Harmonizing with machine learning: using AI as a tool to generate classical music

Joshua Min

Abstract:

Generating music using machine learning involves training algorithms on large datasets of music to learn the underlying patterns and structures characteristic of different genres or styles. We approached this task by initially collecting and preprocessing a diverse set of musical pieces, which include melodies, harmonies, rhythms, and timbres. Subsequently, the machine learning model, in this case deep neural network, was trained on this data. The trained model learned to predict or generate new musical sequences that are stylistically similar to the input. By adjusting parameters and refining the model, it can enhance its ability to create music that is both novel and aesthetically pleasing. In this work, the application of various network architectures such as recurrent neural networks (RNNs) was explored for this purpose. These results demonstrate a proof of principle that machine learning can become a relevant tool in the creative process, providing new avenues for musical expression and experimentation.

Introduction:

This research project aims to answer the question, “Can machine learning models generate music in the style of different classical music composers?”. For the past hundreds of years, sheet music, which includes the use of notations such as notes, chords, and time signatures, has been the traditional way for musicians to read music. However, current machine learning models have used machine learning techniques to not only read and analyze sheet music but also create their own. This has opened up opportunities for people to use machine learning models as a way to replicate the style of music of composers, using factors such as how often composers “use” a certain note or key.

Machine learning is the process of training a model based on a processed training dataset and using algorithms, allowing models to “learn” and improve. Machine learning models can find patterns or make decisions, and are differentiated by supervised and unsupervised learning. Supervised learning means training a model based on labeled data, and is often used in classification and regression, while unsupervised learning means training a model based on unlabelled data, and is often used for clustering. There are also many algorithms for machine learning, such as linear regression, deep learning models, and neural networks.

Neural networks are a method in machine learning that processes data in a way similar to how the brain does. In neural networks, there are layers of neurons, with neurons between each layer being connected. The strength of these connections between these neurons is called weights, and these weights are used to determine how important a neuron is and how much influence that input data will have on the product that is outputted. Biases help the model be more accurate by making translations for activation functions. The Universal Approximation Theorem states that if a neural network has one hidden layer, it can, regardless of the complexity of the function being approximated, represent it accurately if provided with enough neurons within that hidden layer.

In the present study, we use machine learning to generate music and identify a model that would be suitable for background music in a public space. The transformer model GPT-3 and the LSTM model Performance_RNN were used for music generation. Using the COSIATEC algorithm and a custom tonality metric, the recurrent patterns and tonal accuracy were evaluated. Both models effectively generated long-term musical structure, though training time and accuracy varied with different data sets.

The model's performance was measured on the centricity metric. The centricity metric, which evaluates whether certain notes are more frequent than others, could indicate the quality of music. A lower centricity value means that some notes are used more often, suggesting that the model isn't selecting notes randomly but has learned the common harmonic patterns in music.

Learning rate	**Consonance**	**Macroharmony**	**Centricity**	**Note Density**
0.00001	0.7759 ± 0.0671	11.99 ± 0.0995	3.3349 ± 0.1484	1.0429 ± 0.2909
0.0001	0.8361 ± 0.0864	11.78 ± 0.6258	3.1174 ± 0.2405	1.6938 ± 0.3352
0.001	0.9057 ± 0.0678	11.0 ± 1.1576	2.7623 ± 0.3059	1.8005 ± 0.394
0.01	0.6378 ± 0.0183	12.0 ± 0.0	3.5299 ± 0.0193	1.18 ± 0.1241
Learning rate	**Groove**	**Empty beat ratio**	**Compression ratio**	**TECs**
0.00001	0.9764 ± 0.0033	0.0005 ± 0.0028	1.3214 ± 0.0931	28.01 ± 4.68
0.0001	0.9641 ± 0.0065	0.0182 ± 0.0387	1.4561 ± 0.0452	37.35 ± 6.80
0.001	0.9646 ± 0.0067	0.0041 ± 0.0142	1.5934 ± 0.0991	33.47 ± 7.49
0.01	0.9771 ± 0.0023	0.0002 ± 0.0016	1.3591 ± 0.0280	31.55 ± 3.1571

Figure 1: Recreated from Rickard, E. (2022). *Table 4.3: The metric evaluations on music generated by Performance RNN for different learning rates.* Generating Music using AI. Retrieved from https://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=9093922&fileOId=9093927.

Dataset Review:

Source	Citation (APA)	Data Type
#1: This music dataset was created by Yash Bhaskar on Kaggle. It contains mp3 files of classical, electronic, pop, and rock music.	Bhaskar, Y. (2024, January 31). *Emotify - Emotion Classificaiton in songs*. Kaggle. https://www.kaggle.com/datasets/yash9439/emotify-emotion-classificaiton-in-songs	mp3
#2: This music dataset was created by Bohdan Fedorock on Kaggle. It contains midi files of works of classical composers such as Sibelius, Tchaikovsky, and Wagner.	Fedorak, B. (2019, January 28). *Midi_classic_music*. Kaggle. https://www.kaggle.com/datasets/blanderbuss/midi-classic-music	midi
#3: This music dataset was created by Kritanjali Jain on Kaggle. It contains midi files of works of famous pop artists such as Shawn Mendes.	Jain, K. (2021, April 26). *Music_midi_dataset*. Kaggle. https://www.kaggle.com/datasets/kritanjalijain/music-midi-dataset	midi
#4: This music dataset was created by Chetan MJ on Kaggle. It contains midi files of famous pop songs such as “Best Day of My Life” and “Shake It Off.”	MJ, C. (2018, August 19). *Pop music collection*. Kaggle. https://www.kaggle.com/datasets/chetanmj23/pop-music-collection	midi

#5: This music dataset was created by Soumik Rakshit on Kaggle. It contains midi files of classical works by composers such as Bach, Brahms, Chopin, and Mendelssohn.	Rakshit, S. (2019, May 17). *Classical music midi*. Kaggle. https://www.kaggle.com/datasets/soumikrakshit/classical-music-midi	midi

Methodology:

To start, Python libraries such as music21, Matplotlib, NumPy, pandas, Tensorflow, Matplotlib, scikit-learn, seaborn, and IPython were imported. The dataset for this project was compiled from the music datasets found on Kaggle, consisting of 48 different MIDI files of Chopin's music. In the Jupyter Notebook, this dataset was parsed as a music21 stream. The corpus is created using a function that extracts the chords and notes of each MIDI file. From there, the most frequently played notes and the least frequently played notes are identified, and the least frequently played notes are removed from the Corpus. Throughout the dataset, these notes that were played least frequently were so rare that some only appeared once or a couple of times throughout the whole dataset. It can be assumed that since Chopin did not include these notes as much,and that they either have less of significance within Chopin's music or Chopin did not enjoy playing these notes as much. Additionally, taking out these rare notes can improve the model's efficiency and accuracy as there is less computational complexity, training times, and errors.

Then, a dictionary is created to translate musical notes into numerical representations, so that the computer can efficiently work with these notes. The dictionary can be used to encode and decode information using the long short-term memory (LSTM), a variant of the Recurrent Neural Networks (RNN). Recurrent Neural Networks process sequential data, holding past information about it. The LSTMs are able to hold sequential data and past information over long sequences making them the perfect candidate to store the music, which includes elements of both time and series. The musical corpus is then encoded and broken down into smaller, unique sequences. These sequences are part of a segment of the music. The features and corresponding targets are indexed in the dictionary. After labels and targets are adjusted, datasets are split into two different kinds: the train and seed datasets. The seed dataset is used as an initial input for generating sequences for the machine learning model, acting as a "starting point." The machine learning model uses the test dataset to see and analyze the model's performance and learn how well it works.

Additionally, the model also defined durations for the corresponding notes and predicted durations, similar to the process of predicting the notes. This helped make the notes in the songs be able to play in varied durations, such as quarter notes, eighth notes, and half notes, allowing for potential in creating the stylistic, Romantic tunes of Chopin.

The model has two LSTM blocks with 512 and 256 outputs, respectively. Dropout was added to help prevent overfitting to the training data and allowed the network to generate more unique pieces of music. The model was finally trained with the training dataset for over 200 epochs, meaning that the machine learning model was iteratively trained over the entire training dataset 200 times.

Finally, the model was evaluated with training loss plotted with respect to the epochs. While ideally we would be using testing loss or validation loss, we ended up plotting this relationship with the test data rather than the training data instead. After generating the melody, and organizing the music back into the notes and chords list, the final MIDI file by model is created.

Additional information regarding the materials and methods utilized by this study can be found at the following resources:

Dataset - Google Drive Folder (Folder of Chopin MIDI Files):
→ **chopin**

Python Jupyter Notebook - .ipynb file (Machine Learning Model Program Code):
→ **MusicMix.ipynb**

ML-generated music using Chopin dataset - .mid file
→ **ChopinMix.mid**

Full Github Repository
→ **Github Repository**

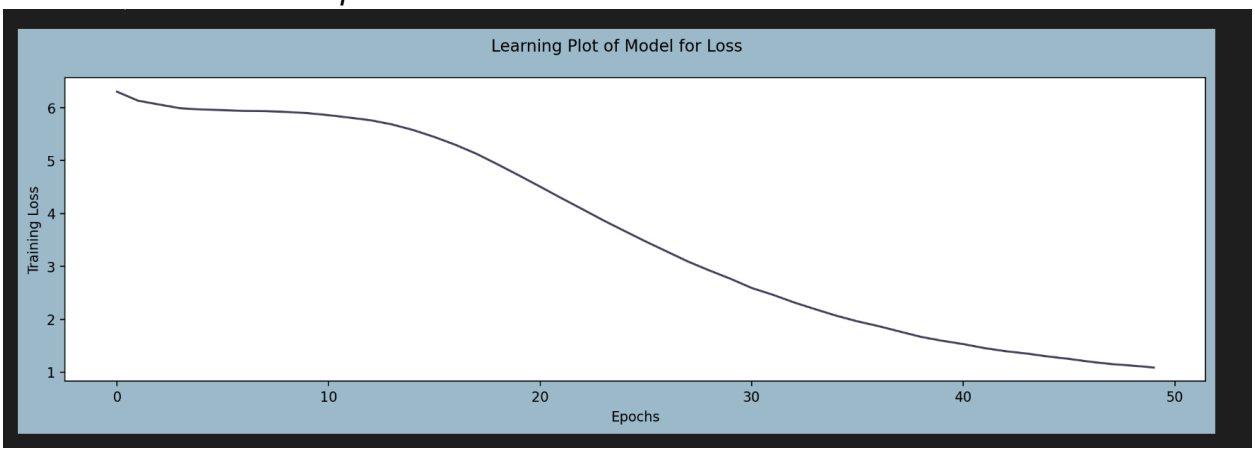
Results:

The final music file was produced after 200 epochs, as a midi file, which was around 1 minute long. The piece was in C Major, at a tempo of 120 BPM, in common time. Overall, the generated song by the model was arguably subpar in many aspects compared to many of Chopin's original works, mostly owed to the song's rather one-dimensional styled melody, and lack of lyrical nuance and sophistication that adds complexity such as chord progressions, scales, and arpeggios. This is the result of the machine learning model only selectively picking the notes to use based on how often they were used in Chopin's music, which resulted in a composition that superficially resembles Chopin's style but lacks the depth and emotional intricacy that characterizes his true masterpieces. The chords and notes plated often did not match in terms of harmony as well, which occasionally resulted in somewhat bizarre sounding moments. Moreover, variation in tempo, such as rubato, as well as dynamics in Chopin's pieces, cannot be easily captured by an algorithm that adheres strictly to predefined patterns.

Despite some of these parts of the model's shortcomings, the model performed quite well in terms of its overall structure as a song. There were no instances where the song failed in which the song failed to maintain a coherent form, and the transitions between sections were smooth and logical, with the same tempo being maintained at all times. Additionally, the music had its own charm in its simplicity and was overall very enjoyable to listen to. The aspect of

adding varied durations as necessary throughout the song also contributed to an engaging listening experience that can make the song more distinctive and exciting.

Figure 2 *shows the training loss plotted against epochs, to measure the performance of the model. The model decreases gradually over the first 20 epochs before decreasing at a steady rate until it reaches 50 epochs.*



Training loss is the measure of the error between the predicted output and the true output, with the loss curve in Figure 1. The training loss indicates the performance of the machine learning model, with a lower training loss signifying a better performance of the model, and a higher output accuracy.

In the context of generating the music, as this loss is being applied to both the notes and durations of those notes, it can show how well the model is able to pick up the various nuances and patterns from the data provided. Figure 2 demonstrates how the machine learning model is, for the most part, improving over every iteration, with only some instances of plateaus that may result from overfitting, which is when the model performs poorly on new data it is provided due to becoming too familiar and tailored with the training data. This shows how 50 epochs will generally be an optimal length to train a model to generate music with likeliness for sophistication.

Figure 3 *shows a graph of the frequency distribution of notes, indicating how frequently notes were played, with respect to the number of chords.*

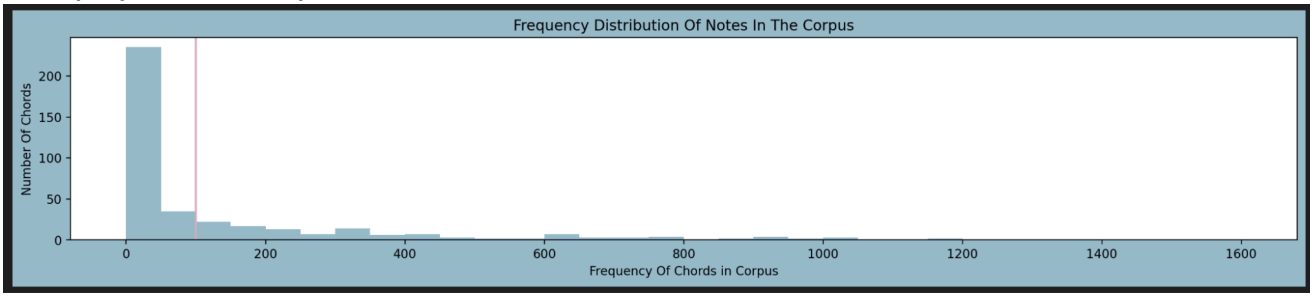


Figure 4 *demonstrates the statistics of occurrences of notes in the corpus, with the average recurrence of all notes, most frequent note, and least frequent note in the corpus being shown.*

```
Average recurrence for a note in Corpus: 145.8287153652393
Most frequent note in Corpus appeared: 1627 times
Least frequent note in Corpus appeared: 1 time
```

By identifying which notes appear more often than others, it gives an important clue for the machine learning model to use: how rare which notes are. By knowing how rare these specific notes are, it is possible to reduce problems and errors that can be encountered by removing some of these notes that occur the least. This removal of rare notes was done under the assumption that composers of song would have used the notes more often if they liked it, therefore, attempting to mirror the style that the composers would have written the song. Figure 4 demonstrates the significant contrast between the most frequently played (1627 times) and least frequently played notes (1 time), with the average recurrence for a note being around 146, demonstrating that there is a big enough statistical difference between the frequencies to be able to specifically exclude some of these rare notes. In this model, we removed all notes that were played less than 100 times, indicated for the red divider in Figure 3.

Discussion/Conclusion:

In the future, we look forward to experimenting with other models such as Transformer models general adversarial networks (GAN) that have the potential to generate music due to their ability to sequentially store data in their memory. By testing these various architectures and adjusting hyperparameters, we can try to find another unique style of music that can be generated, possibly with better performance than the RNN model. We also intend to work towards building a model that can generate music more than just a melody, such as an accompaniment or beats, and be able to use other instruments aside from just purely piano, so that a wider variety of songs that use various, unique instruments can be used to train the model, and can more accurately reproduce the music of other genres such as folk songs, and not just classical, piano pieces. While training the machine learning model, we found that duration produced a high accuracy, while the note values produced an accuracy comparatively lower than that. This discrepancy suggests that while the model is proficient at learning the timing and rhythm of the music, it struggles more with the melodic content. To address this, we hope to continue techniques that can better capture the harmonic and melodic structure of music.

Through this research, researching the applications of machine learning to generate more sophisticated music is shown to be a promising endeavor. The potential for people to use these models to continue to not only mimic the styles of any composer or genre, but to have the potential to also combine different types of music based on the choice of users to possibly

create a new type of music that has yet to be heard before, makes this area of research and innovative, revolutionary field of study.

Acknowledgments:

I would like to extend my gratitude to Eric Cheek, whose expertise and support has guided the success of this research project.

References:

[1] Rickard, E. (2022, June 8). Generating Music using AI. https://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=9093922&fileOId=9093927