Recurrent Neural Networks for Melody Generation

Nathan Clairmonte

1 Introduction

Automatic music generation is an exciting topic that has been the focus of much research in recent years [1-4]. While it is arguably not quite on the level of generated visual art [5], this area of AI research is quickly growing and is very exciting to learn about.

In this project, I attempted to generate melodies using Recurrent Neural Networks (RNNs) consisting of Long Short-Term Memory (LSTM) units. I designed and investigated the performance of two RNN-based systems derived from two relevant papers I found on the topic. The objective of both systems was to learn on a set of training melodies and then generate novel melodies of their own. The first system, **Gen-pitch**, learned to generate melodies based solely on pitch, which is the sequence of notes and/or chords in a melody. The second system, **Gen-duration**, learned to generate melodies based on pitch and duration, which is how long each note or chord makes a sound for.

Though the interpretation of the results obtained (i.e. the melodies generated) is subjective, my results indicated that Gen-pitch created more attractive melodies than Gen-duration did, on average. This was somewhat unexpected, as I had suspected that the added element of duration would give Gen-duration a greater understanding of the nuances of the training melodies, thus producing more attractive melodies. These results will be discussed more deeply in the results section (Section 4) of this report.

2 Background Theory

2.1 Musical Instrument Digital Interface (MIDI)

The Musical Instrument Digital Interface (MIDI) is a protocol and interface designed for talking to electronic instruments [6]. This protocol is what facilitated the translation of musical sounds into data that can be processed by a network, and vice versa. The MIDI protocol defines things like pitch, duration, volume and more for a given note or chord, and can be interpreted by a MIDI instrument to produce a sound.

The training data used for this project were 50 MIDI files containing classical symphonies by Beethoven, a renowned composer (taken from Kaggle [7]). Additionally, once a melody had been generated, the MIDI protocol was used to create a file that could be played, allowing us to hear what the generated melody sounds like.

2.2 RNNs
RNNs are a class of neuron-based models specifically designed for processing continuous sequential or time-varying data. The main architectural difference with RNNs is that they include closed-loop feedback connections across time steps in the data [8], allowing for exploitation of the temporal relationships between datapoints. Regular feed-forward neural networks don't exploit the sequential aspect of data [9], and therefore RNNs are better suited for problems that have a strong temporal element; such as natural language processing or, in our case, melody generation.

2.3 LSTMs
One important advantage of RNNs is their ability to learn contextual information across timesteps in the training data. However, one closely related disadvantage is that for the typical RNN unit, the range of its access to this context is quite limited. This happens because the influence of a given input point on the output of the network will either decay or increase exponentially as it cycles through the hidden layers of the network. This results in a situation where datapoints very far away in time will have little to no effect on the output, leading to what is commonly referred to as the vanishing gradient problem [10]. LSTMs solve this problem by modifying the standard RNN unit, replacing hidden layer summation units with "memory blocks". These memory blocks allow an LSTM unit to store and access information over much longer periods of time, which mitigates the aforementioned vanishing gradients problem [11].

3 Methods

3.1 Overview of Approach
As mentioned in the introduction, two systems for generating melodies were implemented. Both systems were derived mainly from the methodology in [1]. In this paper, researchers utilised three LSTM subnetworks, operating at different levels of granularity in time, to generate sequences at the bar level, beat level and pitch level, respectively. This allowed their system to learn nuances not only about which notes should come next in a given melody, but also about the rhythm of the melody, as defined by the bar and beat level subnetworks.

For the first system designed in this project, Gen-pitch, the approach taken in this paper was slightly simplified. Gen-pitch therefore operated only on the pitch level of granularity. This system would learn on the pitches of each note in the training data and generate melodies by predicting a pitch and assigning a fixed duration for each note or chord generated. The pitches were predicted using an RNN built using LSTM units. The general flow of information through the Gen-pitch system is shown in Figure 1.

For the second system, Gen-duration, an element of note duration was incorporated into the model, inspired by [3]. This system would learn on both the pitches and the durations of each note in the training data and generate melodies by predicting a pitch and a duration for each note or chord generated. Here, the pitches were generated using the same RNN as for Gen-pitch, while the durations were generated using a separate (albeit very similar) RNN, also built using LSTM units. The general flow of information through the Gen-duration system is shown in Figure 2.

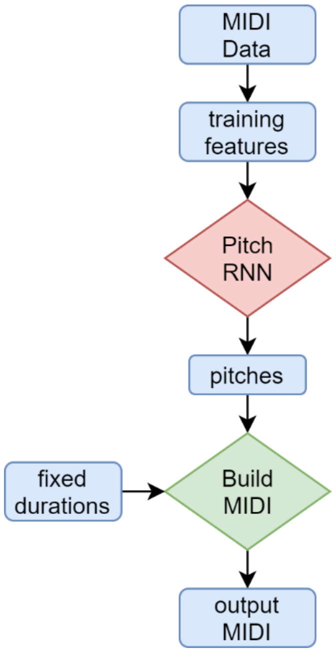

Figure 1. Representation of the information flow through Gen-pitch.

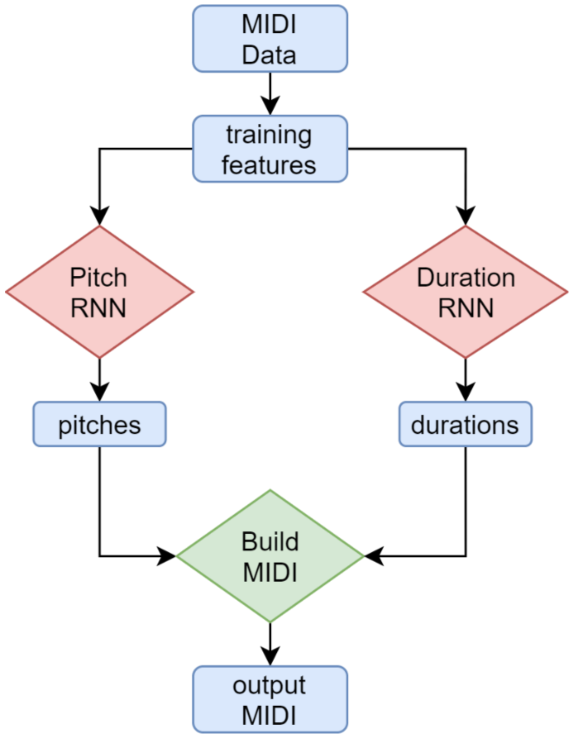

Figure 2. Representation of the information flow through Gen-duration.

3.2 RNN Architecture

The RNNs used to predict pitches and durations had very similar architectures, with the only difference occurring in the number of dense layer neurons before the output. This difference was necessary as the number of unique pitch values was consistently different to the number of duration values. This general architecture is shown in Figure 3, with the respective number of unique pitch or duration values being denoted as "n_classes" on the figure. The output of each model was an array of probability values, with the highest value corresponding to the event that the model believed should come after the input sequence.

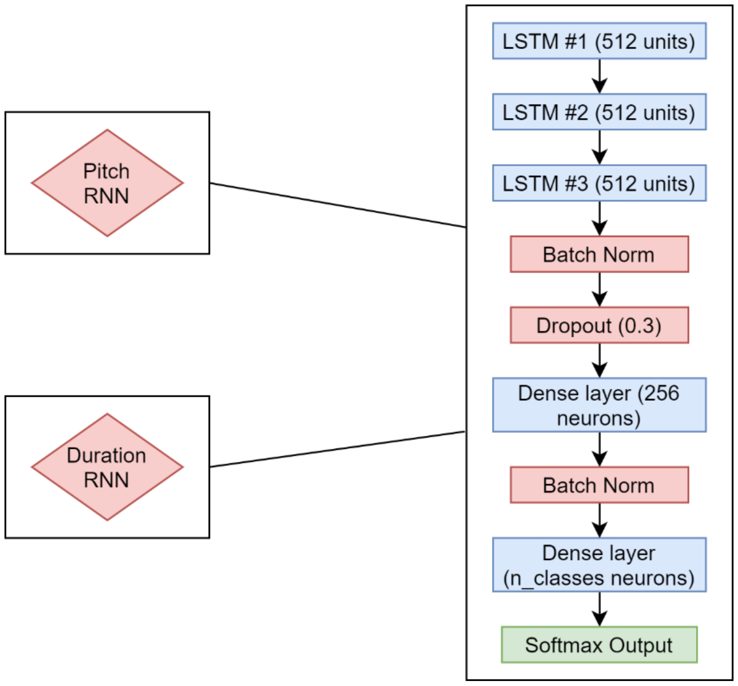

Figure 3. The model architecture utilised by the Pitch and Duration RNNs.

3.3 Data Pre-processing

Every MIDI file is essentially just a sequence of events, where each event can either be a single note or a chord (i.e. multiple notes being played at the same time). Therefore, when we load a MIDI file, we obtain a raw sequence of events from which we can extract both the pitches and the durations of each note or chord in the file. Each pitch was represented by a letter (corresponding to which pitch it is) and each duration was represented by a number (corresponding to how many quarter-beat lengths that note or chord should emit sound for).

Once all the MIDI files in the Beethoven dataset had been loaded, the pitches and durations of each note or chord were extracted and mapped to integers. These sequences of pitch and duration integers were then separated into training features and targets to be fed to the models. Each feature fed to a network was a sequence of event pitches or durations of a certain input length, and each corresponding target was the event pitch or duration immediately following that sequence. This was done to facilitate the models learning an answer to the question "given a sequence of notes, which note should come next?". The input length used for this project was 100, but shown in

Figure 4 is a simple dummy example of how the features and targets were constructed in order to illustrate the process.

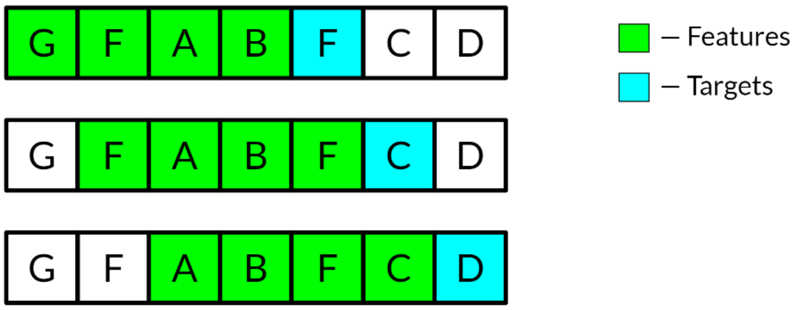

Figure 4. A dummy example (using an input length of 4) illustrating how training data were separated into features and targets. The letters shown here correspond to the pitches of notes.

3.4 Melody Generation

This section describes how to use an already trained model to generate new melodies. To start, a random sequence of events (pitches or durations) were chosen from the training data. This sequence was fed to the network to obtain the event that the model believed should come next. This predicted event was then concatenated to the input sequence, the input sequence was shifted right (to preserve its length), and that new sequence was again fed to the network to obtain a predicted output event. This process was repeated for as many times as necessary to obtain a generated melody of the desired length. Figure 5 shows a simple example (again with input length 4) to illustrate the input sequence to output event process utilised to generate melodies.

Once the entire generated sequence had been obtained, it was then used to create a MIDI file in order to hear what the generated melody would sound like. For Gen-pitch, this process was done with the pitches alone, assigning a fixed duration to each pitch generated. For Gen-duration, this process was done for both the pitches and durations, assigning each pitch its corresponding duration.

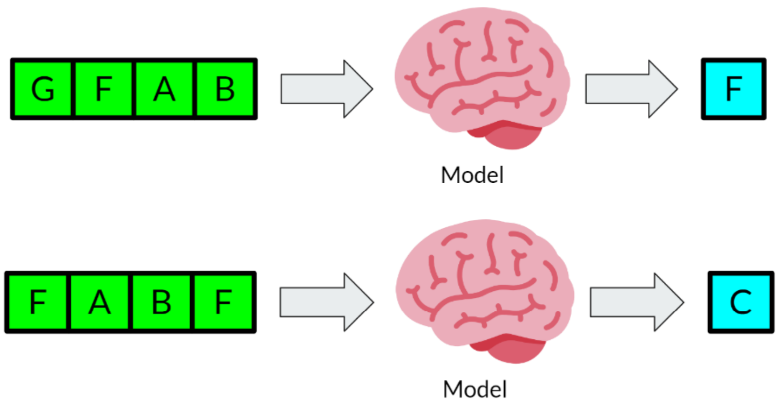

Figure 5. The process of feeding an input sequence to the model in order to obtain a predicted event, followed by adding the predicted event to the sequence to obtain another prediction. In this case, the pitches are displayed (represented by their respective letters).

4 Results

4.1 Melodies!

As a baseline, melodies were generated from both Gen-pitch and Gen-duration using an untrained network for each. After that, melodies were generated from both Gen-pitch and Gen-duration again, but this time after having been trained for 200 epochs on the Beethoven training melodies. Unfortunately, I was unable to figure out how to include these melodies in the report, but if the reader would like to hear them, they will be included as MP3 files in the ZIP of this submission.

Overall, the Gen-pitch system tended to generate more attractive melodies than the Gen-duration system. Though this is just my personal opinion, there is a potential explanation. Since each note is played for a fixed duration with Gen-pitch, a certain unintended rhythmic effect is achieved. This rhythm is something that Gen-duration would have to learn for itself, since it generates its own durations. Therefore, these results suggest that there may be something fundamentally different about how rhythm and melody present themselves in music, which lead to the observed disparity in a neural network's ability to learn their respective nuances. This could be an interesting avenue of further research into automatic music generation.

4.2 Musical Turing Test

As an additional evaluation metric of the generated melodies, a musical Turing test was implemented. A snippet of a Gen-pitch melody was played alongside a snippet of a Beethoven melody for 6 subjects, and they were asked to determine which of the two was generated and which was real. The results of this experiment are shown in Table 1.

	Gen-pitch	**Beethoven**
Subject #1	Generated	Real
Subject #2	Real	Generated
Subject #3	Generated	Real
Subject #4	Real	Generated
Subject #5	Generated	Real
Subject #6	Generated	Real

Table 1. The musical Turing test results.

Though the generated melody only managed to fool two people out of six, this was still a pleasantly surprising result, as I had not expected it to fool anyone. This leads me to believe that if I had more time to improve the models, maybe Gen-pitch (or even Gen-duration) might perform even better on this musical Turing test.

5 Conclusion

In conclusion, I designed two RNN-based systems and used them to generate new melodies after being trained on existing melodies. The generated melodies were far from perfect, but they showed promising signs of musical structure, leading to the conclusion that the pitch and duration RNNs were learning meaningful information about the structure of the melodies they saw.

One area for potential improvement, which I unfortunately did not have the time to implement, is that of musical rests. Generally, a melody does not consist of constant sound throughout the melody, and instead there are portions where there is a musical rest or an absence of sound. Rests like these are, perhaps counterintuitively, integral to a great-sounding melody. Therefore, incorporating a representation for musical rests and their durations would likely have improved the attractiveness of the melodies generated by both Gen-pitch and Gen-duration.

Another area for potential improvement is that of the training melodies. Beethoven was used for this project because MIDI files of his work were easily accessible, but I believe that the generated melodies would sound better if the systems had been trained on a more dynamic genre of music, like jazz for instance. This is just a hunch, but it was strengthened when the Gen-pitch generated melody sounded somewhat like jazz (listen to **gen-pitch_trained.mp3** if you would like to hear it). Unfortunately, I was unable to find any readily available jazz MIDI files to use as training melodies.

Overall, this was an extremely interesting project from which I learned a lot about both music and the AI concepts generally employed to automatically generate music.

6 References

[1] J. Wu, C. Hu, Y. Wang, X. Hu, and J. J. I. T. o. C. Zhu, "A hierarchical recurrent neural network for symbolic melody generation," vol. 50, no. 6, pp. 2749-2757, 2019.

[2] A. Mishra, K. Tripathi, L. Gupta, and K. P. Singh, "Long short-term memory recurrent neural network architectures for melody generation," in *Soft Computing for Problem Solving*: Springer, 2019, pp. 41-55.

[3] F. Colombo, A. Seeholzer, and W. Gerstner, "Deep artificial composer: A creative neural network model for automated melody generation," in *International Conference on Evolutionary and Biologically Inspired Music and Art*, 2017, pp. 81-96: Springer.

[4] N. Boulanger-Lewandowski, Y. Bengio, and P. J. a. p. a. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," 2012.

[5] L. A. Gatys, A. S. Ecker, and M. J. a. p. a. Bethge, "A neural algorithm of artistic style," 2015.

[6] R. A. J. J. o. t. A. E. S. Moog, "Midi: Musical instrument digital interface," vol. 34, no. 5, pp. 394-404, 1986.

[7] S. Rakshit. (2018). *Classical Music MIDI*. Available: https://www.kaggle.com/soumikrakshit/classical-music-midi

[8] L. V. Fausett, *Fundamentals of neural networks: architectures, algorithms and applications*. Pearson Education India, 2006.

[9] L. R. Medsker, L. J. D. Jain, and Applications, "Recurrent neural networks," vol. 5, 2001.

[10] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," ed: A field guide to dynamical recurrent neural networks. IEEE Press, 2001.

[11] A. Graves, "Long short-term memory," in *Supervised sequence labelling with recurrent neural networks*: Springer, 2012, pp. 37-45.