Music Generation Using Generative Adversarial Networks

Diogo de Almeida Mousaco Pinho

diogo.pinho@tecnico.ulisboa.pt
Instituto Superior Técnico, Universidade de Lisboa, Portugal

Abstract—The idea of a machine being able to generate music is somehow intriguing. The music composition process implies the manipulation of baseline sounds or notation to create more complex structures. In this thesis a generation system based on raw waveforms representing musical bars is proposed, taking advantage of Machine Learning techniques. A preprocessing of the audio samples is performed, consisting on a transformation of the waveforms into a time-frequency representation, commonly used to deal with music signals. A state-of-the-art generative model was implemented with the purpose of creating music segments similar to those in the dataset, which is composed by 2 second long music bars. The original model is known Generative Adversarial Network (GAN) but the approached variant benefits from convolutional layers in its networks architecture and is called Deep Convolutional Generative Adversarial Network. Several approaches were made regarding different architectures and hyperparameters, in order to evaluate the model’s capability of meeting the proposed objectives. By means of an user study it is concluded that the music segments generated by the implemented system are not noise, and are actually musically pleasing.

Index Terms—Music Generation, Generative Adversarial Networks, Deep Learning, Deep Convolutional Generative Adversarial Networks, Machine Learning

I. INTRODUCTION

Music can be interpreted as the art of mixing sounds and silences in a way that produces beauty, is harmonic, and expresses emotions. Those responsible for composing music intend to spark some emotions on the listener. As a composer myself, I have dealt with the underlying struggles of expressing one’s ideas into melodies, harmonies or even rhythms. Having this in mind, one can wonder whether if it is possible to achieve the same purpose when the source is not a human being, but actually some algorithm.

In this thesis the task of generating a music segment waveform is addressed. A generation system based on raw waveforms is designed and implemented. The output is expected to represent music segments that should follow certain constraints imposed by the input data, such as being a 2 second long segment.

Algorithmic musical composition is a subject of work that dates back to 1959 [1]. However, the recent developments of deep neural networks, which have proven astonishing results in learning from big datasets, allowed this topic of music generation to be further developed. Over the past couple of years, tons of proposed models addressing music generation have been published, all of them on deep learning algorithms [2]–[10].

A big part of the neural network based models for music generation use Recurrent Neural Networks (RNN) and some variants, once the music generation process can be seen as creating dependent sequences [5]–[7]. RNNs are neural networks that possess a directed connection from the output of a unit in a certain layer with another unit’s input in another layer, closer to the input one, just as a closed loop system. As an example of these variants, Nayebi and Vitelli [6] present the Gated Recurrent Unit (GRU) and the Long Short-Term Memory (LSTM) architectures to address music composition. The used data in these experiments was preprocessed to be represented in a time-frequency domain.

Another way of representing data is directly from the time-domain audio waveforms, and it is used in WaveNet. Introduced by Oord, Dieleman, Zen, et al. [4], WaveNet is a fully probabilistic and autoregressive model, where all previous audio samples will condition the distribution of the next one. WaveNet is actually a Convolutional Neural Network (CNN), where dilated (or trous) causal convolutions are present in all layers. This kind of convolution consists on each filter taking every n-th element of the input matrix (n corresponds to a layer), rather than the whole elements. The WaveNet model reached state-of-the-art performance when applied to text-to-speech and foresees promising results when applied to music modeling, proving that CNNs are a valid option to generate music, alongside RNN.

Yang, Chou, and Yang [2] proposed a model based on a CNN, but trained with a Generative Adversarial Network (GAN) [11], called MIDINET. This model works over a piano-roll representation of Musical Instrument Digital Interface (MIDI) data, which is an encoding protocol of musical features such as sound and silence, notes, tempo, among others. MIDI is by far the most used method to compute music structures, once its complexity can be as simple as a binary code for whether a note is played on a certain time-step or not, disregarding the audio source variable. Back to the MIDINET model, the GAN uses an adversarial learning algorithm with two networks. One of them is called the generator which aims at converting a random noise sample into a realistic artificial data. The other is called discriminator and is just a classifier that tells whether the sample comes from real data or from the generator network. In this case, both of them contain convolutional layers on its architecture, such as CNNs.
do, which is a variant of the original GAN model, proposed by Radford, Metz, and Chintala [12] and known as Deep Convolutional GAN. This GAN alone generates bars and does not consider temporal dependencies between them, although this issue is dealt with a conditioner CNN.

According to the proposed work, the main contributions are the following:

- An approach to music generation based on raw audio data, through machine learning techniques.
- An implementation of a generative model with two deep neural network architectures.
- An approach to convolutional strategies to exploit feature locality.
- A statistical analysis of a user study to validate the results.

II. BACKGROUND

Sounds consisting of harmonic frequency components, when plotted against log-frequency, have this property that the distance between these frequency components is the same independently of the fundamental frequency [13]. Thus, the absolute position depends on the fundamental frequency, but the relative position of the harmonics are constant. As a result, when represented in the frequency domain, this components create a pattern which only depends on the instrumental source of the sound. This representation can be achieved by computing the Constant-Q Transform (CQT).

The time-frequency domain structures established by the CQT will be used as the input of the generative model.

A. Generative Adversarial Networks (GAN)

Generative Adversarial Networks (GAN) were developed by Goodfellow, Pouget-Abadie, Mirza, et al. [11] in 2014. They proposed the use of an adversarial process to estimate generative models by means of two neural networks training in parallel. One of these is called the generator network, which generates samples based on a vector sampled from latent space distribution, and the other is called the discriminator network, that learns to determine whether a sample comes from the training data or the generator. The training procedure for the generator network is to maximize the probability of the discriminator misclassifying the generations. Meanwhile, the discriminator network is trained to distinguish between real data and generated data. This process corresponds to a minimax two-player game. A visual inspection of figure 1 regarding the networks high level architecture may turn this clearer.

In order to provide a better understanding of this training idea, a comparison with a more practical problem is commonly made, namely the interaction between a counterfeiter and a bank. The bank classifies the money as real or counterfeit based on different features between them. However, the counterfeiter gets feedback on those classifications and, in order to be successful, he tries to mitigate the differences between real money and his counterfeits so they become as realistic as possible. If the counterfeiter is competent, he will eventually end up making indistinguishable money. This can be seen as a competition between the bank and the counterfeiter, just as the discriminator and the generator networks.

To train both networks, a loss function has to be defined. Intuitively, the discriminator loss evaluates how well it did at letting real samples go through (i.e. comparing $D(x)$ to 1) plus how often it was “fooled” by the generator (i.e. comparing $D(G(z))$ to 0), whilst the generator loss evaluates how much did it fail at making realistic samples (i.e. comparing $D(G(z))$to1). In a formal approach, this training method can actually be interpreted as a two-player minimax game between a discriminator, $D$, and a generator, $G$, with a value function $V(D, G)$, mathematically formulated by:

$$
\min_{G} \max_{D} V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}} \left[ \log D(x) \right] + \mathbb{E}_{z \sim p_{z}} \left[ \log(1 - D(G(z))) \right]
$$

(1)

where $p_{\text{data}}$ is the data distribution over real samples, and $p_{z}$ is the data distribution over noise input.

Training GANs was found to be a very challenging task. The main issue here is that the application of the gradient descent algorithm works well when the goal is to minimize a loss function, which is not exactly this case. The big deal in training GANs, is to find the Nash equilibrium of a non-convex game, which defines a state where neither players change their strategy regardless of opponent decisions. The utilization of the gradient descent to change the parameters of the discriminator may have a positive impact on the discriminator’s loss but a negative one on the generator’s loss, or vice versa. Hence, instead of converging, the solution oscillates.

The lack of performance measures is one big cause to the difficulty of using GANs. When the discriminator and generator networks training is not well balanced the GAN may enter in a failure mode known as vanishing gradient or mode collapse [14].

Vanishing gradient occurs when the loss drops to zero, ending up with no gradient updates. This problem arises when the discriminator is too good, resulting in a super slow learning process. On the other hand, when the discriminator behaves poorly, the generator does not have accurate feedback keeping it from representing reality.

Mode collapse is when the generator network generates a limited set of samples, or even a single sample, regardless of the random vector $z$. Once the discriminator network does not
actually force the diversity in the generator outputs, all these may converge to the same point that the discriminator network believes is realistic. This failure mode may be identified by inspecting the classification of the discriminator network or the generations of the generator network. If the generations keep the same regardless of the random vector, then the generator managed to "fool" the discriminator.

B. Deep Convolutional Generative Adversarial Network (DCGAN)

Radford, Metz, and Chintala [12] proposed a convolutional architecture of the GAN model that was proved to be more stable to train, and named it DCGAN. This network is a GAN where both the discriminator and the generator networks use convolutional layers, that can also be found in Convolutional Neural Networks (CNN) [15].

A convolutional layer consists on a set of weights, also known as filters or kernels. The filters, which are the learnable variables of this layer, are small spatial extents that have a depth equal to the one in the input data. As the name may suggest, this layer performs a mathematical operation called convolution, in this particular case a 2-dimensional convolution. This operation consists on the computation of dot products between the filter’s entries and the input, at every position. This results in a spatial connection between neurons.

There are four hyperparameters that directly affect the output of this layer. The first is the filter, which concerns the spatial extent of the filter. Another one is the depth, which corresponds to the number of filters learned. Then there is the stride, which defines the way the filter slides through the data. Finally, the zero-padding fills the spatial borders of the data with zeros.

The work of Radford, Metz, and Chintala lead to the adoption of some already proven modifications to the CNN standard architecture:

**Strided Convolutions:** The first modification regards the use of pooling layers do reduce spatial dimensions. Questioning the requirement of different layers in the pipeline, "The all convolutional net", by Springenberg, Dosovitskiy, Brox, et al. [16], proposes the drop of the pooling layer from the architecture, relying only on convolutions with non-unitary stride to do that job. This approach is broadly known as strided convolutions. The stride has a scaling property on the output dimensions of the convolutional layer. Hence, by changing the stride value, one can achieve the same downsampling as with a pooling layer. Furthermore, Springenberg, Dosovitskiy, Brox, et al. found that not only this proposal is valid without any loss in accuracy on recognition tasks, but actually gives state-of-the-art performance.

**Fractional-Strided Convolutions:** Regarding the generator network, the process requires an upsampling to create a shape with dimensions equal to the input ones, based on a single vector. This is called fractional-strided convolution, also known as transposed convolution [17]. This concept is very similar to the previous one, but the transposed convolution is computed instead.

**Eliminating Fully-Connected Layers:** The next modification adopted by Radford, Metz, and Chintala concerns the trend to eliminate fully-connected layers upon convolutional features. One convincing example of this is global average-pooling, used in some state-of-the-art image classification models [18], [19]. This alternative though has its drawbacks, namely hurting convergence speed. Still, a middle ground of connecting convolution layers with inputs or outputs of the network was proven to work well.

**Batch Normalization:** In order to accelerate training in deep neural network’s architectures, Ioffe and Szegedy [20] developed the batch normalization method. The change in the distribution of layer’s input parameters caused by the variations of the previous layer’s ones during training is defined as internal covariate shift. The batch normalization aims at reducing this issue. Through normalizing the mini-batch input of a layer to each unit having zero mean and unit variance, the batch normalization has been proven to be a useful tool to deal with training problems, for instance really high or really low activation’s output, helping the gradient flow.

III. Proposed Models

The main part of the algorithm is shared by all the models proposed in this work. The dynamics between the discriminator and generator networks has already been covered in the previous section, thus the focus now is their individual structure.

The process starts by taking 100 samples from a normal distribution $\mathcal{N}(0, 1)$, to create the random vector $z$. This vector will be fed to the generator network, and pass through a linear layer, consisting on linear operations, with the weight matrix randomly initialized from a normal distribution $\mathcal{N}(0, 0.2)$ and the bias initialized as 0. The output of this linear layer is reshaped into a 3-dimensional shape and batch-normalized, and the ReLU non-linearity is then applied. The following steps consist on convolutional layers, whose parameters will be specified later, once they differ according to each proposed model. Still, the input of each convolutional layer is batch-normalized and the output subject of a non-linearity, namely the ReLU, with the exception of the last one where the tanh non-linearity is used.

The discriminator network takes as input a 3-dimensional shape regarding real or generated data. The following steps consist on convolutional layers, whose parameters will be specified later, once they differ according to each proposed model. Still, after each convolutional layer a non-linearity is applied, namely the leakyReLU with a leak of 0.2, and with the exception of the first, all convolutional layers input are batch-normalized. After the convolutional procedures, the 3-dimensional output shape is flattened and subject of a linear layer, whose parameters are initialized as the linear layer used in the generator network, to produce a single output unit. To this last unit, a sigmoid non-linearity is applied.

The loss function used to measure the performance of each network was the cross-entropy. Note that the used of mini-batches is accounted by averaging the loss of the whole batch.

Regarding the convolutional procedures skipped above, 3 models have been considered. Every model’s dimensions are...
TABLE I
PROPOSED MODELS CONVOLUTIONAL LAYER’S SPECIFICATIONS.

<table>
<thead>
<tr>
<th>Model</th>
<th>Network</th>
<th>Layer</th>
<th>Input</th>
<th>Filter</th>
<th>Stride</th>
<th>#Filters</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>Generator</td>
<td>Conv 1</td>
<td>17 × 23 × 256</td>
<td>11 × 11 × 256</td>
<td>5 × 5</td>
<td>128</td>
<td>85 × 115 × 128</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Conv 2</td>
<td>85 × 115 × 128</td>
<td>9 × 7 × 128</td>
<td>4 × 3</td>
<td>2</td>
<td>340 × 345 × 2</td>
</tr>
<tr>
<td></td>
<td>Discriminator</td>
<td>Conv 1</td>
<td>340 × 345 × 2</td>
<td>9 × 7 × 2</td>
<td>4 × 3</td>
<td>128</td>
<td>85 × 115 × 128</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Conv 2</td>
<td>85 × 115 × 128</td>
<td>11 × 11 × 128</td>
<td>5 × 5</td>
<td>256</td>
<td>17 × 23 × 256</td>
</tr>
<tr>
<td>Model 2</td>
<td>Generator</td>
<td>Conv 1</td>
<td>1 × 23 × 512</td>
<td>1 × 11 × 512</td>
<td>1 × 5</td>
<td>256</td>
<td>1 × 115 × 256</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Conv 2</td>
<td>1 × 115 × 256</td>
<td>940 × 7 × 256</td>
<td>340 × 3</td>
<td>2</td>
<td>540 × 345 × 2</td>
</tr>
<tr>
<td></td>
<td>Discriminator</td>
<td>Conv 1</td>
<td>340 × 345 × 2</td>
<td>340 × 7 × 2</td>
<td>340 × 3</td>
<td>256</td>
<td>1 × 115 × 256</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Conv 2</td>
<td>1 × 115 × 256</td>
<td>1 × 11 × 256</td>
<td>1 × 5</td>
<td>512</td>
<td>1 × 23 × 512</td>
</tr>
<tr>
<td>Model 3</td>
<td>Generator</td>
<td>Conv 1</td>
<td>17 × 1 × 512</td>
<td>11 × 1 × 512</td>
<td>5 × 1</td>
<td>256</td>
<td>85 × 1 × 256</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Conv 2</td>
<td>85 × 1 × 256</td>
<td>9 × 345 × 256</td>
<td>4 × 345</td>
<td>2</td>
<td>340 × 345 × 2</td>
</tr>
<tr>
<td></td>
<td>Discriminator</td>
<td>Conv 1</td>
<td>340 × 345 × 2</td>
<td>9 × 345 × 2</td>
<td>4 × 345</td>
<td>256</td>
<td>85 × 1 × 256</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Conv 2</td>
<td>85 × 1 × 256</td>
<td>11 × 1 × 256</td>
<td>5 × 1</td>
<td>512</td>
<td>17 × 1 × 512</td>
</tr>
</tbody>
</table>

found in table I, their architecture is shown in figure 2, and its differences are described next.

Model 1 is very similar to the original DCGAN model. Due to the different dimensions of the data, the deepness of the networks had to be changed, having only 2 convolutional layers each. The dimensioning of the filter and the stride also have changed, but the filters are still relatively small spatial extents and the strides are also in the same order of magnitude.

Model 2 proposes striding along the width, which means that the receptive field will cover a temporal interval at a time but consider the whole action occurring in that interval. This ensures that the first convolutional layer is fully responsible for the analysis of the frequency component, i.e. which notes are being played. The following convolutional layers will, therefore, control the temporal relationship between each time interval.

Model 3 is kind of the opposite of Model 2. In this model, it is proposed striding along the height, which means that the receptive field will cover a frequency interval at a time but consider the whole time series. This ensures that the first convolutional layer is fully responsible for the analysis of the temporal component, i.e. in which time-steps a certain note is being played. The following convolutional layers will, therefore, control the relationship between different notes.

IV. IMPLEMENTATION AND RESULTS

One dataset will be used to train the proposed generative model. This was built over an improved melody created by a musician over a diatonic chord progression. The full melody was split into 100 segments where each represents, in musical notation, one bar.

The whole implemented system takes as input audio waveforms that represent data in time domain. This time domain signals are subject of a transform, namely the CQT, changing their representation into a time-frequency domain one. After that, the transformed signals are fed into a generative model. The generative model is trained to produce audio samples with their representation into a time-frequency domain one. After some validation tests were performed. Firstly, the CQT invertibility has to be guaranteed, so that the output of the whole system is, as the input, a waveform. Then, the developed generative model should perform decently when fed with image data, once this has already been achieved in other works. Finally, the generator network has to prove to be

![Diagram](image-url)
able to generate a certain sample when the training set only comprehends that specific sample.

These validation tests were completed successfully, proving the implemented system can be further explored. Thus, the considered parameters and its influence in the generative model’s performance will now be addressed.

**Number of filters:** The number of filters used in the convolutional layers of each of the 3 proposed models were set as stated in Table I. Different number of filters were tested following a power of 2 basis. The variation of that number was found to have impact on the resolution of the CQT plot, resulting on weaker audio representations when the number was set too low. It was concluded that after a certain number of filters the improvements of the resolution have stopped, hence being the ones adopted.

**Batch size:** The amount of samples evaluated by each network at a time was found to be critical to the generative model’s performance. As mentioned, the mode collapse is a very common training failure when addressing GANs. As approached in [14] and verified in this case, the use of batch discrimination helps to deal with this failure.

**Network updates per iteration:** The number of updates of each network per iteration was found to be crucial to balance the training of both networks. The lack of balance in training leads the loss of the discriminator network to drop to zero, which makes the gradients get too small ceasing the learning procedure (known as the vanishing gradient problem). Updating the generator network more than once per iteration was found to be a viable strategy to deal with this problem. The balance found between each network’s update per iteration that provides a well-behaved training, and consequently better results, was to update 1 time the discriminator network and 5 times the generator network, per iteration.

**Adam optimizer’s parameters:** The Adam optimizer parameters are the learning rate and the momentum. Their influence in the training concern stability and speed of convergence, which in this case means reaching the "oscillatory stage". The values used in original DCGAN model were found to be stable enough and tuning them did not significantly improved the networks performance. Therefore, the learning rate was kept at 0.0002 and the momentum term at 0.5.

After training each of the proposed models for the whole melody dataset for 10000 epochs, with the hyperparameters stated above, it was found that Model 1 was by far the one that produced the best results, once Model 2 and Model 3 were trapped by their large kernels. However, training Model 1 took almost 3 times longer to compute than the other models. The results of the trained models are presented in figure 4.
The networks training balance may be verified in figure 5 for the different proposed models trained with the melody dataset. One may verify in the generator network plot that the losses keep oscillating around the same value from epoch 2000 until the end, for all models. This implies that the discriminator network’s loss does not drop to zero, which might not be completely clear in the leftmost plot for Model 2 and Model 3 due to the lines superposition.

![Discriminator’s loss](image)

(a) Discriminator’s loss

![Generator’s loss](image)

(b) Generator’s loss

Fig. 5. Discriminator (a) and generator (b) losses regarding the trained generative model’s best performance. The red, blue and orange lines are relative to Models 1, 2 and 3, respectively. The plotted data is smoothed in order to provide an easier analysis.

It may be concluded that exploitation of different convolutional architectures, namely performing them in only one direction and with a significantly bigger filter size (Model 2 and Model 3), does not create value in the networks performance. Once the generations from those models are nothing like the input ones, the consequent transformation to audio samples will not be interesting.

A different set of parameters could perhaps have been set to train Model 2 and Model 3 to achieve other results. However, after long days of parameter tuning, the results provided by training Model 1 in these conditions were found to be undoubtedly the best.

A prior evaluation of the trained generative model was made to consider this as the best performance within the all the experiment. Still, the evaluation of the output of the whole system, which is an audio sample, is very subjective. This results were considered not to have enough significance when judged only by one individual. Therefore, a user study regarding the quality of the generated samples was found to be essential.

V. User Study

As a method to evaluate the results of the generative model implemented a user study was developed. This work proposes the exploration of music generation based on audio. The generative model proposed should be able to produce a musical audio sample with some constraints defined by the dataset.

This study should infer over the following hypothesis:

Hypothesis 1: The generated audio samples are not classified as a noise sound.

Hypothesis 2: The generated audio samples are ranked as musically pleasant.

The design and results of this experiment will be detailed next.

A. Participants

5 people (3 males, 2 females), aged between 22 to 25 years old, voluntarily participated in the study. All participants have at least a basic theoretical music knowledge, and none of them suffered from any hearing disorder. No participant had prior contact to the study.

B. Design

The study involved 10 samples from each of the following groups:

- X, the dataset.
- Y, the trained generative model.
- Z, the untrained generative model.

In order to mitigate practice effects, the study design followed a block randomization, i.e. each participant evaluated a block with the same samples but randomly sequenced. Two questions were formulated, and should be answered for each sample in the block. These are the following:

- Q1 — Do you classify the sound you heard as a noise sound or as a tone sound?
- Q2 — The sound you heard is musically pleasing.

C. Procedure

A brief overview of the procedures was given to participants before they were submitted to the experiment. They had to fill demographics information (gender, age, theoretical music level, hearing condition) first. Then, participants were asked to read some definitions in order to proceed to the following stage.

The experimenter played each sample in the randomly ordered block guaranteeing an interval of 5 seconds between each 2 second sample, so that participants could answer the 2 questions. After listening to the 30 samples block the experiment was concluded.
D. Results

Participants were asked to answer the questions Q1 and Q2. For each of the three sample groups X, Y and Z, 50 answers were evaluated.

As Q1 expects a binary answer (Tone/Noise), the relative frequencies regarding each sample group were computed and are plotted in figure V-D. One may conclude that all the samples from the X group were classified as tone, all the samples from the Z group were classified as noise, and 95% of the samples from the Y group were classified as tone.

![Fig. 6. Relative frequency per sample group regarding Q1.](image)

For Q2, the answer is a rank (Likert scale), therefore, the average ranks per sample group were computed and are plotted in figure V-D. One may conclude that in terms of musical pleasingly, in average, the samples from the group Z are not pleasant at all, the samples from the X group are very pleasant, and the samples from the Y group are relatively pleasant.

![Fig. 7. Boxplot of the average rank per sample group regarding Q2.](image)

The above discriminative statistics already shown evidence that both hypothesis are verified. However, the survey results were tested for statistical significant differences for the different sample groups by means of a Friedman test, and its correspondent post hoc.

The Friedman test regarding Q1 and Q2 proved the following:

- There was a statistically significant difference in the noise/tone classification (Q1) depending on which group the sample belongs to, \( \chi^2(2) = 94.360, p\text{-value} < 0.05 \).
- There was a statistically significant difference in how musically pleasing the sample was (Q2) depending on which group the sample belongs to, \( \chi^2(2) = 89.805, p\text{-value} < 0.05 \).

The Friedman test proved the existence of statistical significant difference, but does not prove anything else. In order to determine where these differences actually occur, it was necessary to perform post hoc tests. The appropriate are the Wilcoxon signed-rank tests on the different combinations of related groups although requiring a Bonferroni adjustment on the significance level. Hence, the following combinations will be compared: X-Y, X-Z and Y-Z. The post hoc test results regarding Q1 and Q2 proved the following:

- There was a statistically significant difference in the noise/tone classification (Q1) between the sample groups X-Z (\( Z(2) = -7.071, p\text{-value} < 0.017 \)) and Y-Z (\( Z(2) = -6.856, p\text{-value} < 0.017 \)), but not between the sample groups X-Y (\( Z(2) = -1.732, p\text{-value} \geq 0.017 \)).
- There was a statistically significant difference in how musically pleasing the sample was (Q2) between the sample groups X-Y (\( Z(2) = -4.118, p\text{-value} < 0.017 \)), X-Z (\( Z(2) = -6.372, p\text{-value} < 0.017 \)) and Y-Z (\( Z(2) = -6.199, p\text{-value} < 0.017 \)).

Based on the performed tests, one may confidently state that samples generated by the trained generative model (Y) are tone sounds. Moreover, the samples from the untrained generative model (Z) are less musically pleasing than the samples from the trained generative model (Y), and the samples from both Z and Y are less musically pleasing than the samples from the dataset (X).

VI. Conclusions

The user study conducted tested the hypothesis that the generated audio samples are not noise, i.e. controlled pitch characteristics were found, and are musically pleasing. This study compared the trained generative model’s output samples, Y, with samples from the dataset, X, and with samples from the generative model before training, Z, i.e. noise. It was concluded that the Y samples are classified as not noise, i.e. a tone sound. It was also concluded that the samples from the trained generative model group (Y) are more musically pleasing than the samples from the untrained generative model group (Z) but still not as musically pleasing as the samples from the dataset group (X).

The implemented system was expected to generate better music samples. However, once music is being represented in such a low level, even generating a sample that is not noise was a challenging task. Considering that and the results of the user study, it was found that the initial expectations were too high, and that the results achieved are actually interesting.

In future work, one may explore the application of larger datasets with different constraints, the inclusion of other layer to correlate the generated samples, or even the conditioning of the generations to match some prior sequence.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Prof. Rodrigo Ventura for guidance and support, and I would also like to thank Pedro Ferreira, Jose Corujeira, Lia Laporta and Raquel Miranda for spending their time discussing my work.
REFERENCES


