AI Beats: Generating Music with Artificial Intelligence CS221 Final Report

Krishna Kartik Patel, Lea Jabbour, Anthony Daniel Mayfield
Stanford University, Computer Science Department

1 Introduction

The Turing Test asks whether artificial intelligence (AI) can be utilized to imitate human behavior, but can AI agents learn creativity? We set to find out by attempting to train a model to compose classical piano music. Among other things, music is a conglomeration of complex rhythms, harmony, melody, dynamics, themes, and perhaps most extraordinarily, emotion. This makes the task of composing music incredibly difficult for humans, which makes it even more so for computers. Composing music is a perfect task, then, to determine whether a model can learn a uniquely human process – creativity.

2 Task Definition

For the scope of this project, we are looking to generate roughly 30 second sound clips of classical piano music. Our input is approximately 300 single-instrument piano MIDI files (Fig 1) from various classical artists, obtained from http://www.piano-midi.de/midi_files.htm. Using a Python toolkit, Music21, we are able to break songs into notes and chord objects to be fed into a gated recurrent unit (GRU) network, a type of recurrent neural network (RNN) that has memory of previous notes and chords. After training this network, we can feed the GRU some starting notes and from there have it begin predicting the following notes using a sliding window approach. We can then convert this output back to a MIDI file and later converted it to MP3 for listening. Given that music is creative and subjective, the evaluation of our output will also be very subjective. We had initially hoped to train a generative adversarial network (GAN) discriminator to evaluate our music output, but due to time constraints we used human evaluation. We created a survey to measure our algorithm's output on the following metrics: flow, correctness, melody, and overall musicality.

3 Related Works

At first glance, there is a natural similarity between text generation and music generation. However, since music arguably has a more creative aspect to it then generating certain genres of text, music generation is similar to poetry generation. For this reason, we turned to Chang et al.'s paper, "Deep Poetry: Word-Level and Character-Level Language Models for Shakespearean Sonnet Generation" [1]. From this paper, we learned to look into the usefulness of LSTMS, and in particular their ability to have a certain memory of aspects that came before. Chang et al. found relative success with Gated LSTMS that combine word and character level information in its features. However, even more useful was "A First Look at Music Composition using Long Short-Term Memory Recurrent Neural Networks (LSTM RNNs)," authored by Eck and Schmidhuber [2]. They found that Recurrent Neural Networks (RNNs) don't successfully allow for the learning of a global structure, which is a key component of music. They found more success utilizing LSTMs in composing blues music with good timing and proper structure. Furthermore, Chen and Miikkulainen's paper,

”Creating Melodies with Evolving Recurrent Neural Networks” utilized pitch and duration pairs, with pitches restricted to three octaves in order to learn ”musical grammar” [3]. They utilized three customized cells to structure their evolving RNN, but ultimately noticed that their music lacked a global, thematic, structure. The great majority of our research included attempts at music generation utilizing LSTMs because of their propensity to remember long term dependencies. However, we also examined Chou et al.'s experience with utilizing Convoluted Neural Networks as a generator and Generative Adversarial Networks. They were able to create a model that produced music that was comparable to the melodies produced by Googles MelodyRNN models [4]. Overall, these papers informed our decision to progress from our baseline to a neural network, more specifically a recurrent neural network.

4 Method

4.1 Data

As stated in the task definition, we used MIDI file representations of classical piano music with roughly 300 single instrument files from various classical artists. Using the Music21 toolkit in Python, we broke these MIDI files into notes and chords objects, the latter of which being a collection of notes played at the same time. The note objects have a few properties that we are interested in, namely pitch, volume, offset and duration. For this report, we extracted the pitch of notes and chords to use as a feature in our model. Figure 2 illustrates extracted notes and chords - or more generally tokens - sorted by the number of occurrences of that note. This shows the note frequency distribution of our training dataset.

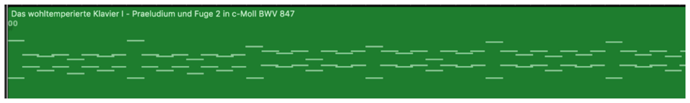

Das wohltemperierte Klavier I - Praeludium und Fuge 2 in c-Moll BWV 847
00

Fig 1 Example MIDI file for a single instrument (piano)

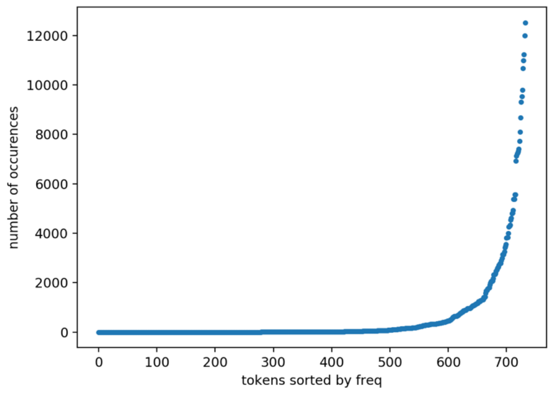


Fig 2 Token frequency sorted by occurrence

4.2 Baseline and Oracle

As a baseline, we created a music sample by picking completely random notes from a list of notes we read in from the files. This is a solid baseline to start off with as it gives us something to compare our ultimate algorithm with. As one might assume, a song composed of random notes is expected to have high variability in the note selection, and therefore show creativity, but will likely have a high level of dissonance or harsh sounds because successive notes are not picked optimally. This is why we selected an RNN for our final model, and in particular a GRU, because it has the ability to employ past information of note selection. We believed this would create an output that sounds better than the random note baseline.

As for an oracle, we initially thought that we could use existing human-composed piano music, or even ask some friends in the music department to compose something for us. However, after speaking with some teaching assistants, we determined that since our goal was to generate new and creative music, and that music can take infinite forms, that there was no ”right” answer. Indeed, we did not want the people evaluating our music to compare our clips to specific human-composed pieces, but rather to implicitly compare them to their personal notion of music. Therefore, we did not explicitly test an oracle. In a sense, the participants' overall knowledge of music, and personal likes and dislikes acted as a sort of oracle that they compared our output to.

4.3 Preparing the data

In order to prepare our data for our model, we first begin by transposing all keys and notes in our dataset to C-Major (if the original file was a major key) or A-Minor (if the original file was a minor key). This allows for better and more consistent results and was inspired by fellow Stanford students research [2]. After this, we further break down each note and chord. For notes, we simply record the pitchs string notation, and for chords, we loop through the notes in that chord and join the string notation of each of those notes (where pitches are separated by a dot). Examples of the string notation of notes and chords would be 'B-3' and 'C.E.G', the latter of which represents a chord. This encoding makes it easy to decode once we get an output from our model. From there, we map these string notation values to integers so that we can input them to our model (Fig 3).

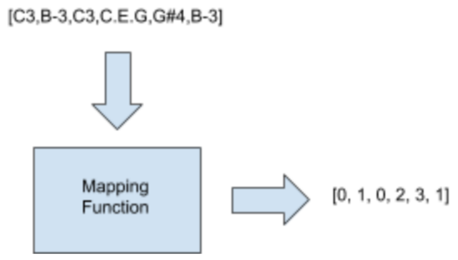


Fig 3 Example mapping function from string notation to integers [5]

Since we use a recurrent neural network (GRU), we further break this input into k-length sequences and take note of the true output: the note that immediately follows the previous k notes. As an example, consider the case where a piece of our input is $\{$A3, C3, D6, G#5, C3$\}$, and $k = 3$. We would have $\{$A3, C3, D6$\}$ as input and G#5 as the true output, as well as $\{$C3, D6, G#5$\}$ as input and C3 as the true output. Effectively, this strategy means that we will use the previous k notes to help make a prediction for the next note to follow. For this report, we have set the length of sequences k to 25. Finally, to end our preparation of the data, we normalize the input and one-hot encode the output.

4.4 Model

After doing some research online, we found a blog post that suggested training an LSTM on various one-instrument MIDI files in order to later generate music [5]. The author of this blog post, Sigurur Skli, uses nine layers: three LSTM layers, three Dropout layers, two Dense layers and one Activation layer. Initially, we wanted to start by using all of these layers and then tweak parameters and layers once we had that working. However, after meeting with our TA mentor, we realized that each additional layer added a lot of additional computational cost to our model, as well as complexity. We also learned that LSTMs typically take significantly longer to train than GRUs, although performance is typically similar. For these reasons, we decided to use a simpler model with GRU layers.

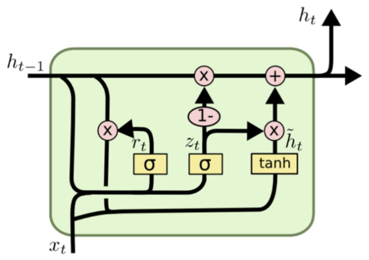


Fig 4 GRU diagram

In more detail, a GRU network makes use of update and reset gates, two vectors which help decide what information is passed on to the output. In particular, the update gate determines how much past information collected from previous time steps should be passed to the future, while the reset gate determines how much of this past information should be forgotten. Combined, these gates allow the network to effectively keep certain information from before while getting rid of anything that is irrelevant, and when trained on our dataset, this means utilizing the information from previous notes and chords in a song, and attempting to learn the most important information in order to predict the next note (Fig 4).

Our model consisted of two stacked GRU layers, outputting with dimensionality n_{vocab}, which is equal to the total number of notes found in our training data. This ensured that we could later map the output of our model back to the classes of notes. We also used a softmax activation, because we are doing multiclass classification. At each iteration of our model, we calculated the training loss using categorical cross entropy, since each of our outputs only belongs to a single class. We then trained our model on 100 epochs on a Google Cloud virtual machine (though we implemented early stopping since the training loss often did not get any better after a set number of iterations).

4.5 Music generation

After training our model, we generate music by creating a sliding window effect, using k previous notes and asking our model to predict the next note. We initialize the window by selecting one of the rows of our input at random. This row contains $k = 25$ notes, which represent the starting point for our music generation. We then ask the model to use these k notes to predict the next note. Once we have a prediction, we record it in our output notes, append it to the end of the input, remove the first note from the input, and effectively slide the window right by one note (Fig 5). We can then keep sliding the window any number of times to generate output notes. For this report, we are outputting music clips of 100 notes.

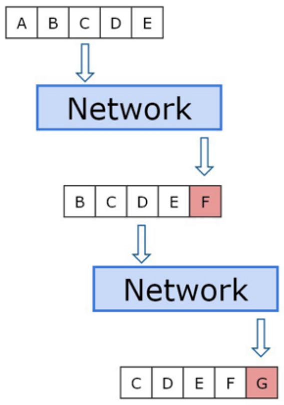


Fig 5 Generating output using sliding window [5]

In order to predict the next note, we run the input through our model and it returns an array of probabilities that a certain note is the next note (Fig 6). So to get the next note, we simply take the index of the max value in the array, run that index in reverse through our mapping function and get the string notation value of the note. Once we have all of our outputs as string notation, we can decode to turn them back into note/chord objects. Finally, we set each output note/chord at a constant +0.5 offset from the last note, and write these notes out to a MIDI file for listening.

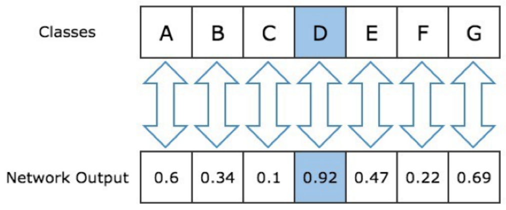


Fig 6 Prediction mapping of notes to their probabilities [5]

One challenge we faced with our model was that it often predicted the same note repeatedly and this was probably due our notes distribution where a few notes had the most occurrences (Fig 2). Therefore, the output created was not musical at all, which led us to consider another approach. Instead of choosing the note with the highest probability from the prediction vector, we instead take the top 25 notes and their probabilities, normalize, and then resample from these notes given the updated probabilities to choose as the output note (Fig 7). This allowed us to overcome the hurdle of playing the same note repeatedly and led to good results with variable note choices.

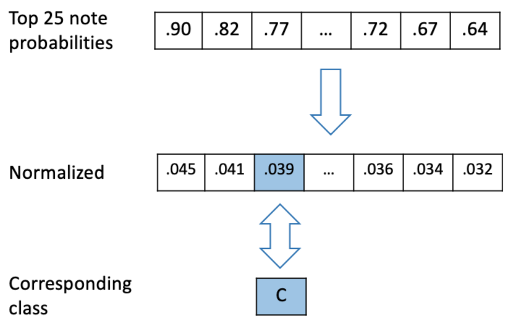


Fig 7 Sampling from top 25 notes

Using this method, we generated two music clips to be evaluated. Fig 8 shows the sheet music for each of these clips. The sheet music was obtained using GarageBand's MIDI file to sheet music converter.

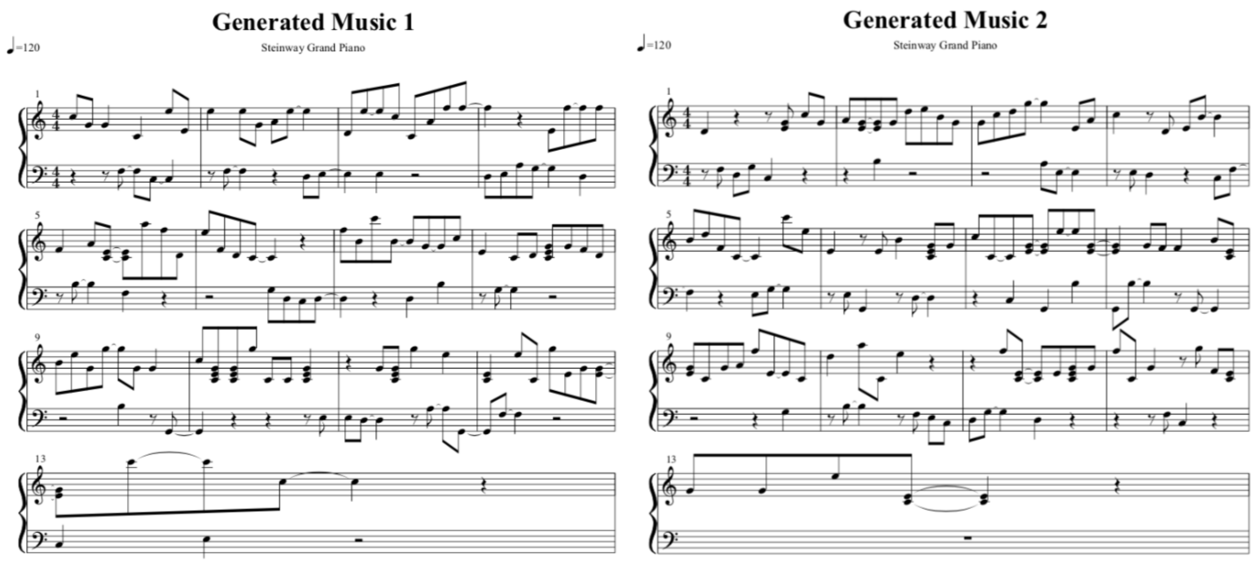
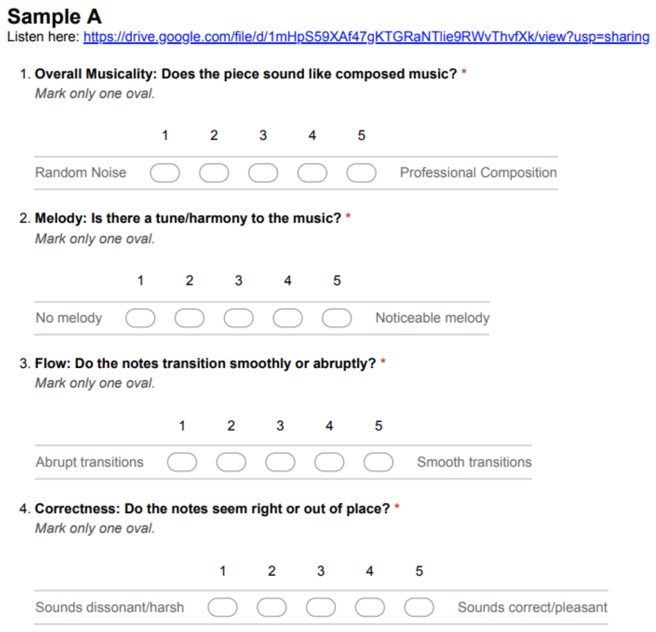


Fig 8 Sheet Music for Two Generated Music Files

4.6 Music Evaluation

In order to evaluate how well our model works, we created a Google Form where participants listened to three music clips, and rated each one a set of metrics (Fig 9). In our form we tested four metrics: Overall Musicality (Does the piece sound like composed music?), Melody (Is there a tune/harmony to the music?), Flow (Do the notes transition smoothly or abruptly?), and Correctness (Do the notes seem right or out of place?).



Sample A
Listen here: https://drive.google.com/file/d/1mHpS59XAf47gKTGRaNTlie9RWvThvfXk/view?usp=sharing

1. **Overall Musicality: Does the piece sound like composed music?** *
Mark only one oval.

	1	2	3	4	5	
Random Noise	○	○	○	○	○	Professional Composition

2. **Melody: Is there a tune/harmony to the music?** *
Mark only one oval.

	1	2	3	4	5	
No melody	○	○	○	○	○	Noticeable melody

3. **Flow: Do the notes transition smoothly or abruptly?** *
Mark only one oval.

	1	2	3	4	5	
Abrupt transitions	○	○	○	○	○	Smooth transitions

4. **Correctness: Do the notes seem right or out of place?** *
Mark only one oval.

	1	2	3	4	5	
Sounds dissonant/harsh	○	○	○	○	○	Sounds correct/pleasant

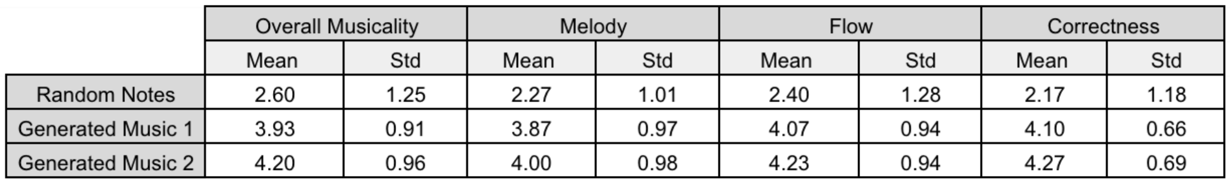
Fig 9 Music Evaluation Google Form

Each metric was rated on a scale of 1 to 5, where 5 was the best (i.e. closest to a professional composition). The first music clip we had participants listen to was generated by picking random notes from a list of notes seen in the training dataset. We included this in our survey so that we could see if our model improved over this baseline. The next two clips were generated as described in our methods section. After participants had submitted their ratings, we informed them that the first sample they heard was made by playing completely random notes, and that the next two samples were composed by our AI algorithm. We sent this survey to friends and family, with a range of musical knowledge. We wanted to ensure we had this range because music is so subjective, and we wanted people to focus on how the music made them feel, given that one of our goals was to capture the emotion and creativity of human composition. Since we had such a varied audience, we did not ask participants to rate the genre of the musical piece. Additionally, our music clips did not truly have a beginning or end, so it would be difficult to assess how ”classical” the clip was, so we will save that for future work.

5 Results and Analysis

We received 30 responses on our Google Form. We took the average and standard deviation for the different metrics across samples, and summarized the results in Fig 10.

We can see that the first music clip, which was generated by sampling random notes, has the lowest scores of any sample in each category. Interestingly, the average scores are all greater than 1, which means that on average, participants thought this was slightly musical. The category with the lowest score for the Random Notes sample is ’Correctness’. This makes sense, because



	Overall Musicality		Melody		Flow		Correctness	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Random Notes	2.60	1.25	2.27	1.01	2.40	1.28	2.17	1.18
Generated Music 1	3.93	0.91	3.87	0.97	4.07	0.94	4.10	0.66
Generated Music 2	4.20	0.96	4.00	0.98	4.23	0.94	4.27	0.69

Fig 10 Results from music evaluation survey

sampling random notes is very likely to result in a dissonant succession of notes. The Random Notes clip probably shows the highest level of creativity, but at the expense of note correctness and musical structure.

Both music clips generated using our algorithm perform quite well compared to the Random Notes sample. We can see that Generated Music 1 was rated slightly lower than Generated Music 2 on all metrics. This is interesting because both clips were generated from the single model we trained. This shows that there remains some ”creative” license in the generation portion of our algorithm. In both generated music clips, ’Flow’ and ’Correctness’ were rated the highest, which shows that our algorithm was able to learn about which notes should follow one another - both in terms of harmony and transitions. We can see that ’Flow’ and ’Correctness’ were quite low for our Random Sample, which shows that the GRU model improves significantly over baseline. Finally, we can see that ’Melody’ seems to be the most difficult metric for our model to capture. The average ’Melody’ score is the lowest of all metrics for both generated music samples. However, these scores are still quite high for the generated music clips (around 4 out of 5), showing a lot of improvement over the baseline (close to 2.3 out of 5).

From the standard deviation measurements, we can see that the Random Notes metrics all have quite high standard deviation (between 1 and 1.28), while the generated music clips have lower standard deviation (between .66 and .98). This is interesting, because it captures how ’certain’ participants were in their ratings. There was evidently much more discord for the Random Notes clips than the Generated Music clips. We asked a few participants to comment on the Random Notes clip, and found out that some of them thought the dissonance and harshness meant that it must not have been the result of human composition. However, others agreed that it sounded a bit dissonant and unlike classical piano music, but believed it may have been the result of contemporary piano music, and therefore human composition. This is an interesting insight that shows just how difficult it is to capture the ”goodness” of music.

6 Other Approaches

6.1 Search Problem

As a first attempt at music generation, we had modeled our task as search problem, and used Uniform Cost Search (UCS) to select notes that were more probable given the note that preceded it. The components we are follows:

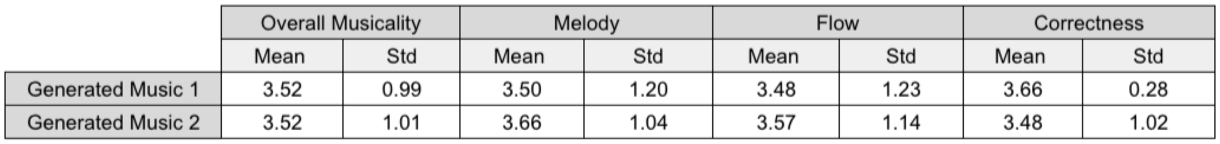
States: (last note, total notes so far)
isEnd: total notes so far == the number of notes we want in the song

Cost = the likelihood of a note following another note (based on songs we read in)

While the UCS baseline allowed us generate songs with notes that are more likely to follow one another, it often faced the issue of getting stuck in a repeated pattern for the entirety of the clip. This is because it lacks variability, and sticks to a more rigid structure of notes, which often means choosing the same few notes over and over. This was probably due to the fact that our dataset notes distribution was quite unbalanced with a few notes having very high occurrences (Fig 2) and therefore having high chances of being played over and over.

6.2 Beethoven Only

Once we made the transition to using a GRU, we set up a basic network in order to see if the network was capable of producing good results. Thus, we utilized a single GRU layer and trained on a smaller dataset consisting of 16 Beethoven MIDI files. To assess our results, we used the same categorical evaluations, namely overall musicality, melody, flow, and correctness and rated them on a scale of 1 to 5 and surveyed 42 people. Figure 11 shows our results. From this, we can see that this experiment was clearly better than just choosing random notes (see Fig 10) and allowed for a decent level of musicality. Of course it did not achieve the same scores as our model trained on the full dataset and with additional layers. This is not surprising, because we had fewer files to work with and a simpler model (1 GRU layer versus 2). The standard deviation of responses is also higher than in the more complex GRU model, meaning that there was a bigger spread in participant opinion here, as expected.



	Overall Musicality		Melody		Flow		Correctness	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Generated Music 1	3.52	0.99	3.50	1.20	3.48	1.23	3.66	0.28
Generated Music 2	3.52	1.01	3.66	1.04	3.57	1.14	3.48	1.02

Fig 11 Results from Beethoven-only music evaluation survey

6.3 Additional Features

Some other aspects we considered for feature extraction were duration and volume. After all, there is more to music than just notes; musicality also depends on dynamics and varying note lengths. However, in trying to incorporate additional features, we ultimately ran into some problems, mainly because generating too many classes led to memory issues on our Google Cloud GPU given the number of credits we had to work with. We first tried to combine a note, its duration, its and volume into a single token to be fed into our two layer GRU network, but this maxed out the memory on the GPU, even after greatly reducing batch size and the number of nodes in our first GRU layer and trying to decrease the number of token classes by rounding durations up to the nearest eighth notes and volume to the nearest multiple of 5. Another thing we tried was training each of these features separately and then later joining them together in the generation phase. While this did help overcome our memory issue, it didn't contribute to creating musical outputs.

We figured this was due to the fact that duration, volume, and notes are often correlated directly with one another. For instance, rising notes are often followed with an increase in volume and falling notes with a decrease in volume. Therefore separating these features for training and later combining them in the generation phase ultimately ended up being less effective than just using the notes for our feature extraction.

7 Conclusions and Future Work

The original goal that we set out to accomplish for this project was to create classical piano music. From our results, we can see that our AI algorithm is certainly capable of composing music that is musical and correct while also having elements of melody and flow, but as far as creating truly ”classical” music, we don’t believe the scope of our project directly accounts for this. This is because of how complex classical music is, having features that our simplified model could not account for in addition to us having to remove features such as note durations, volume, and the ability to overlap notes. In order to reach the genre of classical music, however, there are some potential future steps that we can take.

One aspect we would be interested in focusing on is creating a complete composition, from start to finish. Currently, our model creates an output that just seems to start out of the blue and just cuts off after a set number of notes. One interesting avenue to pursue would be to see if we can generate a beginning and an end to attach to our current output. One idea could be to train a model on only beginnings and a model on only endings and to concatenate these results to our model to arrive at a complete composition. For more complex pieces with multiple movements, we could extend this idea to train separate parts of the overall piece, and stitch them together.

Another area that would be interesting to pursue is finding a way to successfully include other features such as volume and duration, which might require obtaining more GPU memory in order to run all the produced classes (or finding a way to reduce the number of classes in an effective manner), which would solve the problems we ran into, as discussed earlier. It would also be interesting to allow for note overlaps that are more complex than just chords, such as a melody playing over a chord. Finally, it would be very interesting to train different models on different musical genres (e.g. pop, rock, contemporary), in order to compare output across genres. We could then ask music experts to aid us in classifying our music generation output, and use the success rate to further enrich our models.

While there is still some future work to be done in order to truly complete our original goal to create classical music, we believe the project as is provides support for the claim that AI is capable of learning and exhibiting creativity, and this is displayed in the ability of our model to generate music that people, based on our survey, find enjoyable, musical, and creative.

8 Acknowledgments

We would first like to acknowledge Sigurour Skuli, the author of the blogpost ”How to Generate Music using a LSTM Neural Network in Keras” from which a lot of the infrastructure and inspiration for this project came from [5]. We would also like to acknowledgment our professor, Percy Liang, as well as our TA/mentor, Stephanie Dong. Lastly, we would like to thank everyone who completed our surveys in order to evaluate the outputs of our models.

9 Music Links

If you would like to listen to the music clips we referenced in this paper, you may access them via the following links:

- Random notes: https://drive.google.com/file/d/1mHpS59XAf47gKTGRaNTlie9RWvThvfXk/view?usp=sharing
- Generated music 1 from Beethoven-only training data: https://drive.google.com/open?id=1Mns1iPaTp1ZUqXPwZ0nO9kbqz__UeW_M
- Generated music 2 from Beethoven-only training data: https://drive.google.com/open?id=1PcKBL_az4GZGCENMBbidwtFPTKYg9xPk
- Generated music 1 from all training data (final model):
  https://drive.google.com/file/d/1URMpjoXhCjZVBJDdT-_w6tHtCIlfmEKz/view?usp=sharing
- Generated music 2 from all training data (final model):
  https://drive.google.com/file/d/1l0fCbUQmPF8oCxAIlMCwLOd7Onq-QnZk/view?usp=sharing

10 References

[1] https://web.stanford.edu/class/cs224n/reports/2762063.pdf

[2] http://people.idsia.ch/ juergen/blues/IDSIA-07-02.pdf

[3] http://nn.cs.utexas.edu/downloads/papers/chen.ijcnn01.pdf

[4] https://arxiv.org/pdf/1703.10847.pdf

[5] https://towardsdatascience.com/how-to-generate-music-using-a-lstm-neural-network-in-keras-68786834d4c5

[6] https://cs224d.stanford.edu/reports/allenh.pdf

[7] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[8] https://arxiv.org/pdf/1804.07300.pdf

[9] https://medium.com/cindicator/music-generation-with-neural-networks-gan-of-the-week-b66d01e28200

[10] http://mogren.one/publications/2016/c-rnn-gan/mogren2016crnngan.pdf

[11] http://people.idsia.ch/ juergen/blues/

[12] http://community.wolfram.com/groups/-/m/t/1435251

[13] https://towardsdatascience.com/generative-adversarial-networks-explained-34472718707a

[14] http://yoavz.com/music_rnn/

[15] https://towardsdatascience.com/lstm-by-example-using-tensorflow-feb0c1968537

[16] https://github.com/aymericdamien/TensorFlow-Examples/blob/master/examples/3_NeuralNetworks/recurrent_network.py

[17] https://www.oreilly.com/ideas/introduction-to-lstms-with-tensorflow

[18] https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be