Music Generation from Textual Input

Report

Giulia Saresini, Vanessa Maeder

January 23, 2025

Abstract

This report explores the generation of music from textual input through deep learning techniques. A natural language processing (NLP) model, specifically Word2Vec, is utilized for textual analysis, while a Conditional Generative Adversarial Network (cGAN) is employed for music generation. The study uses the MAESTRO dataset, emphasizing key musical parameters such as pitch, key, and velocity. The findings reveal that music generation is a challenging task, requiring models to go beyond learning data distributions to effectively replicate the emotions and variations that music conveys to listeners in order to produce satisfactory results.

Contents

1 Introduction

This project explores the field of music generation, focusing on both the current state of the art and the development of a pipeline to create music from textual input. Recent advancements in deep learning have significantly improved the quality of algorithmically generated compositions. Earlier methods predominantly utilized recurrent neural networks RNNs and long short-term memory LSTM networks due to their ability to model sequential data effectively [7,8]. Transformer-based architectures, such as OpenAI's MuseNet [5] and Google's MusicLM [6], have set new benchmarks by producing compositions that are coherent, stylistically rich, and highly expressive. Similarly, Generative Adversarial Networks (GANs) [1] have emerged as powerful tools for music generation, enabling the synthesis of music conditioned on specific attributes such as style, genre, or emotion.

This project integrates natural language processing (NLP) techniques with conditional GANs (cGANs) to generate music. The NLP model encodes the textual input, while the cGAN synthesizes music conditioned on features like key and tempo, ensuring consistency with the stylistic and dynamic cues described in the input text.

2 Methodology

This section first addresses the dataset and the cGAN trained with it, followed by an analysis of the NLP model that converts textual input into the required format for the cGAN.

2.1 Description of the MAESTRO Dataset

The MAESTRO (MIDI and Audio Edited for Synchronous TRacks and Organization) dataset [4] is a comprehensive collection curated for the study and generation of classical piano music. It comprises over 200 hours of paired audio and MIDI recordings from piano performances in the International Piano-e-Competition. Each MIDI file includes detailed annotations, such as note pitch, velocity, duration, and timing, along with metadata like the composer and performance year. The dataset is publicly available and widely utilized for tasks in music generation and transcription.

2.2 Data Preprocessing

Before feature extraction, several alternatives were considered regarding the tracks to be used for training the neural network. Initially, it was proposed to use all tracks by extracting only a portion of each song. However, it was later determined that this approach could hinder the network's ability to learn patterns within the compositions. Therefore, the decision was made to use the entire compositions.

Subsequently, it was assessed whether to use all available tracks or only those within a specific group, particularly considering the duration of the tracks. *Figure 1* shows the distribution of tracks based on their duration.

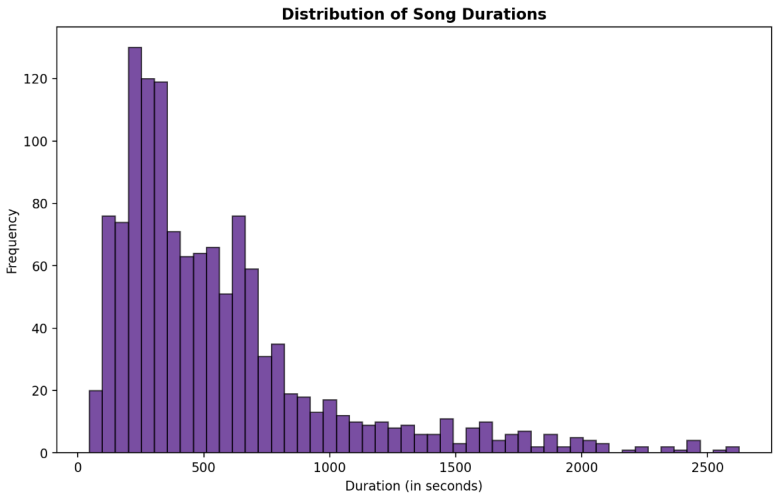

Figure 1: Distribution of tracks based on duration.

The decision was made to select compositions with a duration between 180 and 300 seconds.

After selecting the tracks of interest, the feature extraction process was carried out to define the inputs for training the cGAN network. For each track, the following features were extracted:

- **Pitch**: The pitch value of each note, representing its frequency and corresponding musical tone.
- **Start**: The starting time of each note in the composition.
- **End**: The ending time of each note in the composition.
- **Velocity**: The velocity of each note, which indicates the intensity or volume of the note when played.
- **Key**: A binary variable indicating whether the composition is in a major or minor key.

Before using these features to train the cGAN network, it was necessary to normalize them and make them all of equal length. Specifically, pitch, start times, end times, and velocity are lists whose lengths depend on the track being considered. However, the neural network requires inputs of uniform length.

To address the issue of sequence length, padding was applied to make all sequences equal to the maximum sequence length (5011). At this stage, compositions with a number of notes considered outliers in the note distribution were excluded. Specifically, two compositions with more than 6000 notes were removed. Below is the distribution of the number of notes for the tracks under consideration (*Figure 2*).

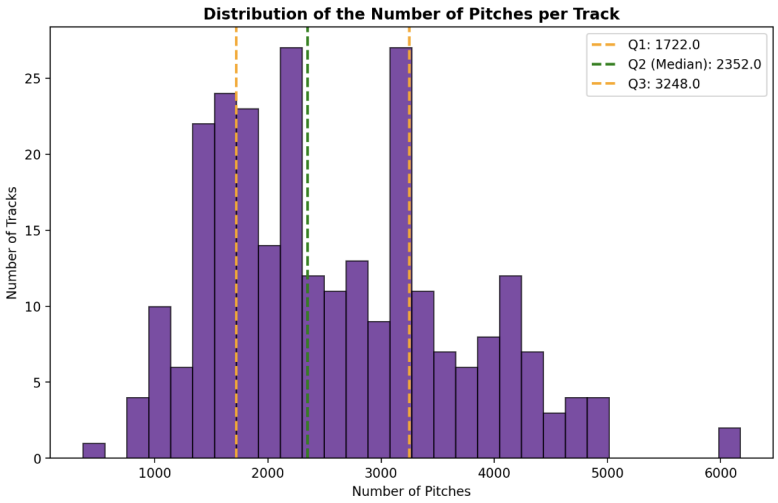

Figure 2: Distribution of number of notes in each track.

After padding, the variables were normalized. Specifically: Pitch and Velocities were normalized by dividing by 127, as this is the maximum value they can take within their domain. Meanwhile, Starts and Ends were normalized by dividing by 300, since the maximum duration in seconds considered is 300.

Although a cGAN network was tested to generate the starts and ends of each note, the network presented in the next section focuses exclusively on the generation of pitches and velocities conditioned on the key. This decision was made because the generation of starts and ends failed to produce satisfactory results, as all the notes were generated within roughly the same time range.

2.3 Music Generation Model - cGAN

The conditional Generative Adversarial Network (cGAN) [3] is implemented for music generation. The model generates musical sequences conditioned on the categorical feature key, which indicates whether the composition is written in *Major* or *Minor* key. The network architecture consists of a generator and a discriminator, each designed to handle structured and conditional data.

Generator The generator takes a random noise vector of dimension 5011 and a one-hot encoded vector representing the condition key. These inputs are concate-

nated and processed through several layers. First, a dense layer with 512 units and ReLU activation is applied, followed by batch normalization. Next, two LSTM layers are used for sequential modeling, with dropout included to prevent overfitting. Finally, a dense layer with 1024 units and ReLU activation is applied, followed by batch normalization. This final layer produces two outputs: pitch and velocity, each with sigmoid activation to constrain their values.

Discriminator The discriminator receives concatenated real or generated sequences (pitches and velocities) along with the conditional information. It is composed of a minibatch discrimination layer to improve gradient flow and mitigate mode collapse, followed by dense layers with ReLU activations and dropout for regularization. The final layer is a dense layer with sigmoid activation, which is used to classify real and fake sequences.

Wasserstein Loss with Gradient Penalty To stabilize training, the Wasserstein loss is used, defined as:

$$\mathcal{L}_D = \mathbb{E}_{x \sim p_{\text{real}}}[D(x)] - \mathbb{E}_{\tilde{x} \sim p_{\text{gen}}}[D(\tilde{x})]$$

where $D(x)$ represents the discriminator output. To enforce the Lipschitz constraint, a gradient penalty term is added:

$$\mathcal{L}_{\text{GP}} = \lambda \mathbb{E}_{\hat{x} \sim p_{\text{interp}}} \left[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2 \right]$$

Generator Loss The generator's loss aims to generate data that the discriminator classifies as real. Additional penalties are introduced to improve musical quality:

- **Repetition Penalty:** Reduces abrupt pitch changes and prevents excessively short durations.

$$\mathcal{L}_{\text{repetition}} = \text{mean}(|p_{i+1} - p_i|) + \text{mean}(\mathbb{I}(|d_{i+1} - d_i| < \delta))$$

- **Pitch Range Penalty:** Penalizes sequences with a pitch range that is too narrow.

$$\mathcal{L}_{\text{range}} = |\max(\mathbf{p}) - \min(\mathbf{p})|$$

Optimization The models are optimized using Adam optimizers with the following learning rate schedules:

- **Generator:** Initial learning rate 0.0002, with exponential decay.
- **Discriminator:** Learning rate 0.0001 with a fixed schedule.

Early stopping is applied to prevent overfitting, and the best model is selected based on the generator loss.

2.4 Natural Language Processing Model

For the natural language processing (NLP), a Word2Vec model was chosen, which can be used to transform the text input into the input values for the cGAN model. This was accomplished by selecting *"major"* and *"minor"* for the key input, and *"slow"*, *"moderate"*, and *"fast"* for the velocity input of the cGAN model as reference vectors. The input for music generation was then determined by transforming the text into a vector and selecting the input based on which reference vector the text vector is closest to.

To ensure that the Word2Vec model has a deep understanding of semantics, the pretrained `"en_core_web_lg"` model from SpaCy [2] was used. This model includes over 343,000 unique vectors, enabling it to process a wide range of inputs.

3 Results and Analysis

The reported results, particularly those regarding the cGAN neural network for music generation, although analyzed separately, were obtained using textual input generated by the Word2Vec model. It was deemed optimal to divide the results into two distinct sections for the two models in order to analyze in detail what worked and what did not work in both cases.

3.1 Word2Vec

As the Word2Vec model was not trained by the authors of this report and no dataset fulfilling the requirements was available for thorough testing, only observations made while using the model can be relied upon. The first observation is that, although the model could sort emotional words into the key categories one would typically associate them with without issues, it showed problems when identifying the velocity. The model would always choose between the velocities *"slow"* and *"fast"*, while the velocity *"moderate"* was only selected if explicitly stated in the text. This suggests that, although one might assume the *"moderate"* vector would lie between the *"slow"* and *"fast"* vectors, this is not the case. One solution could be to let the velocity for the input *"moderate"* be represented by the vector exactly between the *"slow"* and *"fast"* vectors.

In addition to modifying the vectors used to transform the input, training a Word2Vec model specifically for music descriptions would be a way to achieve better results. Particularly if the vectors could be weighted in such a way that frequently occurring words that do not add value to the music description, like the word *"and"*, for example, have only a small influence on the result. Currently, an excessive use of such words can alter the outcome.

Last but not least, the Word2Vec model views every word independently, meaning that a text like *"not slow"* would still return the velocity *"slow"*. Thus, chang-

ing to a model capable of understanding relationships between words or adding such a model to adjust the input so that the Word2Vec model can process the text more accurately would be a good enhancement.

3.2 cGAN

The neural network was trained, achieving the following as the best epoch:

- **Best Epoch:** 493
- **Best Discriminator Loss:** 0.8059
- **Best Generator Loss:** 0.4247
- **Best Discriminator Accuracy:** 0.5156

Since this is a generative network, standard metrics such as accuracy cannot be relied upon to evaluate the training quality and results. Instead, the quality of the generated data needs to be assessed.

As the generated data consists of music, one way to verify the quality of the output is by listening to the MIDI files produced by the network. The resulting melodies are pleasant but highly repetitive, and they cannot be classified as music in the conventional sense. This is partly because the majority of the pitches generated always fall within a very small range and with highly repeated pitches, as shown in *Figure: 3*.

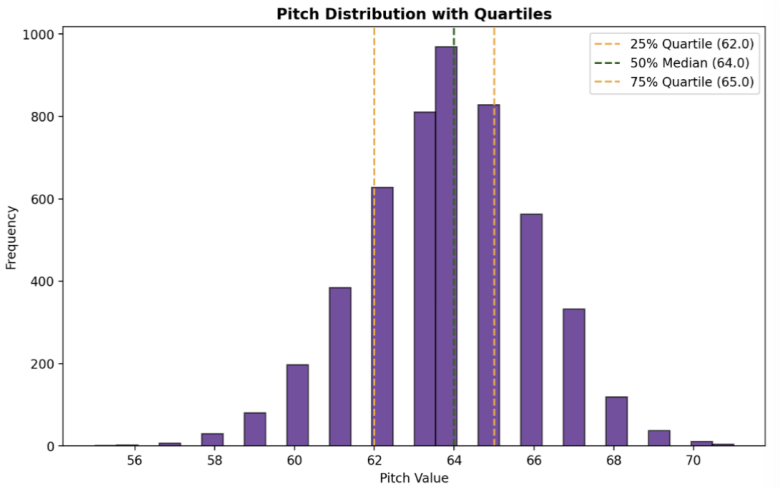

Figure 3: Distribution of the Generated Pitches.

This type of distribution makes sense when the original distribution of the pitches within the dataset is examined, as the pitches of the compositions considered are indeed distributed normally, with a mean around 66, as shown in *Figure 4*.

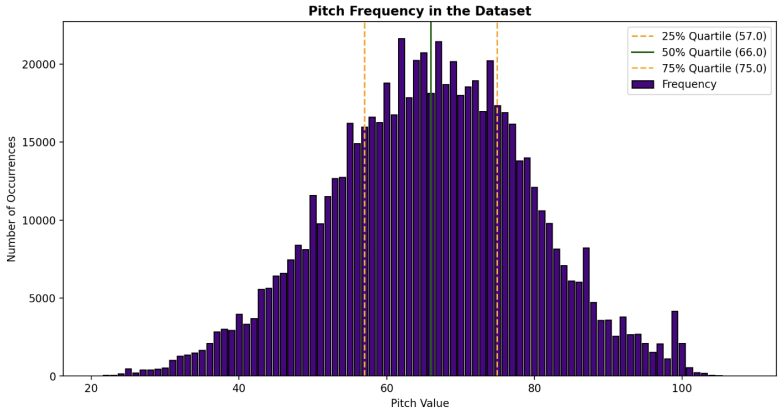

Figure 4: Distribution of all pitches in the dataset.

Another important aspect to consider when analyzing music is the various pitches within a composition are connected to each other. To investigate how the network creates relationships between pitches, a comparison was made between a signal from the generated music (specifically the first 500 pitches) and the signal (also 500 pitches) of a real piece from the dataset. *Figure 5* shows the signal for music generated by the network.

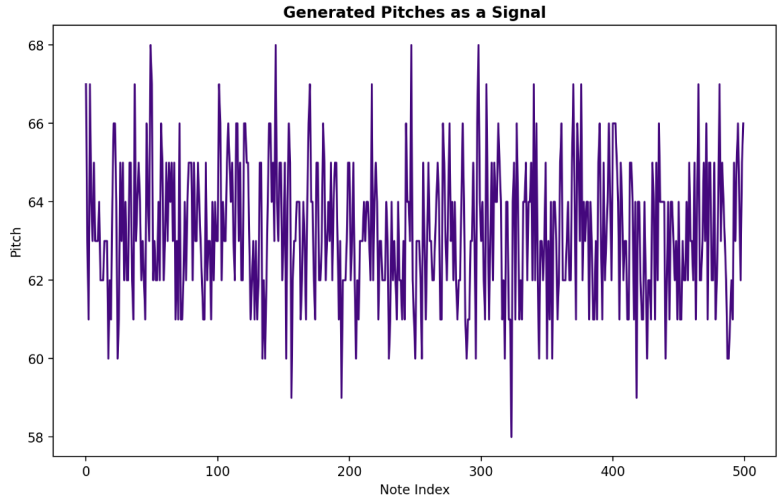

Figure 5: Pitch Signal of the Generated Music.

As can be observed, no clear pattern is present within the signal, and it behaves almost like white noise without any visible trend. What is expected in a signal that should represent music is the presence of trends or even significant jumps between pitches, indicating that something meaningful is being expressed within the signal.

Below, the signal of a real piece from the MAESTRO dataset will be shown (*Figure 6*).

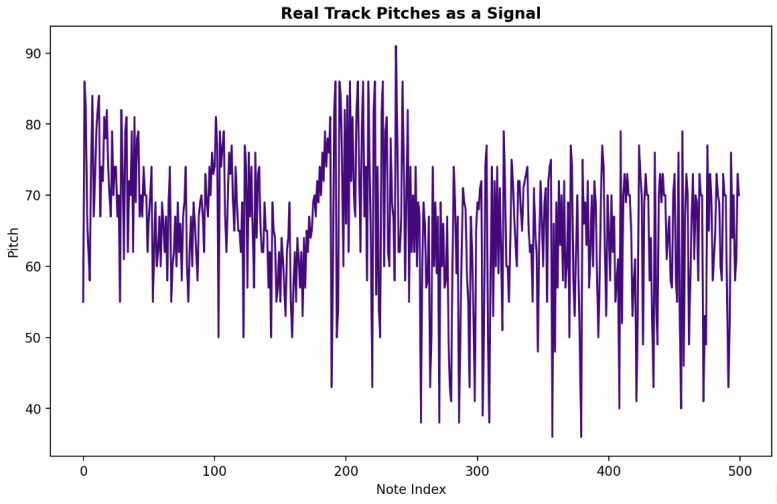

Figure 6: Pitch Signal of a Real Music Composition.

It is evident that, in this case, within the first 500 notes, various patterns and musical scales can already be observed, which characterize the music of the compositions we know.

Given these considerations, it is likely clear that the key conditioning introduced during the training of the neural network did not work as intended. Experiments were conducted by generating 100 tracks conditioned by the key multiple times and verifying the detected key within these tracks. However, the results consistently show a uniform distribution with approximately 50 Minor keys and 50 Major keys each time.

Therefore, while the network has indeed learned something from the data, it has focused more on the most frequent observations and has not been able to learn how to create musical patterns and replicate them (such as musical scales).

4 Conclusion

In this work, the use of a conditional Generative Adversarial Network (cGAN) for generating classical piano music, conditioned on key and velocity parameters derived from textual descriptions, was explored. Through the use of the MAESTRO dataset, a system for generating music sequences based on features such as pitch, velocity, and key, extracted from MIDI files, was developed and tested.

The results obtained with the cGAN network were promising in certain aspects but showed significant limitations. While the generated melodies were pleasant in some instances, they tended to be highly repetitive and lacked the variety typically seen in real musical compositions. This can be attributed to the network's tendency to produce notes within a small pitch range and its difficulty in creating

meaningful relationships between notes. The generated signals exhibited a lack of structure and patterns when compared to real music compositions, which can be considered an area for improvement.

Moreover, the key conditioning, which was meant to guide the generation of melodies according to specific musical keys, did not function as expected. Despite attempts to condition the network on different keys, the results showed a uniform distribution of Major and Minor keys, indicating that the network failed to properly learn how to produce music in the intended keys. This suggests that the network focused more on the most common features in the dataset and struggled to understand more complex musical structures.

In addition, the Word2Vec model used for transforming text descriptions into input features for the cGAN network also presented challenges. While it was successful in sorting emotional words into appropriate categories, it struggled with accurately identifying velocities, often defaulting to extremes. Future improvements could involve training a Word2Vec model specifically for music-related text, as well as incorporating a model that understands relationships between words to enhance the semantic understanding of the input.

Overall, this research demonstrates the potential of using neural networks for music generation, but it also highlights the need for further refinement, particularly in terms of conditioning and understanding musical structures. Future work will focus on enhancing the conditioning mechanisms and addressing the model's limitations in generating coherent and varied musical compositions.

References

[1] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. Gansynth: Adversarial neural audio synthesis, 2019.

[2] Explosion. spacy/en_core_web_lg.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1411.1784*, 2014.

[4] Curtis Hawthorne, Andrew Stasyuk, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, and Douglas Eck. Maestro: A dataset for modeling musical performance. *arXiv preprint arXiv:1810.12247*, 2019. Available at https://g.co/magenta/maestro.

[5] OpenAI. Musenet: Openai's generative model for music. `https://openai.com/index/musenet/`, 2019. Accessed: 2024-12-03.

[6] Google Research. Musiclm: Generating music from text. `https://musiclm.com/`, 2023. Accessed: 2024-12-03.

[7] Paperspace Team. Music generation with lstms. `https://blog.paperspace.com/music-generation-with-lstms/`, 2020. Accessed: 2024-12-03.

[8] TensorFlow Team. Generate music with tensorflow. `https://www.tensorflow.org/tutorials/audio/music_generation?hl=it`, 2023. Accessed: 2024-12-03.