confirm their robustness for reaching our desired results.

2.1.3 Considerations on Western Music

Now that we have a better understanding of what concepts are at stake, we can now look at how the stylistic choice results in simplifications.

Western music has obeyed, throughout history, to practically the same set of rules, as previously defined, fixed in the same scales, the same chords and the same underlying rhythmic and harmonic structures. For instance, we can look at Jazz as being the peak of using overly complex rhythms and harmonies, with the style specificities relying strongly on the performer as improvisation is a known key feature of Jazz.

Figure 2.4: Miles Davis performing Jazz (Futterman, 2016)

On the other end, we can look at Pop, with a pre-defined structure of pre-choruses, choruses and bridges, being very directional, with an associated rhythmic and harmonic structure directly associated to each of the sections. Since human beings are creatures of habit, repetitive structures have a tendency for becoming more catchy because we get used to them faster.

We can also identify, as an example for Pop, the structure as a letter code. This way, the structure of: [Intro, Verse 1, Pre-chorus 1, Chorus 1, Verse 2, Pre-chorus 2, Chorus 2, Pre-chorus 3, Chorus 3, Outro] may become [A B C A B C A C A A], for example. The idea here is to simplify the structure into more generic sections of a song/piece.

Anyhow, what both of these styles share is that every single note can be defined in a Twelve Tone Equal Temperament system, in a specific rhythmic figure, in a specific amount of beats per minute. These underlying qualities allow us to, up to some extent and at the stake of losing the nuances of a human performer, to generate pianorolls to be played as MIDI data.

2.1.4 MIDI Data and Pianorolls

According to (MIDI, 2017), MIDI ”Stands for ”Musical Instrument Digital Interface.” MIDI is a connectivity standard for transferring digital instrument data. It is primarily used by computers, synthesizers, and electronic keyboards.

MIDI data includes several types of information. For example, pressing a single key on a synthesizer transmits the note played, the velocity (how hard the note is pressed), and how long the note is held. If multiple notes are played at once, the MIDI data is transmitted for all the notes simultaneously.” This means that with MIDI we can perfectly encode Western Music, as this standard was first developed by musicians in a Western cultural context.

A sample, being a pre-recorded sound of a desired instrument, we can play back MIDI ”as audio samples, such as a piano or strings.” We can also edit MIDI being allowed ”to adjust the timing and velocity of individual notes, change their pitch, delete notes, or add new ones. MIDI data is often displayed in a digital format, with lines representing each note played. Many programs can convert MIDI data to a musical score as well.”

”A MIDI recording simply contains instrument information and the notes played. The actual sound is played back using samples (individual recordings) from real instruments. For example, a MIDI track recorded as a song for piano can be played back with a guitar sound by simply changing the output instrument — though it might not sound very realistic.”

A pianoroll is simply a sheet of paper with punctured holes in it, being the position in the x-axis of these punctures corresponding to which note is played (the pitch axis) and the height is the duration of how long these notes are played (the time axis).

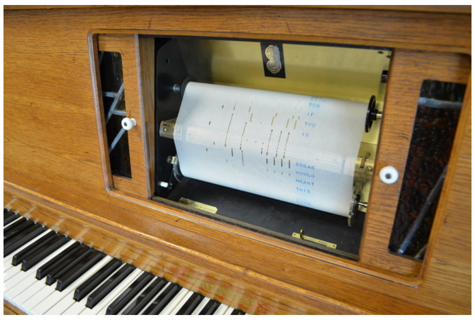
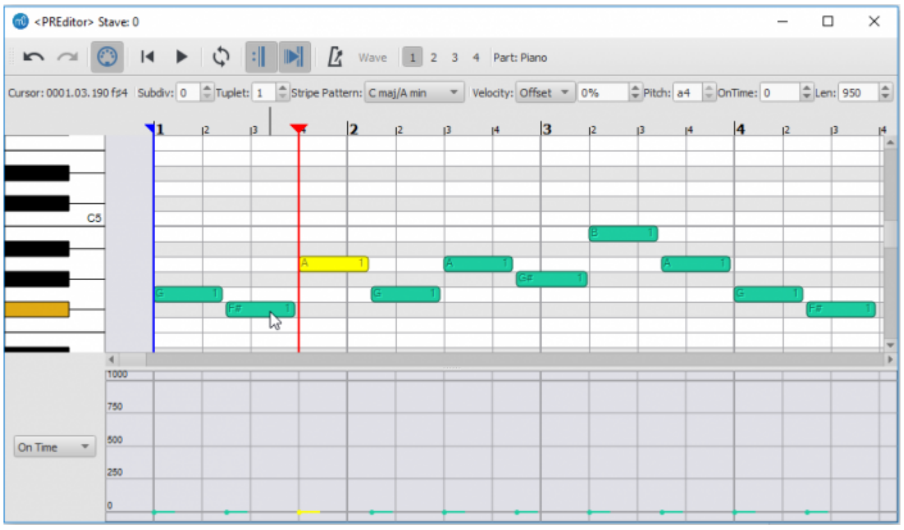


Figure 2.5: An example of a paper pianoroll (Physical Pianoroll, 2020)

In more modern times, the pianoroll became a part of the MIDI editing interfaces, and thus, was rotated by 90 degrees making our time axis horizontal and our pitch axis vertical, making it a parallel to a regular music score. Pianorolls are simple and effective ways to edit MIDI by simplifying the editing into basic features of pitch and duration.



Cursor: 0001.03.190 f♯4 Subdiv: 0 Tuplet: 1 Stripe Pattern: C maj/A min Velocity: Offset 0% Pitch: a4 OnTime: 0 Len: 950

Figure 2.6: An example of a digital pianoroll (Digital Pianoroll, 2020)

2.1.5 Dataset choice

To make our stylistic choices, we need to find datasets that both vary in size and in specificity, for instance, Pop is much more generic than Mozart which in turn may be a smaller dataset. This variation is needed in order to guarantee that the model is robust and that the optimizations made to it are independent of the dataset type or the amount of data that the model is fed.

Another aspect to be considered are the similarities that may occur between different styles, either with their structural aspects, such as chord progressions, or among melodic lines and arrangements, as in a more melodic style such as Pop can be more similar to Power Metal than Videogame music, as both Pop and Power Metal have a defined musical structure, with very strong directional structure, Videogame soundtracks are made to be more repetitive backing tracks and, in the opposite end, styles such as of Mozart or Queen are more free-form.

As a part of the music writing process, our model will need references to learn from and write music alike. Thus, the creation of stylistically consistent datasets is important to achieve a reliable agent. A dataset with NES (Nintendo Entertainment System) videogames, a dataset with Power Metal music, specifically Dragonforce, a dataset with Pop songs, a dataset with Rock music, specifically Queen and a dataset with piano compositions from Mozart, establish our baseline of datasets to train on.

A first dataset with Bach chorales was chosen as the baseline to compare our models, since it is relatively simple, being widely used in similar works, and with it we can obtain a more clear view of the path we may take and how we can create (or as seen later, optimize) a model to suit our needs of generating music in these styles.

2.2 General concepts: Deep Learning

Relating to Deep Learning, we need to define some concepts in order to better understand the models that we will be analyzing next. First off, we can look at an algorithm as being ”An unambiguous specification of a process describing how to solve a class of problems that can perform calculations, process data and automate reasoning.” (Appen, 2020)

Deep Learning is a subfield of Machine Learning, which in turn is ”The subfield of Artificial Intelligence that often uses statistical techniques to give computers the ability to “learn”, i.e., progressively improve performance on a specific task, with data, without being explicitly programmed.” (Appen, 2020)

With the basic concept of Machine Learning, the primary distinction of Deep Learning is its usage of Neural Networks. ”An artificial neural network learning algorithm, neural network, or just neural net, is a computational learning system that uses a network of functions to understand and translate a data input of one form into a desired output, usually in another form. The concept of the artificial neural network was inspired by human biology and the way neurons of the human brain function together to understand inputs from human senses.

Neural networks are just one of many tools and approaches used in machine learning algorithms. The neural network itself may be used as a piece in many different machine learning algorithms to process complex data inputs into a space that computers can understand.” (DeepAI, 2019b) These neural networks are composed by neurons (sometimes defined as nodes). These are defined as ”A unit in an Artificial Neural Network processing multiple input values to generate a single output value.” (Appen, 2020)

In this context, we can also say that an epoch is, ”one pass of the full training data set.” We can also say that a layer is ”A series of neurons in an Artificial Neural Network that process a set of input features, or, by extension, the output of those neurons. Hidden Layer: a layer of neurons whose outputs are connected to the inputs of other neurons, therefore not directly visible as a network output.” (Appen, 2020) A feature is ”A variable that is used as an input to a model.”

Overfitting is ”The fact that a model unknowingly identified patterns in the noise and assumed those represented the underlying structure; the production of a model that corresponds too closely to a particular set of data, and therefore fails to generalize well to unseen observations.”

Underfitting is ”The fact that a Machine Learning algorithm fails to capture the underlying structure of the data properly, typically because the model is either not sophisticated enough, or not appropriate for the task at hand; opposite of Overfitting.”

Tensorflow is ”An open-source library, popular among the Machine Learning community, for data flow programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks.”

The learning rate is ”A scalar value used by the gradient descent algorithm at each iteration of the training phase of an Artificial Neural Network to multiply with the gradient.” (Appen, 2020)

Training is the process of adjusting a Neural Network, through exposure to data, to the desired parameters, in our case, to approximate the style of the generated output to the one of the training data. It is tightly influenced by the values of the hyperparameters. ”In the context of Supervised Machine Learning, the construction of algorithms that can learn from and make predictions from data. Training Data: The subset of available data that a data scientist selected for the training phase of the development of a model.”

A batch is ”The set of examples used in one gradient update of model training.” (Appen, 2020) The batch size is the size of the subdivisions that are made on the training set, as the whole set of data is concatenated into a unique file. Thus, the data, in order to be manageable in memory while training, needs to be separated into batches.

Classification is ”The task of approximating a mapping function from input variables to discrete output variables, or, by extension, a class of Machine Learning algorithms that determine the classes to which specific instances belong.” (Appen, 2020)

”A recurrent neural network (RNN) is a feedforward neural network extended with *recurrent connexions* in order to learn series of items (e.g., a melody as a sequence of notes). The input of the RNN is an element of the sequence(...). In other words the RNN will be trained to predict the next element of a sequence. In order to do so, the output of the hidden layer reenters itself as an additional input (with a specific corresponding weight matrix). This way, the RNN can learn, not only based on the current item but also on its previous own state, and thus, recursively, on the whole of the previous sequence. Therefore, an RNN can learn sequences, notably temporal sequences, as in the case of musical content.” (Briot & Pachet, 2019), (Graves, Alex, 2014).

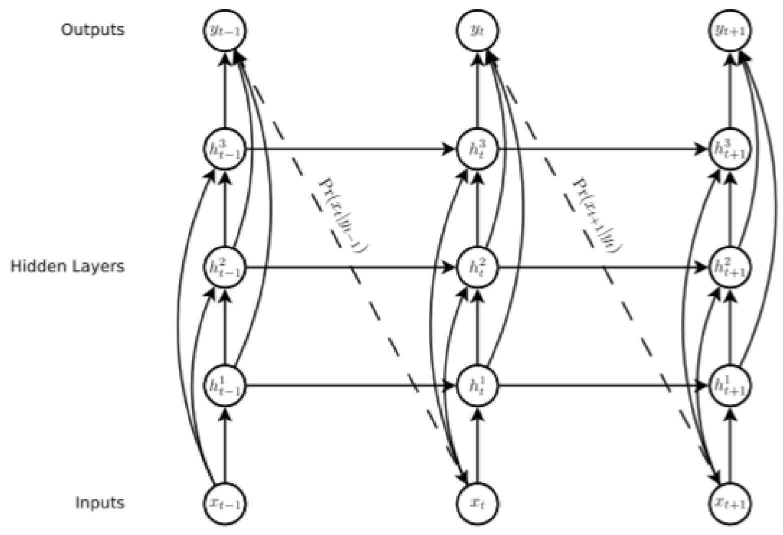


Figure 2.7: Recurrent Neural Network architecture

Reinforcement Learning is a paradigm of machine learning that aims for maximizing reward over a performed set of actions by an agent (Briot & Pachet, 2019). This may be later useful for optimizing the training of our Neural Networks.

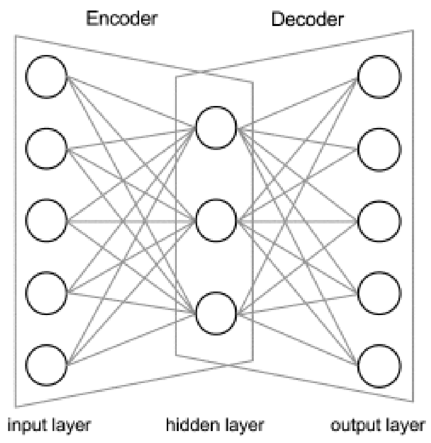


Figure 2.8: Autoencoder architecture

LSTM (Long Short-Term Memory) nodes are special kinds of neural network nodes, that for the sake of simplicity will be treated as black boxes since we are only interested in their behaviour of retaining data if told to.

Embeddings are lower dimension representations of higher dimension vectors, meaning that several data elements can be grouped as a single element of the embedding vector. This is useful, for example, on grouping training parameters to be encoded as a single one.

”The term embedding comes from the analogy with mathematical embedding, which is an injective and structure-preserving mapping. Initially used for natural language processing, it is now often used in deep learning as a general term for encoding a given representation into a vector representation. Note that the term embedding, which is an abstract model representation, is often also used (we think, abusively) to define a specific instance of an embedding”. (Briot & Pachet, 2019)

Training Data: The subset of available data that a data scientist selected for the training phase of the development of a model.

”A hyperparameter is a parameter that is set before the learning process begins. These parameters are tunable and can directly affect how well a model trains.” Hyperparameters are manually adjustable parameters that determine the functional behavior and structure of the neural network, such as learning rate, number of epochs, batch momentum or other regularization constants. (DeepAI, 2019a)

Changing these values heavily influences how the different metrics evolve during training and the final result. Are considered hyperparameters, for example, the learning rate, the normalization momentum, the number of layers, the number of nodes, the number of epochs and the batch size.

Loss is the set of errors that happen between training and test/validation sets and how well the neural network is behaving; a lower loss implies a better behavior, i.e., a better fit between results and original data.

2.3 State of the Art

In order to generate music automatically, we need to check on what technology we currently have, not only to make our job easier, but also to check if our goal has already been accomplished in some way or another. For the sake of simplicity, drum models will not be considered although they do serve a purpose in style, it is still possible to distinguish different styles with the drum tracks being absent. Only Jukebox, AIVA and the Magenta models for drums, use drum tracks as part of the composition process.

2.3.1 AIVA

First off, we can take a look at AIVA, (AIVA, 2020) an online interface that allows the user to select a style, from a set of predefined styles, or to upload a song to serve as an influence.

It allows the user to ask AIVA to generate a given amount of songs and, after a few seconds, it produces an output. If the user chooses a preset for style, it usually behaves well in that matter, following more or less some key features of that style, but if an influence song is used instead, the results do not sound as promising. To test this myself, I’ve used the theme for Super Mario single-handedly since the tool only allows a single song to be used as an influence. I further listened to the results and tried to check if a theme as memorable and catchy would be produced, but to no avail.

AIVA is capable of producing songs with a strong sense of structure, i.e. a beginning, a climax and an ending, in other words, it appears to be capable of capturing cadence. Although these key features are important in many musical styles, capturing chord structure and stylistic nuances such as voicings and instrument choices, are as important as the cadence itself, rendering this model as a quite limited choice for achieving our goal.

Another limitation is associated to the output of the model, that can be further changed in a pianoroll but it only allows for moving the timestamps of where each part/instrument comes in, making it very limited from the standpoint of MIDI editing.

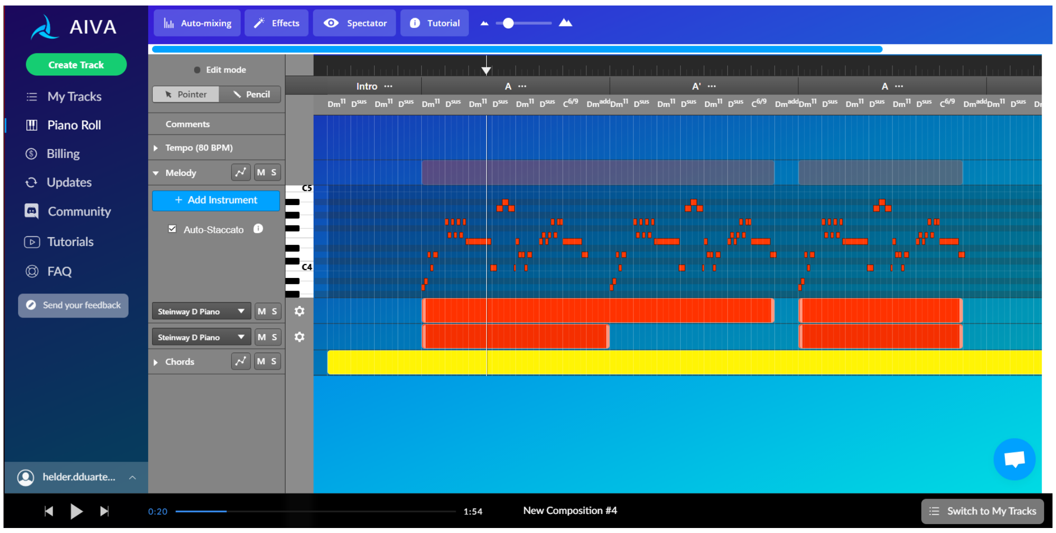


Figure 2.9: AIVA's pianoroll interface

Either way, this system relies on a pre-trained network and is a commercial product. Although at first instance it might look like it can achieve our goal, the style specificity we are looking for cannot be achieved. As an example, we can only select rock, but we cannot select the style of a specific band/decade, making it unsuitable for recreating stylistic details.

2.3.2 OpenAI's Jukebox

Next up, we have Jukebox (Dhariwal et al., 2020), part of the OpenAI project, in the words of its creators: "We're introducing Jukebox, a neural net that generates music, including rudimentary singing, as raw audio in a variety of genres and artist styles. We're releasing the model weights and code, along with a tool to explore the generated samples.", which means that we can not only explore their original code to help us achieving our goal, but also try to improve it to better suit our needs.

Jukebox directly deals with sound waves, which helps it with the task of combining styles. This approach, although being more memory expensive, is better for perceiving nuances, and the online version displays written songs with lyrics, something that may be important for capturing some stylistic details.

Functionally, the model first uses a VQ-VAE (Vector Quantized Variational Auto Encoder), an approach of down-sampling long context inputs into smaller vectors, in order to compress raw audio to a lower-dimensional space. After this, the embeddings are subject to random restarts, and different auto-encoders are used in order to comprehend different levels of compression. The model uses spectral loss, which helps it better capture the mid to high-range frequencies. It is also conditioned in terms of artist, genre, timing, and lyrics, with the lyrics being time aligned to associate the lyrics with the singing.

"But symbolic generators have limitations — they cannot capture human voices or many

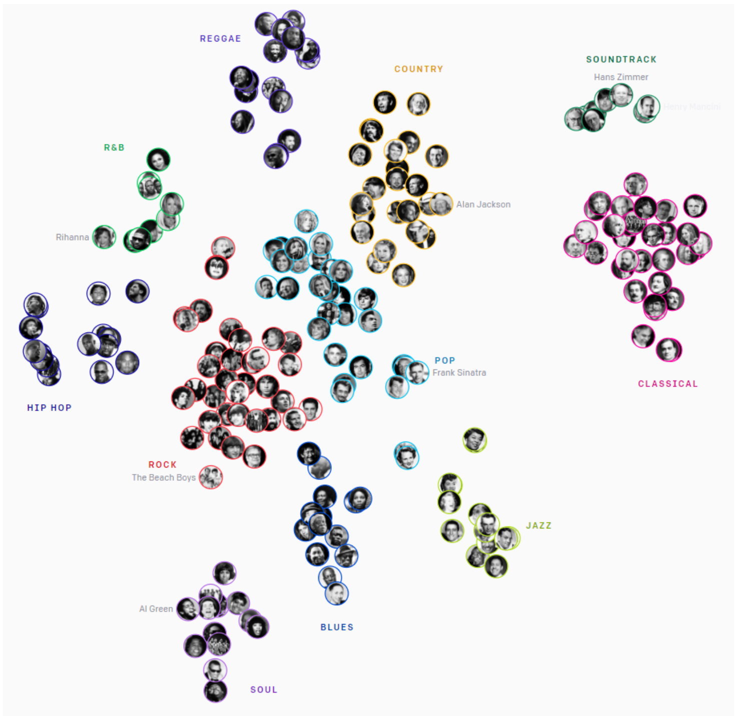


Figure 2.10: Jukebox's map of styles and relationships between each other

of the more subtle timbres, dynamics, and expressivity that are essential to music. A different approach is to model music directly as raw audio. Generating music at the audio level is challenging since the sequences are very long. A typical 4-minute song at CD quality (44 kHz, 16-bit) has over 10 million timesteps."

"Thus, to learn the high level semantics of music, a model would have to deal with extremely long-range dependencies. One way of addressing the long input problem is to use an autoencoder that compresses raw audio to a lower-dimensional space by discarding some of the perceptually irrelevant bits of information. We can then train a model to generate audio in this compressed space, and upsample back to the raw audio space."

I decided to take a look at the examples of Jukebox and listen closely to them and even though it is a promising set of songs, it still has a long way to come along our desired results (an in-style song, preferably catchy and memorable). Overall, the model has incredible ideas in terms of design and structure that might help optimizing our final model.

Although this tool is great to capture style from single songs, it does it in a way that is not the fastest nor the most granular, since we cannot separate the tracks into individual instruments (whereas we can in MIDI) and one of our primary goals is to make the training fast and flexible, allowing for swapping each of the instruments that make up the track, and therefore this will not be our first option. We also cannot rely on the output lyrics and the sound itself has way too many artifacts.

"For example, while the generated songs show local musical coherence, follow traditional chord patterns, and can even feature impressive solos, we do not hear familiar larger musical structures such as choruses that repeat. Our downsampling and upsampling process introduces discernible noise. Improving the VQ-VAE so its codes capture more musical information would help reduce this.

Our models are also slow to sample from, because of the autoregressive nature of sampling. It takes approximately 9 hours to fully render one minute of audio through our models, and thus they cannot yet be used in interactive applications. Using techniques that distill the model into a parallel sampler can significantly speed up the sampling speed. Finally, we currently train on English lyrics and mostly Western music, but in the future we hope to include songs from other languages and parts of the world."

2.3.3 Google Brain's Magenta

We can now consider Magenta (Git_Magenta, 2020), from the Google Brain team, publicly available on GitHub. Magenta comprises a set of models that aim to enable music composition and music generation, from drums to melody to harmony.

There is also a derived set of plugins for the average musician/producer to use but their results are quite underwhelming (Roberts et al., n.d.). For instance, the models are capable of generating music, given an influence, but they lack on producing the promised results, like on the continuation model, that should continue the melody, it often has a lot of pauses and deviates from the "expected" natural evolution of the track. I've tried all of the three different models but none of them captured the style of the track that I've tried (Super Mario's theme) properly.

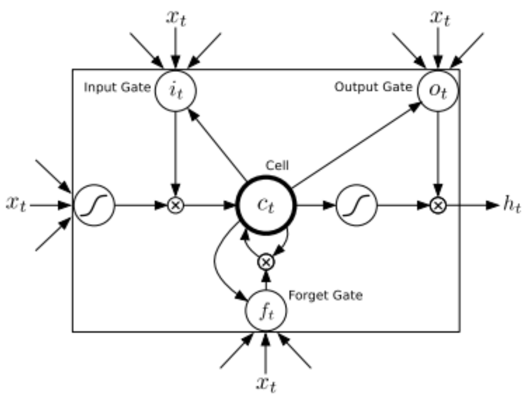


Figure 2.11: LSTM node architecture

2.3.3.1 Polyphony RNN

Searching a bit deeper in our available models, a particularly interesting one is Polyphony RNN, that by training on MIDI data can generate polyphonic MIDI files. This model is a sequence to sequence model, meaning that it directly tries to predict the next step (note) given the previous sequence, using an RNN optimized with LSTM nodes, shown in figure 2.11 instead of regular nodes, capable of being trained. (Graves, Alex, 2014)

From testing with our video-game dataset (a compilation of NES 64 themes) and our Bach dataset, the results of training either get stuck into repetitive loops, probably from overfitting, or just do not sound good with lots of dissonances or unrelated motifs, which is a major problem and would require further investigation on the optimal values for hyperparameters, such as keeping track of the optimal loss or the best value for accuracy.

These problems may arise due to the intrinsic more formal structure of the songs of the training set, that Polyphony RNN fails to follow, since its design may be better-suited for free-form music, which may be useful for other objectives, such as melodic creativity, but not necessarily generating music in style, therefore rendering it inadequate to our goal.

2.3.4 DeepBach

Another model that uses MIDI is DeepBach (Hadjeres, Pachet, & Nielsen, 2017), which consists of an architecture using two separate Deep Neural Networks and computes results with the help of a third Neural Network, like represented in figure 2.12.

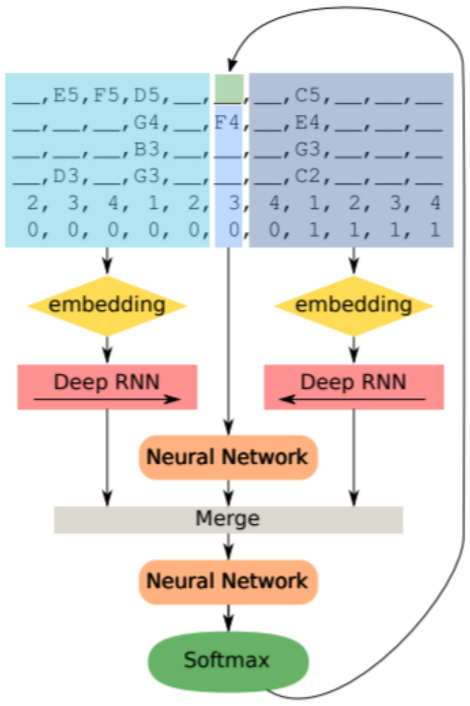


Figure 2.12: DeepBach's architecture

Although similar to Polyphony RNN, DeepBach instead "looks ahead", considering the next steps altogether with the previous steps, predicting the current one. DeepBach also uses MIDI processing in both training and output generation. This model includes an altered form of Gibbs sampling in order to harmonize the soprano pre-generated voice.

This model posed a problem on being usable, since I was not able of getting the code to work and the model's code being unmaintained does not help it either. Anyway, it looks like a very promising approach and, from the published results, was capable of following the style of Bach and even "fooled" professional musicians, as shown in figure 2.13. We may later use these methods, or similar, to help us recreate the desired style to be followed.

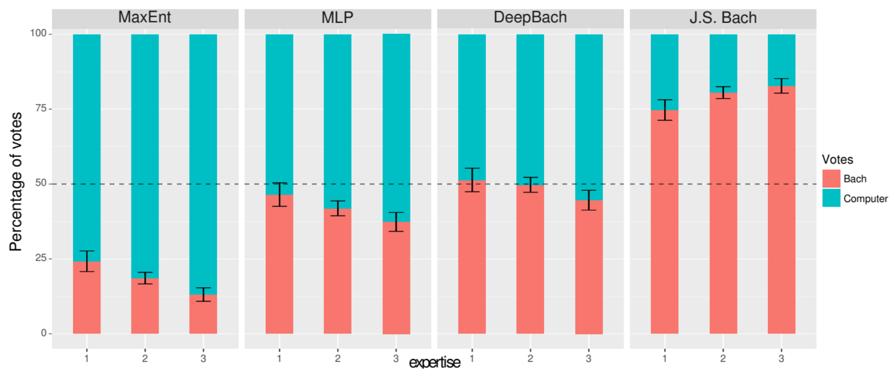


Figure 2.13: Results of the "Bach or Computer" experiment. The figure shows the distribution of the votes between "Computer" (blue bars, with 3 different models) and "Bach" (red bars) for each model and each level of expertise of the voters (from 1 to 3).

2.3.5 Composer

Last but not least, we have a model made by the YouTube content creator Code Parade, initially created to generate videogame music.

This model consists of a convoluted auto-encoder, (HackerPoet, 2018) that first converts the MIDI file, as a pianoroll, into each of its measures to a frame of 96 pitches per 96 steps, and each of the coordinates (step, pitch) corresponds to a different pixel on an image.

There were chosen 96 steps per measure as it evenly divides all the most common time signatures exactly as figure 2.14 shows. A third dimension is added to account for each and every measure of both the generated and training songs.

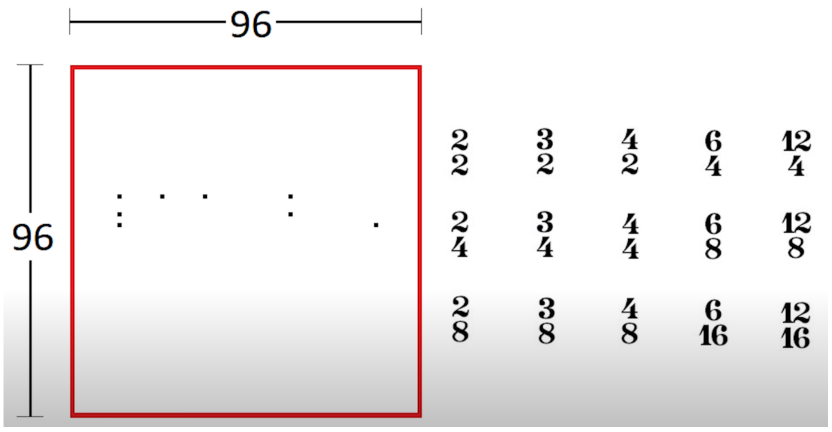


Figure 2.14: Correlation between the amount of steps and the most common time signatures

A dense network encodes each measure into a feature vector, which are in turn fed to a dense auto-encoder, outputting another feature vector, that gets back-converted to measures, like shown in figure 2.15. This network also allows for the use of embeddings.

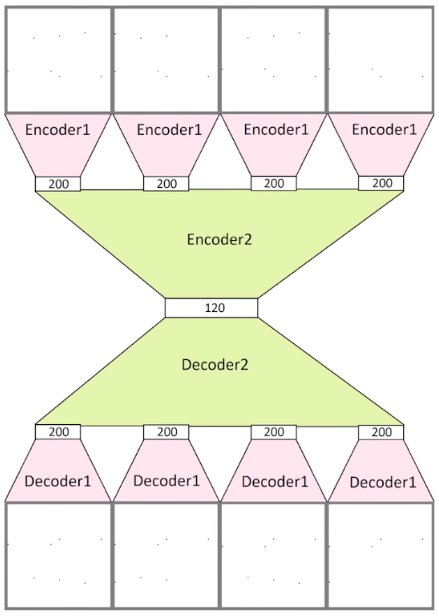


Figure 2.15: Network diagram

The trained encoder is then used in a PyGame interface with sliders corresponding to the top 40 parameters of the encoder, as shown in figure 2.16. These sliders control the parameters of the latent vector, changing the generated output. However, since they are not classified, makes it preferable to utilize random values than tweaking around to see what fits better. This interface allows for real-time generation of measures that can be played as pianoroll with some MIDI instrument, with the interface offering four simple waveforms. On top of this, we can save the generated song as a MIDI or WAV file for later playback.

We have now collected a set of models that we can use as a baseline for our model, providing ideas to optimize the best fitting model, having this choice better analyzed at section 3.2. This also enables us to efficiently create a model based in models known to have worked and fulfilled their purpose.

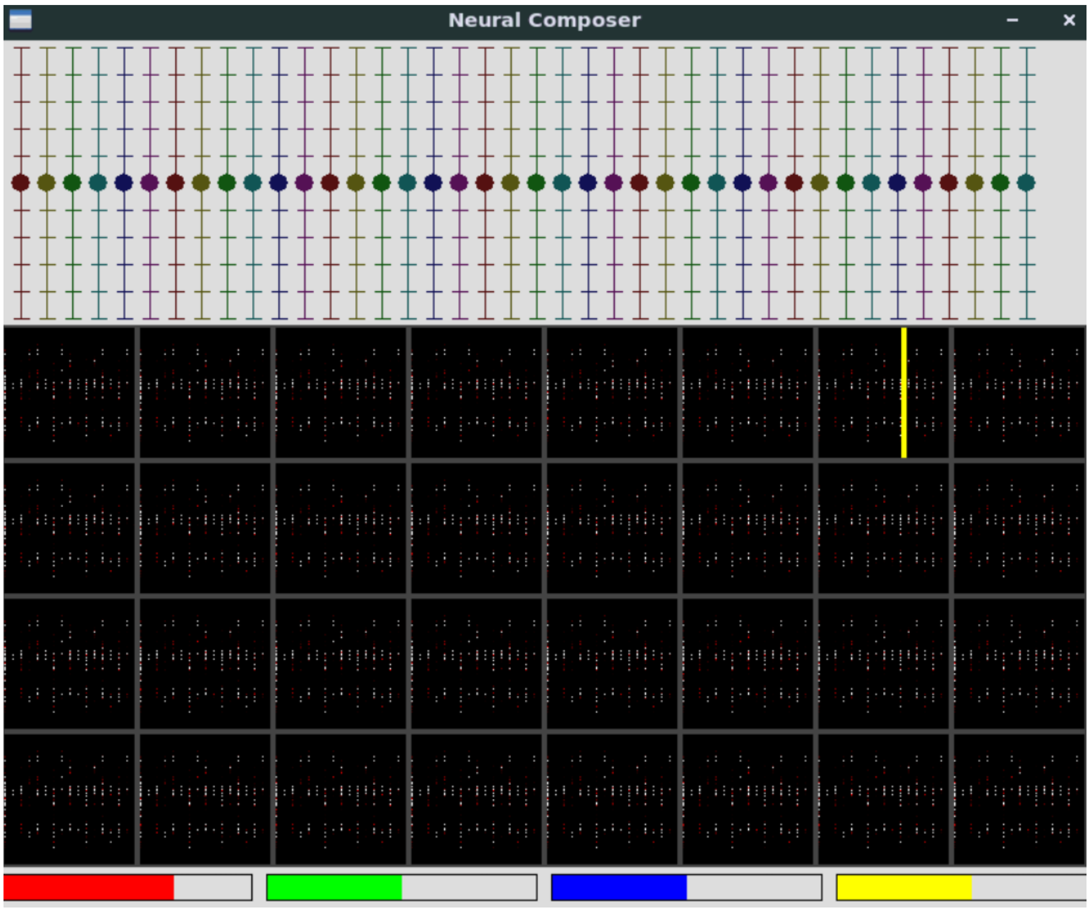


Figure 2.16: Composer interface

3 Methodology

3.1 Setup and approach

In order to achieve our goal of obtaining a reliable model to capture various dimensions of style, we needed to sieve through different models, as presented in section 2.3, finding Polyphony RNN (2.3.3.1) and Composer (2.3.5) as possible models to solve our problem.

With this work, we hope to not only better understand how style can be perceived in terms of purely melodic, harmonic, and rhythmic structure but also how we can devise a model to perform this task. The chosen models, having worked with the datasets that they were originally intended to work with, we now want to know if they are suited for generating music in other styles. We also hope to understand possible ways of, either creating a new model or to optimize one of the pre-selected models, to consistently generate music on the styles given by our datasets that each model trains on.

After testing with these models, we concluded that Composer can consistently produce an output that neither lacks structure nor overly repeats itself, making it our ideal choice to optimize, as better explained in section 3.2.

3.1.1 Dataset collection

To train our model, we need to collect the datasets chosen in section 2.1.5. In order to do so, we need to retrieve their MIDI files from the Web:

- The Bach's chorales dataset was retrieved from (Bach Dataset, 2018)
- The Videogame NES soundtrack dataset was retrieved from (Videogames Dataset, 2019)
- The Pop songs dataset was retrieved from (Pop Dataset, 2020)
- The Mozart pieces dataset was retrieved from (Mozart Dataset, 2020)
- The Queen songs dataset was retrieved from (Queen Dataset, 2020)
- The Dragonforce songs dataset was retrieved from (Dragonforce Dataset, 2020)

These datasets have the particularity of varying in size, style specificity, and historical period; for instance, Queen or Dragonforce are more specific datasets than Pop, and the style of Mozart can also be compared historically with the rest of the datasets. Although Bach was only used as a baseline and not to generate the final results, it served to help us better understand the models.

3.1.2 Model training

Now that we have collected our datasets, we can focus on training our model. For that, a script, now integrated at the initialization of training, converts and loads the dataset into a numPy format for the training program to use. After that, we run the model with our default values and check the number of epochs needed to either converge or reach an acceptable number of epochs (in our tests, 100).

The first training optimizations that this model received were related to hyperparameter tuning, as shown in the tables below, in order to have the model converging faster:

	Batch momentum			
		0.9	0.99	0.999
	0.002	21	23	21.3333
Learning rate	0.0025	20.6667	22.3333	20
	0.003	20.6667	17	17
	0.0035	22.6667	21.6667	24

Table 3.1: Training epochs for certain combinations of learning rate and batch momentum to obtain a given loss value (0.0015), lower is better

With the results shown at table 3.1, we can deduct that for our model, to a given dataset, we can get, on average, a better loss value if we choose a 0.99 batch momentum. Since our learning rate will be adaptive, as inspired by (Xu & Metz, 2019), we can consider that a learning rate, at most twice the minimum proposed (0.0015), may suffice.

An adaptive learning rate is simply a small decrement in the value itself, every time the loss value increases, until we hit a pre-established minimum, in hopes of driving our model in the right direction.

	Batch size			
		5%	10%	25%
	0.0025	18.6667	19.6667	33.3333
Learning rate	0.00275	21.3333	20.3333	30.3333
	0.003	23.3333	28	28.33333

Table 3.2: Mini batch experimental confirmation, for a fixed loss (0.0015) and batch momentum values (0.99), lower is better

From table 3.2 we can conclude that lower batch sizes, along with smaller learning rate starting values allow for better convergence. However, for larger datasets, a smaller learning rate may lead to very large training times and thus, we may benefit from using larger batch sizes, up to 10 per cent, for similar convergence while spending less time.

The observed better convergence at smaller batch sizes may be due to a greater flexibility related to how the model captures the latent values, being capable of focusing in certain nuances, allowing it to converge earlier.

3.1.3 Model optimizations to the interface

Since there is room for improvement on not only the model itself but also on the interface that we use to generate music, shown in figure 2.16, we can first look at how the model behaves. Aside from the hyperparameter optimizations talked about in section 3.1, we have an interface where we can operate with the obtained latent vector by moving a set of knobs to change the factors for each of the values. Below we can see the relative importance of each of the multipliers/knobs on the produced output:

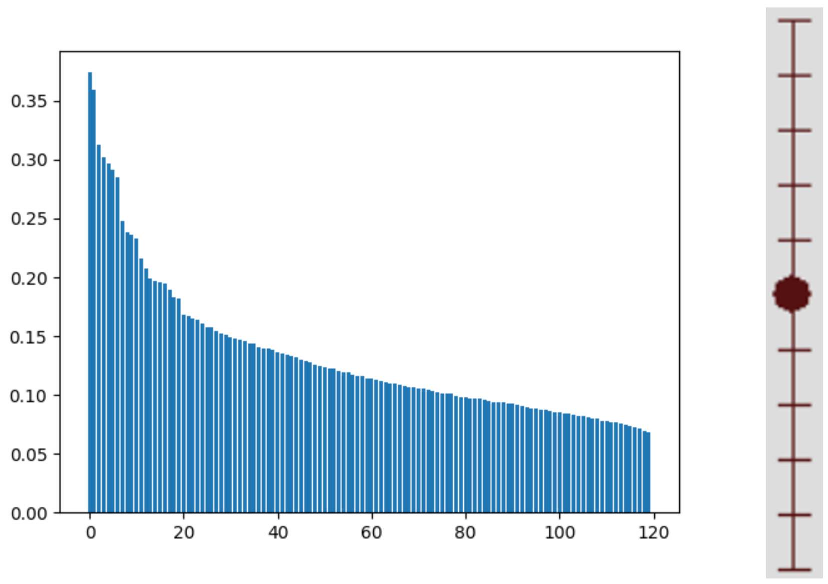


Figure 3.1: Weight distribution of the latent vector for the NES videogames dataset (left) Tuning knob for the interface (right)

Each bar represents a different knob, with the most important ones being visible on the interface for fine-tuning. The least important weights do not appear as knobs for us to directly change, but a key can be pressed to randomize these ”hidden” parameters. An appendix is included with all the details to use the interface.

Our interface was also optimized in a way that allows us to change the key of the track by multiplying or dividing every pitch by $\sqrt[12]{2}$, as this value represents the smallest interval ratio in western music, the semitone. With the associated keybindings we can freely change the key of the track, semitone by semitone.

A waveform similar to the one of a piano was also added, using numPy to encode a function generator, by adding the first sine Fourier components of a piano sound wave.

3.1.4 Output generation

In order to generate our output, we can rely on our interface to extract the results. By varying the knobs, or by randomizing their values, we can start obtaining our outputs as we hear it since the model works in real-time, with a set of waveforms we can choose from.

However, these outputs may need to be hand-picked since some of the results may not reflect the dataset as intended, and since we're trying to evaluate how well we can replicate style with our model, and to improve our results, we can straight ahead remove songs that don't sound as good as expected and also avoid parameters that, at least for some datasets, may not make sense, such as the note threshold, (the red bar in the interface) where increasing it too much may lead to harmonies with many undesired dissonances.

We can now either export our files as MIDI to be sampled in a sampler or directly as WAV files for generation. For some models, like the videogames one, since NES' audio relied in waveforms that require lower encoding, it may make sense for us to use the directly generated WAV files. We can then, for each dataset, have an associated sampler to recreate our outputs.

For the sampled MIDI files, a different sampler was used for each of the datasets, to help better distinguish them. For Mozart, a piano sampler was used; for Dragonforce, a distorted guitar sampler; for Queen, an electric piano; for Pop, a synthesizer piano. The NES videogames used directly the waveforms from the generator as they better fit the style of these videogames. These samplers came from the program Reason 5 but any sampler should perform the job.

3.1.5 Experimental data collection

Now that we have generated our outputs, we can generate our results for the general public to help us understand how well the model performs. Therefore, to fix some metrics, we need to categorize our output into:

- Suitable for outputs that can be seen as possible songs/sections of songs of the style of the training set.
- Sample suitable for outputs with motifs that can be considered of being of the style of the training set.
- Unsuitable for outputs that have no relation to the style of the dataset, making dissonant tracks or tracks endlessly repeating a single motif, be a part of this set as well.

With our now defined metrics in mind, we need to produce an experiment where we can categorize the output accordingly. For that, we will generate a form for public data collection where a selection of handpicked outputs will be listened by the respondents for each of the styles (NES videogames, Pop, Dragonforce, Queen and Mozart), in sets of 3 tracks each, answering:

- Which of the songs is the most enjoyable, to try to pick a best track.
- How good the main voice sounds at the picked track, since the melody standing out is usually important - if the answer is negative, the model for that style may be deemed sample suitable.
- If the track could belong to that style as is, and in what degree - separating even further suitable from sample suitable songs.

It is also asked the musical background and the familiarity with each of the styles for each of the respondents to better categorize the answers. With this setup in mind, we need to finally generate our outputs in order to fully implement our experiment.

3.2 Initial model testing and experiments

We can prove why Composer was chosen over Magenta in a comprehensive comparison of both models, with their functioning and their generated outputs for a same dataset, in this case Bach, to test both models under the same conditions.

First off, the dataset was prepared in a way that the models recognize it. Magenta does this with a note sequence file, converting all the MIDI files into a single file that can then be read by the model for it to be trained. By contrast, Composer takes groups of 16 measures and then treats each group as a single batch, with the complete samples in a numPy file and their respective lengths in another file.

Then, during training, Polyphony RNN trains over the sequence, note by note, calculating a value for accuracy and loss. For Composer, the model generates three files for the final latent values, with an associated loss value, that combined with the previous two files will be part of the output generator.

The model for Composer has been optimized with an adaptive learning rate, meaning that this value decreases every time the loss value increases between different training epochs. This allows to speed up training at the earlier stages. Other hyperparameters have been tested in order to help further optimizing the model. The full process, from dataset collection to music generation is summarized at 3.2.

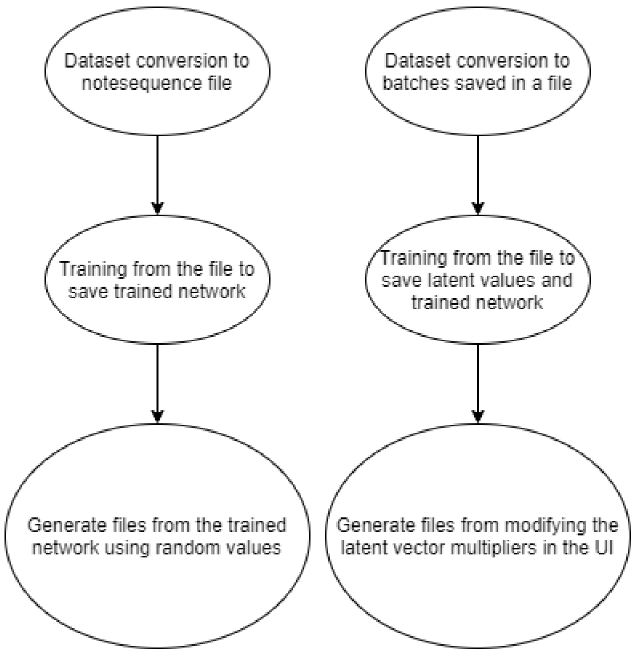

Dataset conversion to notesequence file
Dataset conversion to batches saved in a file
Training from the file to save trained network
Training from the file to save latent values and trained network
Generate files from the trained network using random values
Generate files from modifying the latent vector multipliers in the UI

Figure 3.2: Magenta vs Composer logic

Polyphony RNN's output is generated from the trained network path and may use a set of primer pitches or MIDI files to reference to:

"There are several command line options for controlling the generation process:

- primer_pitches: A string representation of a Python list of pitches that will be used as a starting chord with a quarter note duration. For example: "[60, 64, 67]".

- primer_melody: A string representation of a Python list of note_seq.Melody event values (-2 = no event, -1 = note-off event, values 0 through 127 = note-on event for that MIDI pitch). For example: ”[60, -2, 60, -2, 67, -2, 67, -2]”.
- primer_midi: The path to a MIDI file containing a polyphonic track that will be used as a priming track.
- condition_on_primer: If set, the RNN will receive the primer as its input before it begins generating a new sequence. You most likely want this to be true if you’re using primer_pitches to start the sequence with a chord to establish a certain key. (...)
- inject_primer_during_generation: If set, the primer will be injected as a part of the generated sequence. This option is useful if you want the model to harmonize an existing melody. This option will most likely be used with primer_melody and –condition_on_primer=false.” (Git_Magenta, 2020)

By contrast, Composer utilizes the user interface shown at section 2.3.5 in the figure 2.16 to change the multipliers of the latent vectors while having a real-time playback over 32 measures. This interface also allows for the note threshold, tempo, volume and pitch, as shown respectively in the colored bars, to be adjusted. Each increment allows for:

- Note threshold bar allows the threshold for the notes appearing in red to turn white, meaning that the certainty to play those notes increases and therefore more notes will be played.
- Tempo bar allows for the playback speed to be changed.
- Volume bar changes the output volume.
- Pitch bar allows for each increment to increase or decrease the pitch value of all notes by a semitone (the smallest interval in the 12-TET tuning system, used in Western Music, which all of our datasets belong to) therefore changing the key of the played song.

For Polyphony RNN, the training was performed with a relatively low accuracy, reaching 75 per cent on the best cases and with a loss very close to 1, meaning that the dimension of the errors is relatively high, being reflected on the outputs. These outputs were either unstructured songs without a defined chord progression nor relevant melodic evolution, or very structured songs always repeating the same motifs. Since none of these extremes is of interest to us, the direct usability of this model is questionable.

Composer was tried afterwards, with the training performing with very low loss values (under 0.002) and the produced outputs, although having a few dissonances, are much more diverse while maintaining a chord structure. However, phrase/motif terminations (also known as cadences) appear very rarely and they may occur more often somewhere in the middle of the generated 32 measures, instead of the end. This sole difference in first results is what, due to our time constraints, led to picking the Composer model over Polyphony RNN.

With these choices in mind, and with the Composer model fully optimized as described, we can now proceed to collecting results from respondents. These results will also help reflecting how well the model performs its task.

4 Results and Discussion

4.1 *Expected results and experimental observation*

4.1.1 Expected results

Bearing in mind that our model behaves differently for each of these datasets, having a different trained neural network for each one of them we can expect different results:

- The videogame music dataset is expected to be suitable for most of the outputs, as the structure is very homogeneous (a looping [A B] structure, for example).
- Mozart's pieces are expected to be sample suitable, since the dataset is similar from piece to piece, with shared motifs and cadences although individually each of the pieces have a free-form nature and each has a very heterogeneous internal structure ([A B C D E B], for example).
- Pop music is expected to be sample suitable, since while having a very homogeneous structure ([A B C A B C A], for example), has more instrumental arrangement variety, making it less convenient to deal with, compared, for instance, to Mozart, which is only bound to a single style.
- Queen's songs are expected to be sample suitable or even unsuitable due to their structure variety, since unlike Mozart, the structure varies both across the dataset and in the songs themselves (some songs may have structure [A B C A] and others might have [A B C D E B C], for example).
- Dragonforce style songs are expected to be sample suitable since the individual structure of each of the songs in the dataset is pretty consistent and there is low variation among the different structures (for instance, a song may be [A B C A D A] and another may be [A B C A B A]).

Aside from this analysis, it is to be considered that the differences that exist between the different trained versions of the model, while having the same base hyperparameters, were considered to have negligible impact in how the output is perceived, associating the output differences exclusively to the intrinsic behavior that is respective to the architecture of the model itself.

It is also to be noted that these results are expected since we already know that the model, being far from ideal, is capable of capturing some more prominent musical features very well, such as basic chord structures and harmonies and even some melodies, as shown in 2.3.

4.1.2 Experimental observations from the author

Aside from the presented output generation method, of randomizing parameters, some fine tuning with the knobs and smaller randomization intervals have been tried. From all the tweaks done with the aid of the model's interface, two distinct ways of generating the outputs were identified:

- Using the default randomization interval more diverse results were obtained although more of the generated outputs had to be discarded;
- Using a smaller randomization interval the results were more consistent and way fewer had to be discarded but all the outputs were more similar.

We chose the smaller randomization interval for pop style to check how different the output generation would perform, and, for the remaining styles, we chose the regular default randomization intervals.

Starting with the NES videogames dataset, after listening to the generated results, the style fits, just as expected, and, despite not having a notion of ending, it can loop on itself, making it ideal for the desired environment of videogame soundtracks. Being this what our model was originally designed to do, we can consider that this style has been successfully captured, being suitable for videogame music generation.

Next up, with Pop songs, the model does not sound as good as the videogames' dataset. This may be due to the instrument diversity of this dataset, making it harder for our model to capture some basic arrangement features of this style. This trained model may be therefore classified as being sample suitable due to how diverse Pop is.

With the outputs generated with the piano pieces from Mozart, the captured style resembles Mozart a lot if we ignore the constant pauses that appear in the middle of the track, classifying this version of the model as being sample suitable. The reason why it is not classified as suitable is mostly due to only some motifs being identifiable as possible sections for a Mozart piece.

With Queen, the same problems that Pop suffered also appear due to the instrumental variety. However, we consistently have the same instruments so the range of the tracks does not vary as much. Another problem arises relating to how variable the section sizes are between different songs. These features can render the model sample suitable but, just like with Pop, also need to be carefully selected.

Finally, with the model trained over the Dragonforce dataset, since we have faster notes being played, although this may be a key feature of the style, it makes perceiving the style more complicated. Beyond this issue we can listen to the main voicings of the song and sometimes understand what would be a solo. This makes this model sample suitable but, for some outputs may produce unsuitable results.

4.2 Results obtained from form answers

Bearing in mind some basic notions from the previous output analysis over how our model is expected to behave, we can analyze what our population of thirty one respondents has to say in this matter and check if the statistics are coherent with what we are upholding.

First off, we need to check how trustworthy are the obtained answers. In order to do this, it was asked, in a scale of 1 to 5 how familiar they were with each of the styles, and respondents that were not as familiar with a certain style could skip the section relating to that style, to help us produce more reliable results. It was also asked the level of music theory knowledge in order to further help us legitimize results, according to the average percentages shown below:

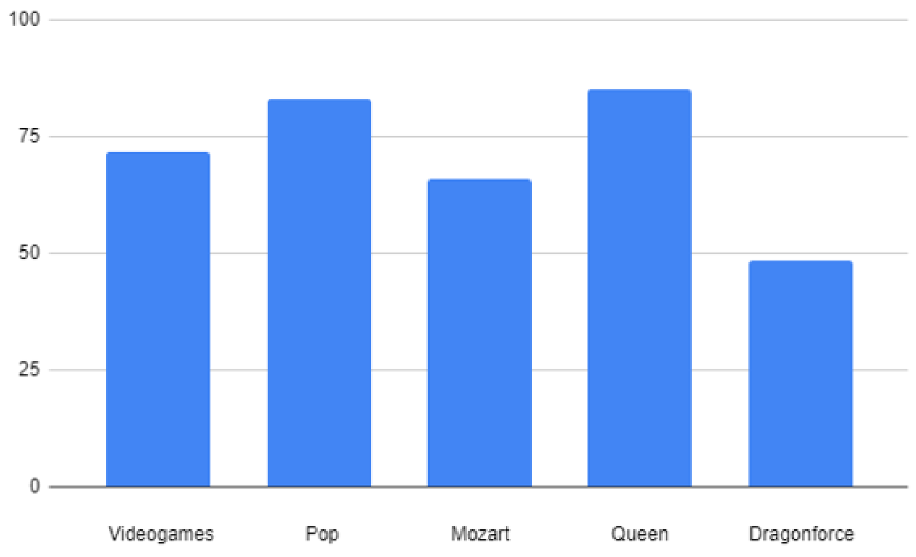


Figure 4.1: Relative familiarity of respondents with the different styles

It was then asked which of the tracks was the most enjoyable. If a certain track has a greater bias than the others, it can be further analyzed why it is more successful than the others.

	Favorite track for each style (%)				
	Videogames	Pop	Mozart	Queen	Dragonforce
Track 1	25,8	29,0	**48,4**	25,8	16,1
Track 2	25,8	22,6	16,1	**38,7**	6,5
Track 3	**32,3**	**38,7**	29,0	35,5	**22,6**

Table 4.1: Tracks chosen by the respondents

It is to be noted that most of the population of our experiment are students from Instituto Superior Técnico, having a different insight from the one that would be provided by professional musicians. And some of the answers may have been obtained with the people only listening to one of the tracks of the dataset since we have no way to ensure that.

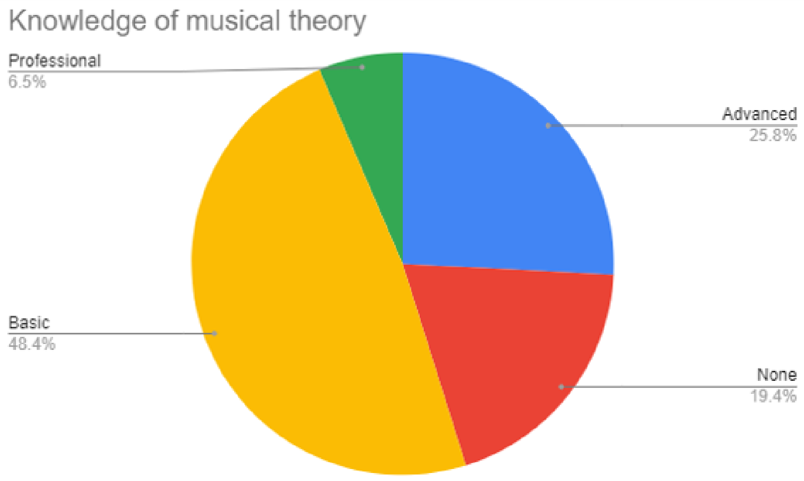


Figure 4.2: Level of musical theory knowledge of the respondents

We can now look at which style has the best/most catchy main voicing. This rating suffers a penalization if the respondents who did not answer to certain styles are accounted. The style that suffered the most with this was Dragonforce where 54,8% of the respondents did not answer to this style, due to their low familiarity. Therefore, the graph shown below does not account for the respondents that did not answer in order to help to even things out.

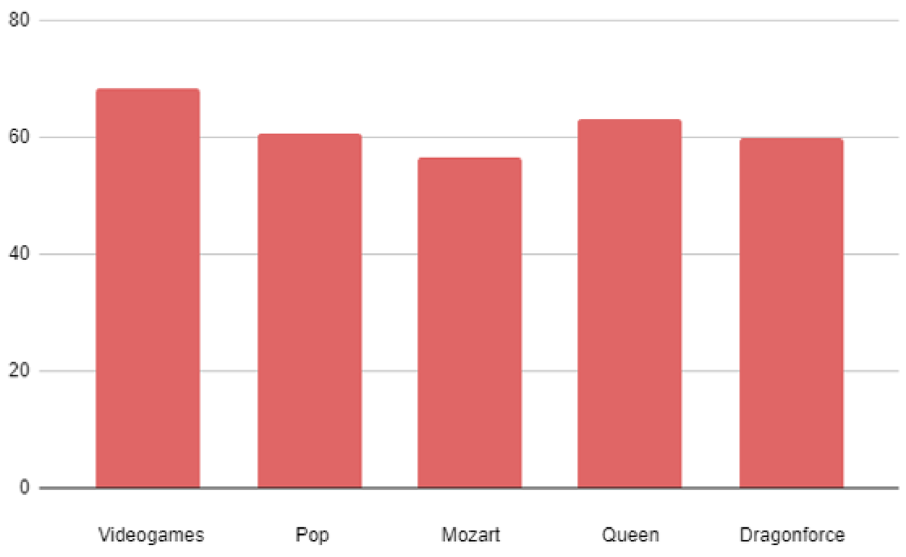


Figure 4.3: Most catchy main voice for each of the styles, according to respondents

Finally, we can check the suitability for each of the trained neural networks in four distinct categories:

- As proposed by our methodology, an unsuitable output.
- For sample suitable outputs, checking if either a lot or a few changes need to be made, in order to help further separating suitable from unsuitable outputs and how close those outputs are from being considered suitable.
- And finally, the outputs that are suitable.

	Suitability for each style (%)				
	Videogames	Pop	Mozart	Queen	Dragonforce
Unsuitable	0,0	19,4	9,7	22,6	3.2
Requires a lot of changes	9,7	32,3	45,2	38.7	25.8
Requires a few changes	54,8	29,0	32,3	29.0	12.9
Suitable	19,4	9,7	6,5	9,7	3.2
Not answered	16,1	9,7	6,5	0,0	54,8

Table 4.2: Suitability of each of the models according to the respondents

4.3 Critique and discussion

4.3.1 Discussion of Form Results

We can now look at the results and compare them to the expected results. This way, we can establish where the model performed better and worse and see if the ideas of structure and harmony comply with these results.

First off, with the NES Videogames dataset, we obtained what could be considered a suitable output, since none of the respondents said that the output was unsuitable and more than half (54,8%) claimed that only a few changes would be required for them to consider the tracks as being part of a videogame soundtrack. The output for this style was also the one that best performed in terms of a melodic line, meaning that some themes may be good enough to be catchy.

For the Pop style, the outputs were sample suitable, with answers claiming that changes need to be made, but how far the model is from being suitable, is harder to tell, since both options of requiring a lot of changes or a few changes have close results (32,3% versus 29,0%). The third track was chosen as being the favorite, probably because it is the one with a structure that may more closely resemble what would be expected from the Pop style. It performed average in terms of its melodic line, probably due to having a track of a style that lives from the instrumental diversity being performed on a single instrument.

With Mozart, the outputs also got to be sample suitable but this time 45,2% of the respondents said that the model needs a lot of changes. A possible problem relating to this is the fact that all the notes that have been produced by the model have the same duration and are relatively short, creating pauses in many places that would have longer notes in a regular composition. The track that performed better was the first one most likely due to its more melodic main line. Despite this, it was still deemed as being the worst dataset in terms of its lead line.

Every respondent listened to the tracks corresponding to the output of the Queen dataset, and both the second and third track got the most votes (38,7% and 35,5% respectively). More than a fifth of the respondents (22,6%) claimed that the output is unsuitable even though two thirds of the population claimed that the model is, in a way or another, sample suitable. This may be due to the previously referred problem that this dataset has of having variable amounts of sections and section sizes, making it very hard for our model to grasp its structure. Despite this, the lead melodic line was the second best.

Last but not least, for the Dragonforce dataset, the one that least respondents answered to (only 45,2% of the respondents), the majority of the respondents claimed that it required a lot of changes, probably having to do, just like with the Pop style, with the instrumental variety being poorly portrayed here. The third track was the most popular one, possibly due to having a more melodic bass line contrasting with what may be perceived as a lead guitar as this model also performed average on its lead melodic line.

In general, the model performed as expected, with the NES Videogames style being the best performing one as it stands for what this model, prior to its optimizations, had been originally designed for. For the other styles, some improvements need to be made, with some solutions suggested in our critique.

4.3.2 Critique to the model and posed problems

The model has performed reasonably and along what was expected. However, it is still missing some features that would better capture style. For the model to be ideal, all the generated tracks should hit the suitable mark, which checks to be true for most of the generated videogame soundtracks, but lack, in a way or another, on the remaining styles.

Our first possible problem to be tackled would be the capturing of the batches of the datasets, which ends up dividing them in batches of sixteen measures. To solve this, we would need to have the several sections marked, having, for example, an "A" section with the measure where it occurs marked and a distinct "B" section marked as well, in order to help producing better results. This approach would also solve the problem of the songs not having a structured ending.

Some problems that are expected to arise for the creation of this proposed solution have to do with:

- Allowing a variable batch size, separating batches into different sections and a batch for each of the songs in the dataset, making each of these batches also vary in size.
- Having a separate trained neural network for each of the sections.
- Joining both these neural network structures in a new architecture to help creating coherent outputs.

On top of that, having each of these sections trained separately could help further improve our outputs. This has shown to be particularly an issue with the Queen dataset where each of the sections' size varies a lot, with the output not having any particularly distinguishable or memorable sections.

Another problem is about the separation of tracks for multi-instrument datasets while allowing for training of them at the same time. For instance, if the model could perceive the different instruments, even reproducing them at the output, a different level of style could be perceived by our model: the arrangement. A problem that is also associated with this is the fact that all the notes have the same duration, regardless of what that should be, impacting the output too and aspects associated with arrangement could also help solve this.

Both of the previously proposed solutions would together allow for outputs that have both a coherent structure, due to section separation, and an arrangement that would now be more bound to the style, due to permitting the capturing of the respective instrumental variety.

The last problem that needs to be analyzed is how the outputs are generated. Although ideally the outputs, being randomly generated, should always produce suitable results, assuming the prior solutions have been applied, they could still not always be suitable. Even though we have tried using a smaller randomization interval for the Pop style, the results do not seem to be any different from the other datasets and thus it may be worth looking at a different approach.

It could, for a possible solution, either be tried an increase on the size of the latent vector, since with the current size a lot of its values have an high impact, being heavily correlated, making it hard to perceive what each one does; or a classifier to perceive what musical aspect each of the positions of the vector may correspond to. This may not be as defining as the other problems but it is something worth considering nonetheless.

Our results, with the author and the respondents having similar opinions over the same outputs for the same datasets, prove our initial ideas and consequent problems that arise from the behaviour of the model. Thus, our critiques can be validated in terms of hypothesizing the problems that our model has. With this in mind, we can now extract conclusions from this work and provide some insight over possible future works.

5 Conclusions and Future Work

5.1 Conclusions

With this thesis we have optimized a model in order to partially fulfil our goal of capturing style and reproducing it in new outputs. This model was successful with:

- Generating composition ideas to a given style. This could help later create a tool for musicians/producers to extract ideas from other works and help with inspiring new works.
- Capturing some styles. The model behaved better for styles with a more coherent structure both at the songs themselves and across the different elements of the explored datasets.
- Creating a baseline for better models. The critiques explored in 4.3.2 settle a baseline for possible future models to be worked from here on, as an upgraded version of our model.

This model was not successful with:

- Instrumental arrangements, as these were proven to be a crucial part of perceiving style, even for the most trained ears, requiring the usage of external samplers.
- Coherent outputs, as a random generation should, ideally, always allow for good results (or at least at the majority of cases), requiring hand-picking the outputs.
- Structure capturing, where varying section sizes were not captured by the model, compromising directly how well some of the results ended up capturing style.

We hope that with the baselines born from this work, better models may arise bearing these ideas in mind.

5.2 Future Work

Some possible implementations or optimizations of this work may arise, namely:

- An application for musicians and producers to tinker with, in order to speed up the creative process.
- As a follow-up of the previous topic, an application to blend different styles for generating new musical ideas.

- A version of this model that preserves both the original section types, intrinsic to the style to be captured, and the original arrangements, providing different possible arrangements on a given musical style.
- A more fine hyperparameter selection, or perhaps a better architecture for the neural network behind the model.

Being the instrumental arrangement and structuring of sections a core concept of understanding musical style, this optimization can be considered to be the most relevant to be performed.

As for creating an application, we can look at the baseline from the current interface of the model to create a conceptual view of what a possible interface for this application could look like. This interface should make the process of training a neural network completely hidden from the end user, by further optimizing our model. It is to be accounted that these preoccupations may vary, according to who the end-users of the application may be.

Bibliography

Braguinski, N. (2019, November). *"428 millions of quadrilles for 5s. 6d.": John clinton's combinatorial music machine* (Vol. 43). https://doi.org/10.1525/ncm.2019.43.2.86.

Briot, G. H., Jean-Pierre, & Pachet, F.-D. (2019, August). *Deep learning techniques for music generation – a survey.* http://arxiv.org/abs/1709.01620.

Cabaret player piano. (2020). Retrieved from https://www.piano4u.com/Piano/Detail/2233

Christensson, P. (2017, July 15). *Midi definition.* https://techterms.com/definition/midi. TechTerms.

Chung, C. G. K. C., Junyoung, & Bengio, Y. (2014, December). *Empirical evaluation of gated recurrent neural networks on sequence modeling.* http://arxiv.org/abs/1412.3555.

Classical piano midi page - mozart. (2020). Retrieved from http://www.piano-midi.de/mozart.htm

ClewsOctober, D. (2020). *The 15 best midi keyboards 2020: Mac, pc, iphone and ipad midi controller keyboards for beginners to pros.* MusicRadar. Retrieved from https://www.musicradar.com/news/the-best-midi-keyboards-our-favourite-laptop-desktop-and-ios-keyboards

Deep learning concepts. (2020, November 11). https://appen.com/ai-glossary/.

Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., & Sutskever, I. (2020). Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*.

Dobrin, P. (2019, April 10). *Itzhak perlman feels the philly love at philadelphia orchestra special concert.* Retrieved from https://www.inquirer.com/arts/itzhak-perlman-philadelphia-orchestra-concert-review-20190410.html

Download midi melodies - dragonforce — midimelody. (2020). Retrieved from http://en.midimelody.ru/dragonforce/

DuBreuil, A. (2020). *Hands-on music generation with magenta: Explore the role of deep learning in music generation and assisted music composition.* Packt Publishing. Retrieved from https://books.google.pt/books?id=SUvODwAAQBAJ

Floriancolombo/bachprop. (2018, December 14). Retrieved from https://github.com/FlorianColombo/BachProp

Futterman, S. (2016, August). *Miles Davis: 15 Essential Albums.* Retrieved 2020-12-03, from https://www.rollingstone.com/music/music-lists/miles-davis-15-essential-albums-247620/

Glossary of terms. (2020). https://www.berklee.edu/core/glossary.html. Berklee College of Music.

Graves, A. (2014, June). *Generating sequences with recurrent neural networks.* http://arxiv.org/abs/1308.0850.

HackerPoet. (2018, November 26). Retrieved from https://github.com/HackerPoet/Composer

Hadjeres, G., Pachet, F., & Nielsen, F. (2017, 06–11 Aug). DeepBach: a steerable model for Bach chorales generation. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international conference on machine learning* (Vol. 70, pp. 1362–1371). International Convention Centre, Sydney, Australia: PMLR. Retrieved from http://proceedings.mlr.press/v70/hadjeres17a.html

Hyperparameter. (2019a, May 17). https://deepai.org/machine-learning-glossary-and-terms/hyperparameter.

Midi db — free midi files. (2020). Retrieved from https://www.mididb.com/

Miranda, I. (2020, January 28). *Kiss: nova etapa da turnê final terá mudanças no repertório, diz Tommy Thayer.* Retrieved from https://whiplash.net/materias/news_747/315731-kiss.html

Mozart musikalisches würfelspiel (mozart's musical dice game). (2012, July 18). https://thebrickinthesky.wordpress.com/2012/07/18/mozart-musikalisches-wurfelspiel-mozarts-musical-dice-game/.

Neural network. (2019b, May 17). https://deepai.org/machine-learning-glossary-and-terms/neural-network.

Oord, N. K. O. V. L. E. A. G. e. K. K., Aaron van den. (2016, 18 June). *Conditional image generation with pixelcnn decoders.* http://arxiv.org/abs/1606.05328.

pelegk11. (2020, January 10). Retrieved from https://github.com/pelegk11/Composer

Piano roll editor. (2020). Retrieved from https://musescore.org/en/print/book/export/html/278688

Queen midi files. (2020). Retrieved from http://tomqueen.tripod.com/midis.html

Roberts, A., Kayacik, C., Hawthorne, C., Eck, D., Engel, J., Dinculescu, M., & Nørly, S. (n.d.). Magenta studio: Augmenting creativity with deep learning in ableton live. In *Proceedings of the international workshop on musical metacreation (mume).*

Shlomo Argamon, K. B., & (Eds.), S. D. (2010). The structure of style: Algorithmic approaches to understanding manner and meaning. In (pp. 45–58). Berlin: Springer-Verlag. https://www.cs.cmu.edu/~rbd/papers/rbd-style-2009.pdf.

Technologies, A. (2020). Retrieved from https://creators.aiva.ai

Tf magenta. (2020, May). https://github.com/tensorflow/magenta/.

Time Signatures 4/4 - KNILT. (2018, February 28). Retrieved 2020-12-02, from https://knilt.arcc.albany.edu/Time_Signatures_4/4

Training a neural network on midi data with magenta and python. (2019, October 4). https://drive.google.com/file/d/14e0MCJD7RH_m7CpsFZWPIpO0WgQrwi64/view?usp=embed_facebook. Retrieved from https://www.twilio.com/blog/training-a-neural-network-on-midi-music-data-with-magenta-and-python

Xu, A. M. D. J. K., Zhen, & Metz, L. (2019, September). *Learning an adaptive learning rate schedule.* http://arxiv.org/abs/1909.09712.

I

Appendices

A Composer Interface commands

Composer Controls:

- Right Click - Reset all sliders
- 'R' - Random Song multiplying each of the slider by a random value that is at most the standard deviation
- 'E' - Random Song multiplying each of the slider by a random value that is at most 2 times the standard deviation
- 'M' - Save song as .mid file
- 'W' - Save song as .wav file
- 'O' - opens the loaded samples and plays the reconstructed version of one of the input songs
- 'Escape' - Quit
- 'Space' - Play/Pause
- 'Tab' - Seek to start of song
- '1' - Square wave instrument
- '2' - Sawtooth wave instrument
- '3' - Triangle wave instrument
- '4' - Sine wave instrument
- '5' - Circle wave instrument
- Red Slider - Threshold to play a note
- Green Slider - Speed of song
- Blue Slider – Volume

NEW OPTIMIZATIONS:

- Yellow Slider – Pitch changer (modifies the key of what's being played)
- 'P' – Resets pitch
- '6' – Piano wave-like instrument

Imported from (pelegk11, 2020) fork:

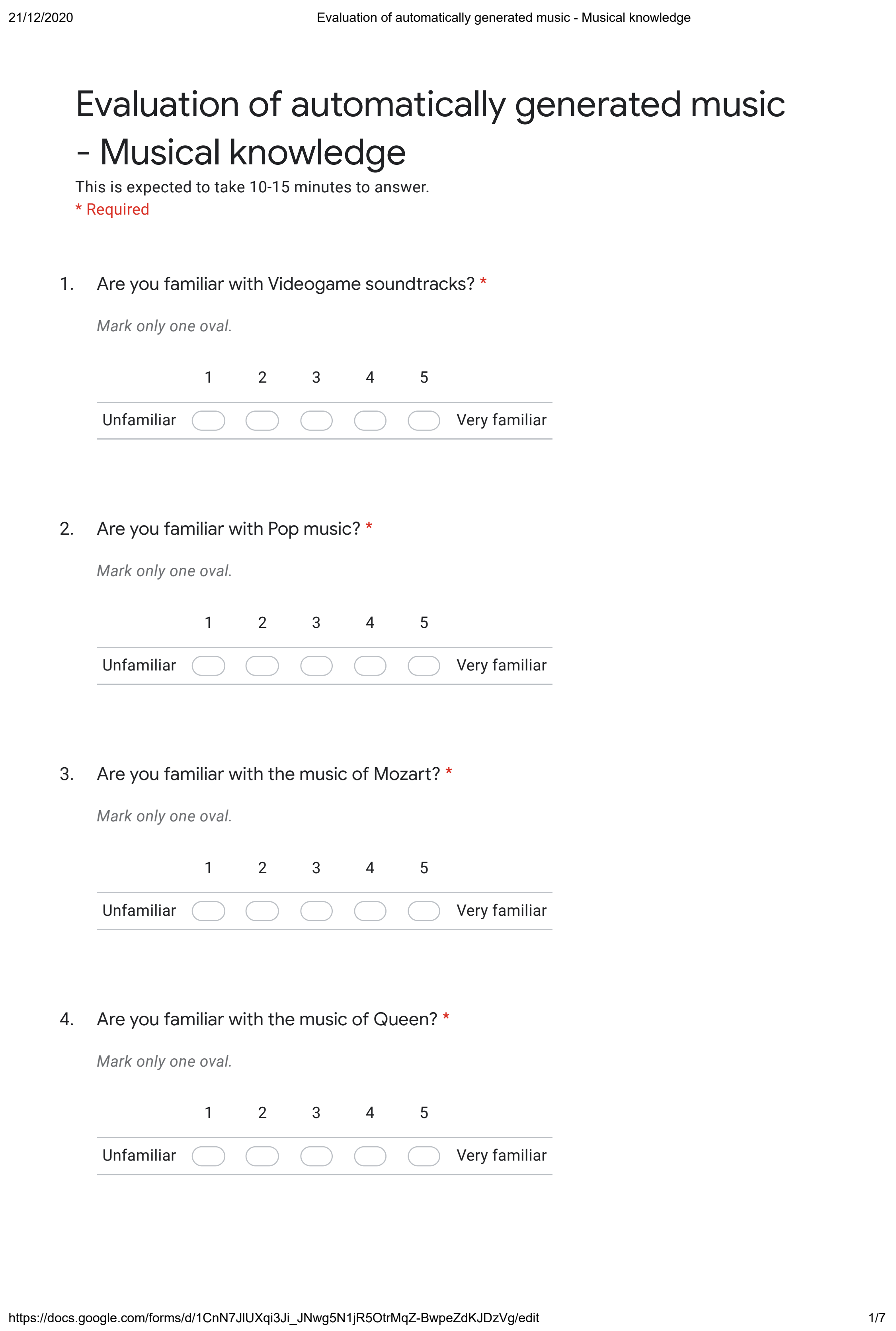
- 'T' - Randomizes last 2/3 of sliders
- 'X' - Randomly alters each slider (adds a random number that's 10 per cent of the standard deviation to the current value)
- ',' - Multiplies each slider by 1.1

- '.' - Divides each slider by 1.1
- '-' - Multiplies each slider by -1 (changed from the original)
- Up Arrow - increments the note threshold (Useful for trying to add one note at a time)
- Down Arrow - decrements the note threshold (Useful for trying to remove one note at a time)

Unused features on this work also from (pelegk11, 2020):

- 'S' - Save slider values to text file (includes threshold, speed and volume sliders, model path and instrument number) Asks in command line what filename to save as To load a saved song run the command py composer.py –model_path song1.txt It knows to load the saved song file because of the .txt extension and gets the model path from inside the file If blend mode is on, it will save a file containing the names of the blended songs that can be opened in the same ways
- 'L' - Loads a song Asks in command line what song file (created by the S command above) to open and play without closing the app
- 'A' - Toggles auto save whenever the song finishes, it is saved and a new song is randomly generated.
- 'B' - Blends smoothly through a series of preset songs by taking a linear combination of the latent vectors. Asks in command line what song files (created by the S command above) to blend.

B Google Form



Evaluation of automatically generated music - Musical knowledge

This is expected to take 10-15 minutes to answer.

* Required

1. Are you familiar with Videogame soundtracks? *

Mark only one oval.

	1	2	3	4	5	
Unfamiliar	◯	◯	◯	◯	◯	Very familiar

2. Are you familiar with Pop music? *

Mark only one oval.

	1	2	3	4	5	
Unfamiliar	◯	◯	◯	◯	◯	Very familiar

3. Are you familiar with the music of Mozart? *

Mark only one oval.

	1	2	3	4	5	
Unfamiliar	◯	◯	◯	◯	◯	Very familiar

4. Are you familiar with the music of Queen? *

Mark only one oval.

	1	2	3	4	5	
Unfamiliar	◯	◯	◯	◯	◯	Very familiar

5. Are you familiar with power metal (like Dragonforce)? *

Mark only one oval.

	1	2	3	4	5	
Unfamiliar	⬭	⬭	⬭	⬭	⬭	Very familiar

Evaluation of automatically generated music - AI Videogame Compositions

6. Listen to the following tracks, supposedly following the Videogame soundtrack style (check the box afterward): https://bit.ly/AIVGSoundtracks *

Mark only one oval.

⬭ Tracks listened.
⬭ If you are not familiar with videogame soundtracks, you can skip this section.
Skip to question 10

Evaluation of automatically generated music - AI Videogame Compositions

7. Which track do you identify as being the most enjoyable? *

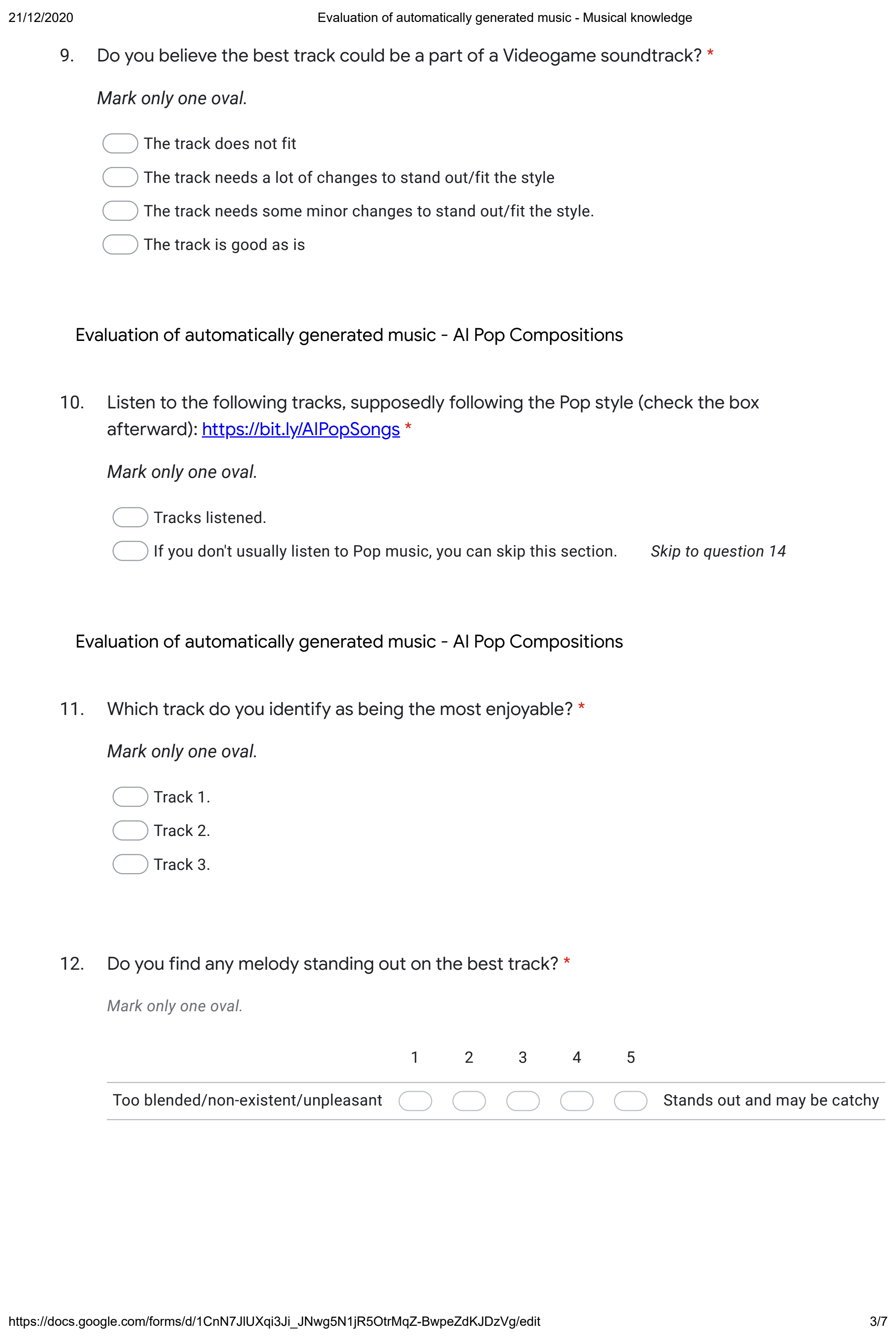
Mark only one oval.

⬭ Track 1.
⬭ Track 2.
⬭ Track 3.

8. Do you find any melody standing out on the best track? *

Mark only one oval.

	1	2	3	4	5	
Too blended/non-existent/unpleasant	⬭	⬭	⬭	⬭	⬭	Stands out and may be catchy



21/12/2020 Evaluation of automatically generated music - Musical knowledge

9. Do you believe the best track could be a part of a Videogame soundtrack? *

Mark only one oval.

- The track does not fit
- The track needs a lot of changes to stand out/fit the style
- The track needs some minor changes to stand out/fit the style.
- The track is good as is

Evaluation of automatically generated music - AI Pop Compositions

10. Listen to the following tracks, supposedly following the Pop style (check the box afterward): https://bit.ly/AIPopSongs *

Mark only one oval.

- Tracks listened.
- If you don't usually listen to Pop music, you can skip this section. *Skip to question 14*

Evaluation of automatically generated music - AI Pop Compositions

11. Which track do you identify as being the most enjoyable? *

Mark only one oval.

- Track 1.
- Track 2.
- Track 3.

12. Do you find any melody standing out on the best track? *

Mark only one oval.

	1	2	3	4	5	
Too blended/non-existent/unpleasant	○	○	○	○	○	Stands out and may be catchy

https://docs.google.com/forms/d/1CnN7JlUXqi3Ji_JNwg5N1jR5OtrMqZ-BwpeZdKJDzVg/edit 3/7

13. Do you believe the best track could be a part of a Pop song? *

Mark only one oval.

- ◯ The track does not fit
- ◯ The track needs a lot of changes to stand out/fit the style
- ◯ The track needs some minor changes to stand out/fit the style.
- ◯ The track is good as is

Evaluation of automatically generated music - AI Mozart Compositions

14. Listen to the following tracks, supposedly following the style of Mozart's pieces (check the box afterward): https://bit.ly/AIMozart *

Mark only one oval.

- ◯ Tracks listened.
- ◯ If you don't know any Mozart composition, you can skip this section. *Skip to question 18*

Evaluation of automatically generated music - AI Mozart Compositions

15. Which track do you identify as being the most enjoyable? *

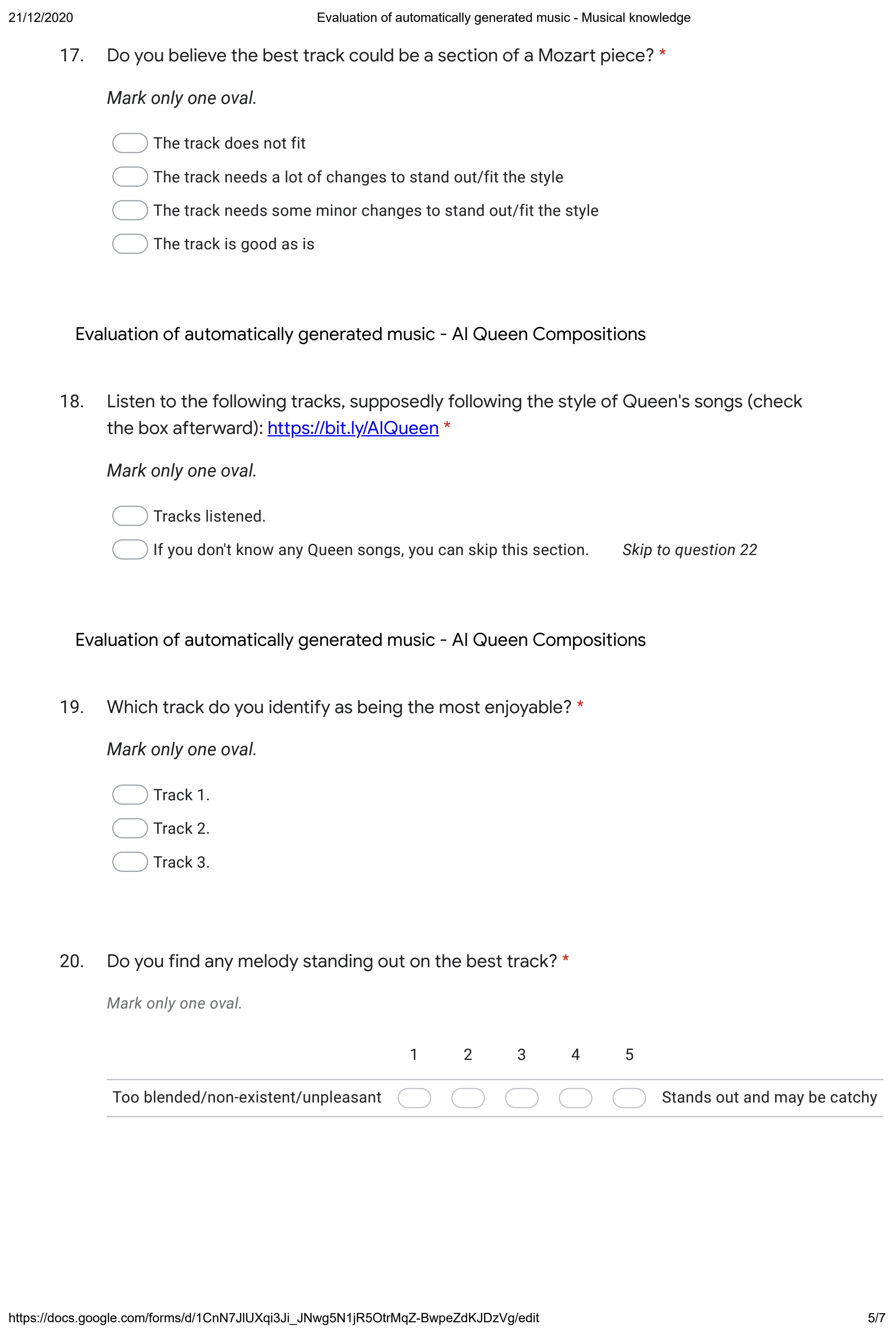
Mark only one oval.

- ◯ Track 1.
- ◯ Track 2.
- ◯ Track 3.

16. Do you find any melody standing out on the best track? *

Mark only one oval.

	1	2	3	4	5	
Too blended/non-existent/unpleasant	◯	◯	◯	◯	◯	Stands out and may be catchy



21/12/2020 Evaluation of automatically generated music - Musical knowledge

17. Do you believe the best track could be a section of a Mozart piece? *

Mark only one oval.

- The track does not fit
- The track needs a lot of changes to stand out/fit the style
- The track needs some minor changes to stand out/fit the style
- The track is good as is

Evaluation of automatically generated music - AI Queen Compositions

18. Listen to the following tracks, supposedly following the style of Queen's songs (check the box afterward): https://bit.ly/AIQueen *

Mark only one oval.

- Tracks listened.
- If you don't know any Queen songs, you can skip this section. *Skip to question 22*

Evaluation of automatically generated music - AI Queen Compositions

19. Which track do you identify as being the most enjoyable? *

Mark only one oval.

- Track 1.
- Track 2.
- Track 3.

20. Do you find any melody standing out on the best track? *

Mark only one oval.

	1	2	3	4	5	
Too blended/non-existent/unpleasant						Stands out and may be catchy

https://docs.google.com/forms/d/1CnN7JlUXqi3Ji_JNwg5N1jR5OtrMqZ-BwpeZdKJDzVg/edit 5/7

21. Do you believe the best track could be a part of a Queen song? *

Mark only one oval.

- ◯ The track does not fit
- ◯ The track needs a lot of changes to stand out/fit the style
- ◯ The track needs some minor changes to stand out/fit the style
- ◯ The track is good as is

Evaluation of automatically generated music - AI Dragonforce Compositions

22. Listen to the following tracks, supposedly following the style of Dragonforce's songs (check the box afterward): https://bit.ly/AIDragonForce *

Mark only one oval.

- ◯ Tracks listened.
- ◯ If you don't know any Power Metal songs you can skip this section. *Skip to question 26*

Evaluation of automatically generated music - AI Dragonforce Compositions

23. Which track do you identify as being the most enjoyable? *

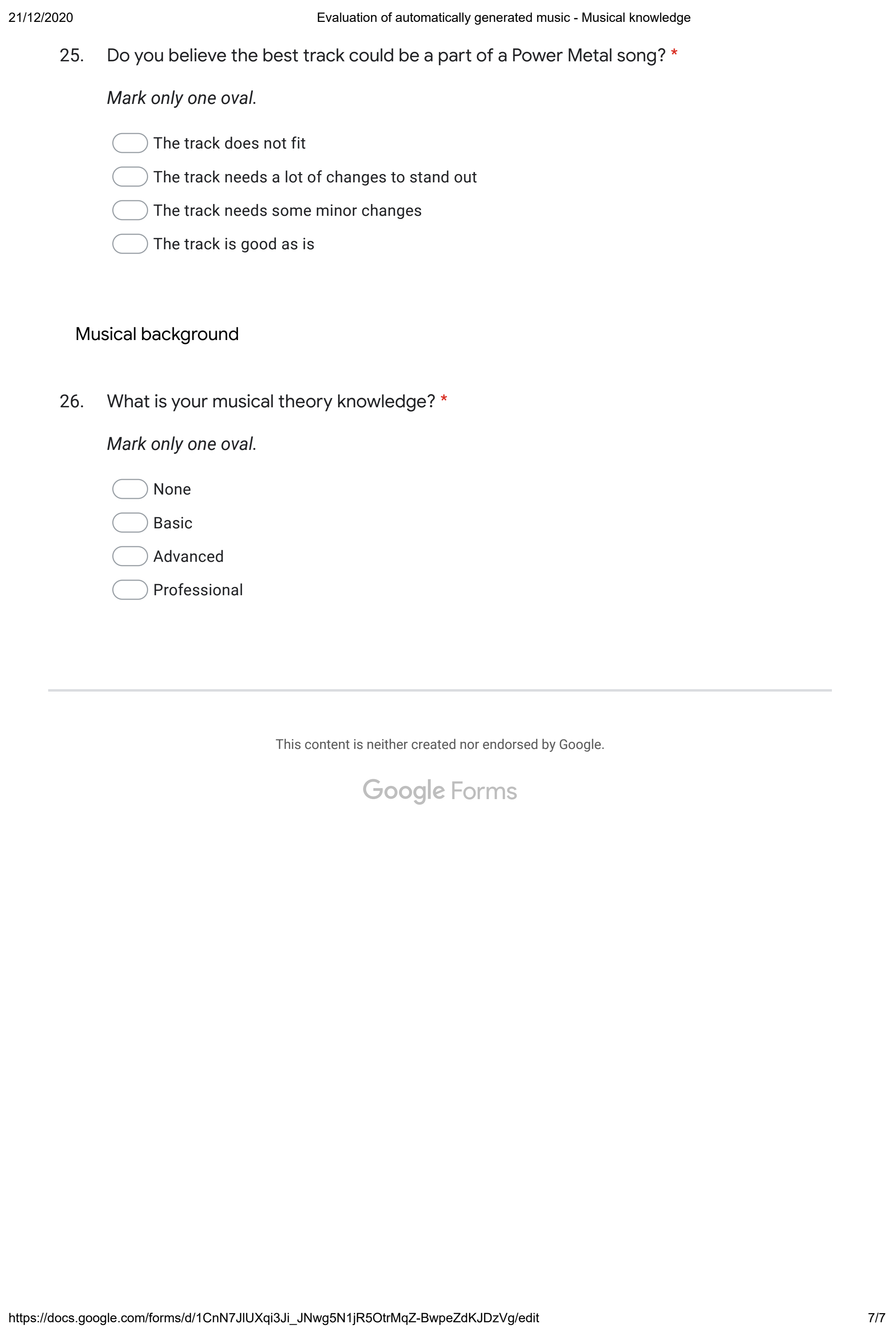
Mark only one oval.

- ◯ Track 1.
- ◯ Track 2.
- ◯ Track 3.

24. Do you find any melody standing out on the best track? *

Mark only one oval.

	1	2	3	4	5	
Too blended/non-existent/unpleasant	◯	◯	◯	◯	◯	Stands out and may be catchy



25. Do you believe the best track could be a part of a Power Metal song? *

Mark only one oval.

◯ The track does not fit
◯ The track needs a lot of changes to stand out
◯ The track needs some minor changes
◯ The track is good as is

Musical background

26. What is your musical theory knowledge? *

Mark only one oval.

◯ None
◯ Basic
◯ Advanced
◯ Professional