Towards a Style-driven Music Generator

Hélder Duarte, Instituto Superior Técnico

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.

1. Introduction

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 (2019) or using dice, as with Mozart's dice game.

1.1. 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 (2014).

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”.

1.2. 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

In this section we will look at how we can define style to better comprehend how our model shall work and which models we have available to work with.

2.1. Musical considerations and Dataset Choice

As defined in Shlomo Argamon and (Eds.), 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, 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.).

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.

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.

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.

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. 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.

2.2.1. AIVA

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. Technologies (2020)

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.

2.2.2. 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.

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."

2.2.3. 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.

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, instead of regular nodes, capable of being trained. Graves (2014)

2.2.4. DeepBach

Another model that uses MIDI is DeepBach Hadjeres et al. (2017), which consists of an architecture using two separate Deep Neural Networks and computes results with the help of a third Neural Network 1.

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.

2.2.5. Composer

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. A third dimension is added to account for each and every measure of both the generated and training songs.

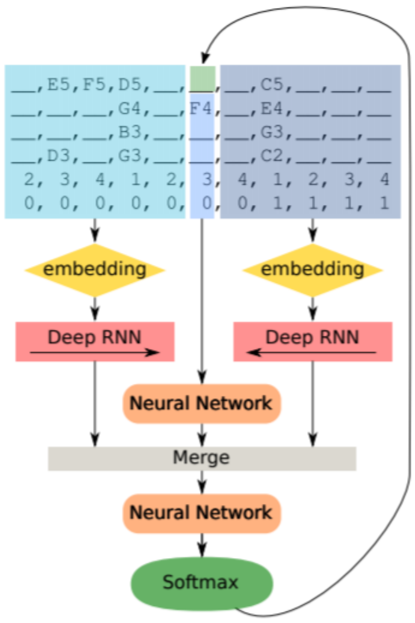

Figure 1: DeepBach's architecture

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. This network also allows for the use of embeddings.

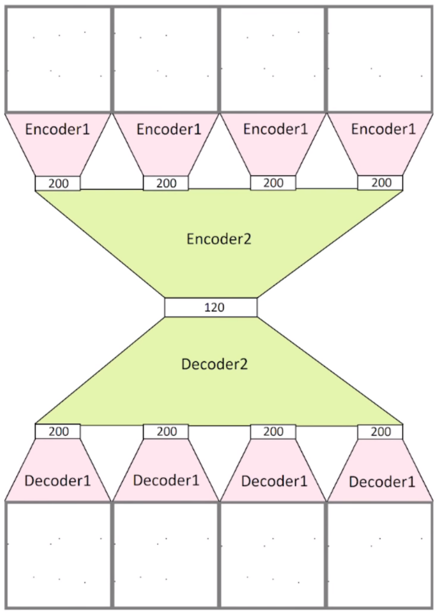

Figure 2: Network diagram

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.

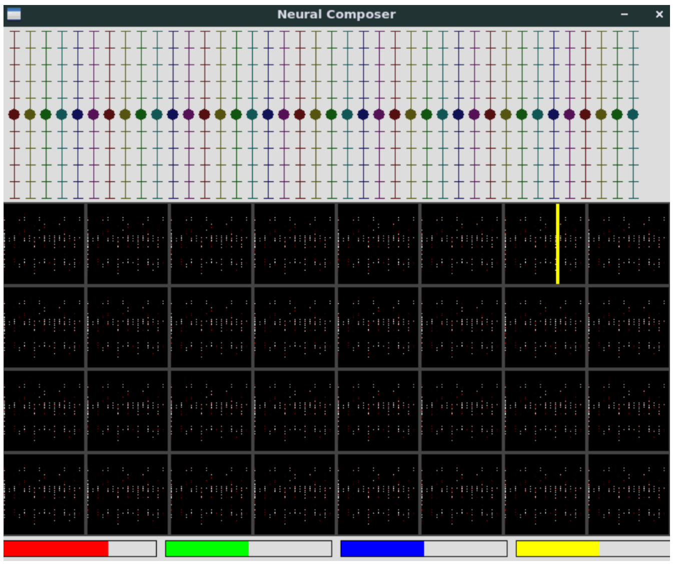

Figure 3: Composer interface

3. Methodology

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 our State of the Art, finding Polyphony RNN and Composer as possible models to solve our problem.

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. Setup and approach

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.

3.1.1. Model training

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:

With the results shown at table 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 and Metz (2019), we can consider that a learning rate, at most twice the minimum proposed (0.0015), may suffice.

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

Table 1: Training epochs for certain combinations of learning rate and batch momentum to obtain a given loss value (0.0015), lower is better

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%
Learning rate	0.0025	18.6667	19.6667	33.3333
	0.00275	21.3333	20.3333	30.3333
	0.003	23.3333	28	28.33333

Table 2: Mini batch experimental confirmation, for a fixed loss (0.0015) and batch momentum values (0.99), lower is better

From table 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.

3.1.2. 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 3, we can first look at how the model behaves. Aside from the hyperparameter optimizations, 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:

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.

3.1.3. Output generation

In order to generate our output, we can rely on our interface to extract the results. By varying the knobs, or

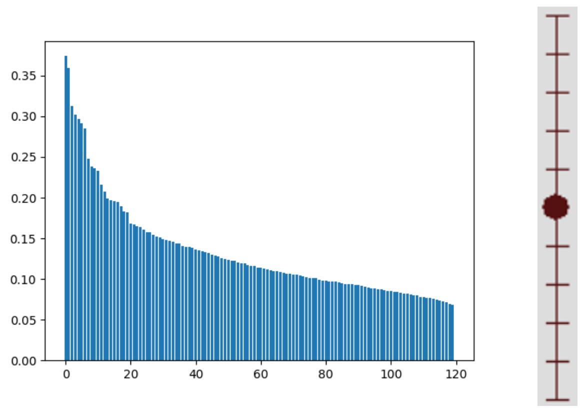

Figure 4: Weight distribution of the latent vector for the NES videogames dataset (left) Tuning knob for the interface (right)

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.

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.

3.1.4. 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 5.

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) may not exist, and when they do, they appear more frequently somewhere in the middle of the generated

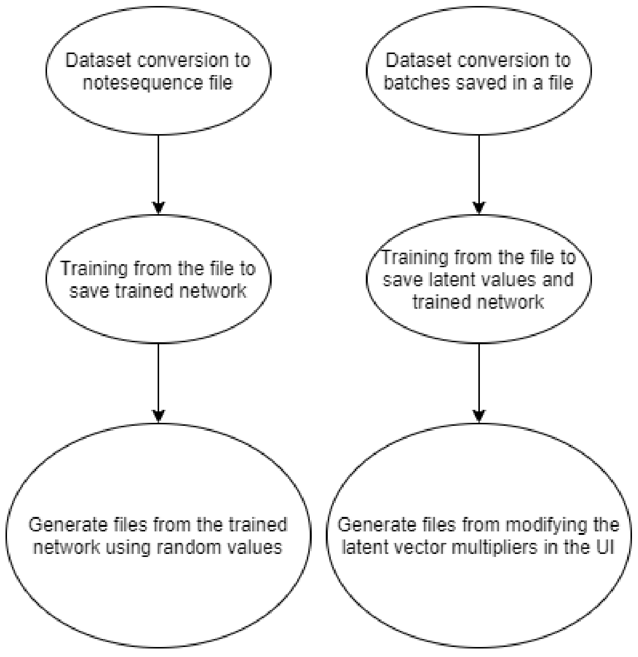
Dataset conversion to notesequence file
Training from the file to save trained network
Generate files from the trained network using random values
Dataset conversion to batches saved in a file
Training from the file to save latent values and trained network
Generate files from modifying the latent vector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

Figure 5: Magenta vs Composer logic

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

In this section we provide some insight over our obtained results to further extract conclusions.

4.1. Experimental observation

Aside from the presented output generation method, of randomizing parameters, some fine tuning with the knobs and smaller randomization intervals have been tried. 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:

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.

- Videogames - Slight bias towards track 3.
- Pop - Bias towards track 3.
- Mozart - Hard bias towards track 1.
- Queen - Bias towards track 2.
- Dragonforce - Hard bias towards track 3.

It is to be noted that most of the population of our experiment are students from Instituto Superior Técnico,

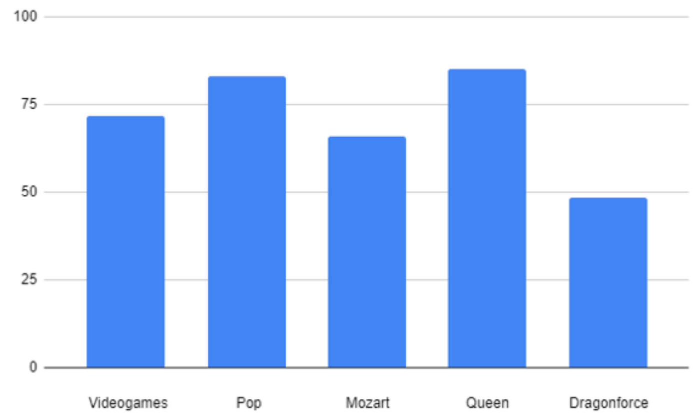

Figure 6: Relative familiarity of respondents with the different styles

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.

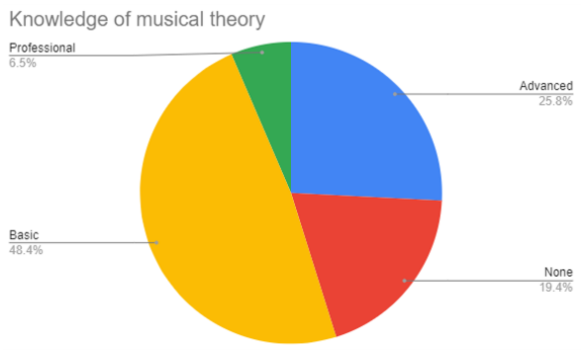

Figure 7: Level of musical theory knowledge of the respondents

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.

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.

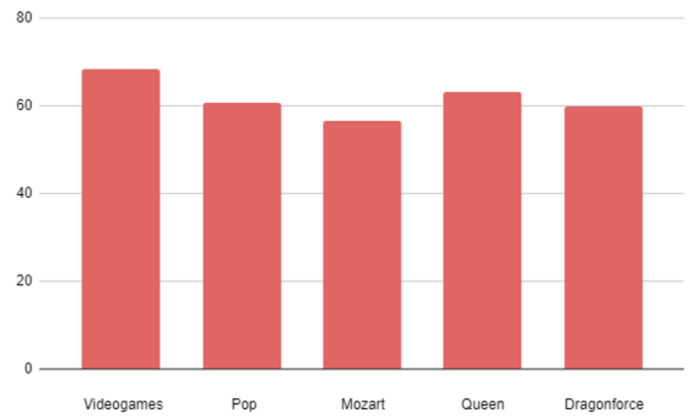

Figure 8: Most catchy main voice for each of the styles, according to respondents

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 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 work 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 with the baselines born with 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.

References

Braguinski, N. (2019). “428 millions of quadrilles for 5s. 6d.”: John clinton’s combinatorial music machine. https://doi.org/10.1525/ncm.2019.43.2.86.

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

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

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

HackerPoet (2018).

Hadjeres, G., Pachet, F., and Nielsen, F. (2017). DeepBach: a steerable model for Bach chorales generation. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1362–1371, International Convention Centre, Sydney, Australia. PMLR.

Shlomo Argamon, K. B. and (Eds.), S. D. (2010). The structure of style: Algorithmic approaches to understanding manner and meaning. pages 45–58, Berlin: Springer-Verlag.

Technologies, A. (2020).

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