An Exploration of Neural Networks and Music Generation*

Omkar Apte, Anand Hande, Santoshreddy Nukala, Sahith Reddy, and Krithik Chitla
(Dated: November 9, 2020)

Generating music through the use of neural networks is an extremely new and active are of research. This project explores how we can generate new music using a Long Short-Term Memory (LSTM) network. Sound is recorded as a continuous waveform which is complicated to analyze, so we focused on music encoded in the MIDI format, which gave us a standardized way to digitize music. We chose video game music that was composed for a single instrument so that the training data would have a degree of similarity (set by the theme of the game) and would not be too complicated to analyze. We found that our LSTM network was able to extrapolate very well from simple music, but tended to get stuck in local minimums, which translated to continuously playing the same note. With complicated music, the network was only able to maintain a cohesive musical phrase for a short time before playing dissonant notes. For future exploration, we would like to train the network over a larger data set, vary the number of epochs, and the sequence length to see how that would affect the quality of the network outputs. Additional extensions to the network include handling multiple instruments, adding vocals, and fusing genres.

I. INTRODUCTION

Music is an abstract form of art that is very prevalent in the modern day. While most people understand it at an intuitive level, there is a rich structure within every composition. One needs to consider pitch, melody, chords, harmony, rhythm, and many other features in order to compose a quality piece. Since we are not professional musicians or composers, we wanted to see if it was possible to generate music through the use of a neural network. This is a growing area of research as more groups venture into using computers in "creative" fields. In this paper, we will explain our methodology and the findings we obtained as we attempted to generate original music using neural networks.

Many people enjoy listening to music, so we felt that our project would be interesting for laypeople, our peers, and experts. The ability to play an instrument (or sing) in addition to composing music is what separates professional musicians from the average hobbyist. We chose this project because the idea of using AI to generate new music to help bridge this gap was intriguing. Artificial music generation is an extremely active area of research, and we were interested to see how we could contribute to this field.

There are many applications for a neural network that is able to generate music with little human input. Automatically generating royalty-free music is very useful for content creators, who constantly have to worry about demonetization and DMCA takedowns because of copyright policies on most content hosting platforms, such as Twitch and Youtube. It can also serve as an assistant for professional musicians, who can feed it a motif that they have already come up with and then use the output for the nueral net as inspiration to extend the composition. The ability to use existing data in order to create something original, yet similar to the existing content is valuable. The techniques used in this network, once refined, could potentially be generalized to generating arbitrary audio waveforms, which has immeasurable value. We felt that this topic would challenge us as well, since neural nets are an advanced extension of the material we learned throughout the semester.

II. BACKGROUND

Sound is continuous so it needs to be discretized in some manner so that we can effectively analyze it. We used the MIDI file format which is a modern standard for digitizing music. Our original goal at the start of this project was to focus on retro video game music to generate new music encoded as MIDI files. The initial approach was to convert MIDI files into C.S.V. in order to simplify data analysis. We encountered difficulties converting MIDI to C.S.V because the structure of a MIDI file varies greatly based on the number of instruments present in the song. Due to difficulties in parsing the MIDI files and inconsistencies in conversion to C.S.V., we switched to working directly on the MIDI files.

A. Music Theory Foundation

In order to understand the design decisions that were made during the model development and training process, we must first discuss some basic principles of music theory. Western classical music has a structure that can be described by a few key concepts. Modern western music, such as pop, rock, and hip-hop do not inherit the same structure necessarily, but the system of notation used to describe the music is the same.

* We would like to thank Dr. Shankar Bhamidi for his support and advice as we tackled this unique and advanced project throughout the semester, and the weekly feedback of our peers.

1. Harmony

Musical notes are assigned the first 7 letters of the Latin alphabet: A,B,C,D,E,F,G. These are used to divide pitch ranges into octaves, which are eight note groupings in which the first and last note have the same letter assigned to them. This construction is such that the last note in an octave is double the frequency of the first note. In between these letter pitches are the raised and lowered pitches denoted by adding a sharp (♯) to raise the pitch, or a flat (♭) to lower it. The frequency of a sharp or flat note is halfway between that of the note letters surrounding it e.g. C♯ has a frequency halfway between C and D, E♭ has a frequency halfway between D and E. Enharmonic notes are sharp or flat notes that have different notation but are really the same frequency e.g. C♯ and D♭. There are reasons for choosing one name over the other, but they are not relevant to this discussion.

Western music can be classified by a key signature, which is a group of 8 notes that are some combination of normal, sharp and flat notes. The rules for choosing which 8 notes go together are not important for understanding this project. The 8 notes in a key signature generally specify what notes are "allowed" because they all sound good together and determine which chords (2 or more notes played simultaneously) can be played.

2. Rhythm

The other main structure is rhythm, which gives the music what is colloquially known as the "beat." Musical notation will indicate a fraction at the start of the piece that indicates the beat structure of the song, called a time signature. The most common one that occurs in Western music is 4/4. The denominator indicates what the base unit of the beat is, which is a quarter note in this case. The only other allowed denominators are 8 and 16 corresponding to 1.5 eight notes and 1.5 sixteenth notes respectively, but 4 and 8 are by far the most common (There is no rule for figuring out the beat unit, these just are the three different possibilities).

The numerator indicates how many instances of the beat unit occur before the current "block" of notes ends, which is called a measure. In the common 4/4 example, a musical block, or measure, contains 4 quarter notes worth of music. The tempo of a song indicates the speed, which is indicated by a number of beats per minute (bpm) in modern times. The bpm of a song indicates how much time one beat unit takes up. For example, most military marches are written in 4/4 time at 120 bpm, which means that each measure is 4 quarter notes long and takes 2 seconds to play because a beat (in this case a quarter note) lasts for 0.5 seconds.

The previous discussion within this section is a rigorous formulation of ideas that humans intuitively understand when listening to music. The next section will discuss how we developed our model and the concepts in this section will be used to show how we chose our training data. They will also come up again in our results when we are evaluating the quality of our outputs.

III. NEURAL NETWORK THEORY

A. Basic Neural Network

In order to understand the advantages of a Long Short-Term Memory (LSTM) neural network, we first need to delve into the mathematics behind neural networks in general. The simplest way to understand a neural network is to first look at the base case, with only one neuron, one output, and n inputs. We shall denote the input sequence as $x_1, x_2, \dots x_n$. This neuron has associated with it a sequence of weights $w_1, w_2, \dots w_n$. Similar to the linear regression framework, each w_i denotes the strength/presence of feature x_i. If we introduce a bias term, then our output resembles that of least squares: the inner product of the feature vector with the weights with some bias, $z = \mathbf{xw} + b$ This setting suffices for linear models, however we are able to move past the linear setting through the use of activation functions.

The purpose of an activation function is largely problem dependent, so there are many different types of activation functions. In this example with one neuron and one output, we can consider the sigmoid function, $\sigma(z) = \frac{1}{1+\exp(-z)}$ in order to solve a classification problem. We use the activation function to produce an output, but often activation functions are used for hidden layers (intermediate layers between input and output) as well.

Assuming this framework, we have a method for deriving our output from a given input. The next step is to determine the optimal weights for the network by fixing the weights to random values and correcting the weights based on error. In a perfect world, this can be done by computing the gradient of a cost function, which tells us the difference between our prediction and the true response. The gradient gives the direction of steepest decrease which we use in back-propagation to adjust the weights in a way that will reduce error the most. Back-propagation starts at the output layer and going to the input layer, by estimating the gradient, which determines how the weights need to be adjusted in order to increase predictive accuracy.

B. LSTM Differences

The basic neural network described above is suited for many types of problems, but there are also many ways in which it falls short. The neural network described above does a good job for predictions that do not rely on prior information, but for our problem, we need prior information for the current prediction. In order to predict what the next note in the sequence is, the network must have

some way of remembering what came before. If we were to predict a new note based solely on the where we are currently, we would get a piece of music with sporadic, unrelated notes. The solution to this is to use a Recurrent Neural Network (RNN), of which LSTM networks are a subset. The main feature of RNNs is that the outputs are fed back into the input in successive epochs in order to maintain access to prior information. Instead of neuron outputs flowing one way towards the final output, an LSTM cell will take information from other parts of the cell. Intermediate outputs can now be related to each other, enabling the LSTM network to predict sequences of data.

While the LSTM network's ability to remember long-term information is valuable, much of this information does not play a role in prediction. Too much reliance on what comes before can result in overfitting. Because of this, We need to introduce another component called the forget gate. Similar to the simple neural network described above, an LSTM cell has an input and output gate to both take information from previous layers and feed them into subsequent layers. The forget gate uses activation functions to "take out" certain weights from previous information. This is valuable for music generation because we can "forget" notes that are not significant towards the entire song, e.g. they occurred long enough ago that they are not relevant to the current song structure. Computational overhead is another thing we had to consider going in to this project, and LSTM neural networks are one of the most efficient ways known to process music, which is why we chose to use this technique.

IV. MODEL DEVELOPMENT

A. Design considerations

As all of us were unfamiliar with neural networks, so we started from a tutorial created by Tyler Doll (see link in Appendix). The first task was to process the MIDI files and extract the notes (note: here the term 'notes' is used to loosely refer to each part of the music. When we want to refer to the musical definition of a note, we specify accordingly). We decided to use the music21 package for Python which would let us extract the musical (note, octave) pairs for each note of a song from the MIDI file, which let us circumvent the need to convert the file format. From this, we were able to map each unique pair to a natural number which reduced the problem to a classification task: given a sequence of (note, octave) pairs, determine which (note, octave) pair comes next.

Once we had mapped the notes to integers, which represented the class labels, we appended each integer to an array. First we needed to decide on a sequence length, which is number of consecutive notes that will determine the size of the input. Based on current work in the field, we felt that 100 notes was appropriate, given that songs were only a couple minutes long. These 100 notes served as the input to the neural network. The 101st note that followed is the desired output. Starting at index 0 and ending at the length of the note array minus 100, we considered each input sequence and output note. Combining the output predictions sequentially generates the entire song.

After experimenting with the structure of the network, we decided to use three LSTM layers, two dense layers, three dropout layers, and two activation layers. The purpose of the LSTM layers is to utilize prior network outputs as intermediate inputs, which is discussed in detail above. The dropout layers give us a systematic way to zero out part of the inputs which will prevent overfitting, and increase sparsity in the matrix, speeding up the computation. The dense and activation layers perform the same function as the single neurons described in section III A.

The first activation function we used is the Rectified Linear Activation Function (ReL). ReL is a piece-wise linear function which will "zero" out negative inputs and sustain non-negative inputs. In mathematical terms, we have $R(z) = max(0, z)$. The non-linearity at 0 allows for some inputs to be ignored, simplifying the overall model. While other activation functions are undefined at 0, ReL allows for values to be "zeroed" out which is useful as a hidden layer of the network.

The second activation layer used the softmax function. The softmax function converts the last numbers in each cell to one of the classes we are trying to predict, which in this case are (note, octave) pairs. The behavior we want is for the layerinputs $(o_1, o_2, ..., o_n)$ to be converted into a vector $(0, 0, ..., 1, 0)$ where 1 indicates the specific (note, pitch, octave). Softmax behaves similar to argmax, but it is differentiable (it is an extension of the logistic function), making it preferable for gradient descent.

B. The Algorithm

After training our network on all of the songs, we have a network that can take in a sequence of notes and output a prediction of the next note. The weights are configured for predictions given the test songs, so we can retrain the original neural network, but initialize it with the weights we have found from this instance of training. In order to create new music, we want to predict many notes, which will get stitched together to form the song. The general procedure to do this is as follows:

1. Pick a random point in array of notes at an index ≥ 99
2. Select the next 100 notes after that index
3. Use these notes to generate a prediction note
4. Remove the first note from the 100 note selection, and append the new prediction note to the end

5. Add the new note to the list of output notes
6. Repeat the above process using the modified sequence of 100 notes for as long as needed

Once the note sequence is created, we have to convert back to a MIDI file using music21. We made the assumption that each note was had an offset of 0.5 from the prior note for formatting purposes. For more details on implementation, see the Google Colab notebook with our code.

C. Training Data Selection

When selecting training data, we thought it would be important to train the network on incrementally more difficult music. The simplest key signature is C major, which is unique because it has no sharp or flat notes. On a piano, this corresponds to playing only the white keys. C major is by far the most popular key in all of western music for this reason, which makes finding training data very easy. The most common time signature is 4/4, which is 4 quarter notes per measure. This naturally leads to musical phrases being constructed in multiples of 4, which adds a clear structure to the music, which we hope that neural network will pick up on.

Even though our original intent was to analyze Mario music, we realized that first we needed to be able to work with single instruments. We still wanted to use videogame music because of the consistency of style, so we decided to use soundtracks from *I Am Setsuna*. This game is known for the quality of its soundtrack and all of the music is composed for a single piano, which makes the MIDI files very simple to process and train on. We found MIDI tracks of the various songs from the game, which we sorted into two groups for training. The first group of songs contained only the simplest music that was in 4/4 time and did not change key signature or time signature so that we could evaluate the baseline performance of our model. The second group contained the remaining music that included key signatures and more complex time signatures so that we could limit test the capability of our model.

D. Challenges

One of the first problems we faced was analyzing songs that had multiple instruments. This is because multiple instruments in MIDI files overlapped in a way that scrambled the output of music21. There was also the difficulty of computing time - it took almost a full day to train the model on 6 songs. There were many extensions and explorations that we would have like to consider, but the sheer amount of computing time and the time constraint of the semester mean that we could not get to them. We discuss these in detail in Section VI A. Overfitting was a constant concern which was exacerbated by the need for large amounts of epochs before the network was able to produce any sort of recognizable output. This resulted in the same notes and rhythm playing on repeat which was not ideal. Repetition in music is not always a bad thing, but it has to be done in a way the supports the structure of the songs (such as choruses in modern music), not mindless copy/paste of the same rhythmic and harmonic structures. There were also issues with transitions, which lead to some erratic jumps and musical dissonance. We discuss how one might deal with this issue in Section VI A.

V. RESULTS

The first run of training was done using the Group 1 training data with a sequence length of 100. We used model checkpoints after every 50 epochs to save the weights and generate outputs up to 200 epochs. This had the advantage of letting us observe the progress of the model over time and saving weights in case the training was interrupted for some reason. The 50 epoch output is shown in Figure 1. We can see that the network was able to identify what a note is, but was not able to do much else with it since it continuously played the same note. The 100 epoch output is shown in Figure 2. The network was now able to understand rhythm as we can see by the variation in note lengths, but it was still unable to understand note relationships, so the actual sound was rather cacophonous. After 150 epochs, the network was able to achieve something similar to a practice exercise. We can see that the network found good note runs that resembled a melody and used interesting rhythm, but had not found a way to harmonize that with supporting chords. This can be seen in the bottom staff of Figure 3, which is relatively sparse. The 200 epoch output resembled something more like what a very new player would attempt since each measure has all the structure of rhythm and harmony that is desired, but does not transition between measures very well, as seen in Figure 4.

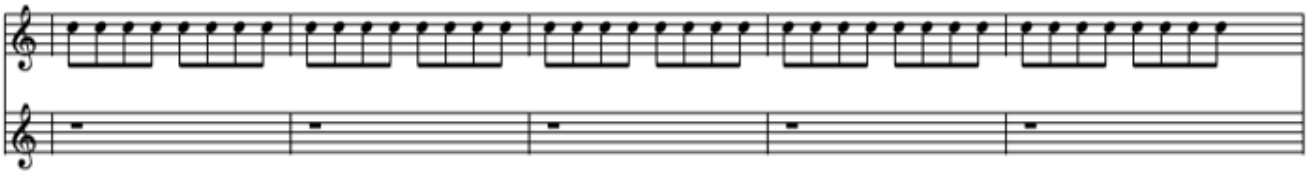

FIG. 1. 50 Epoch output for group 1, sequence length of 100

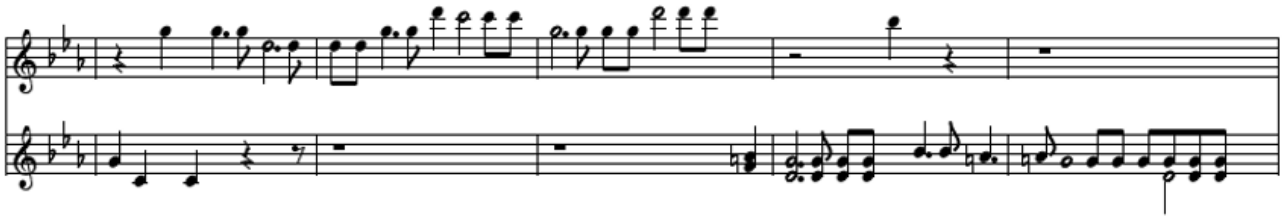

FIG. 2. 100 Epoch output for group 1, sequence length of 100

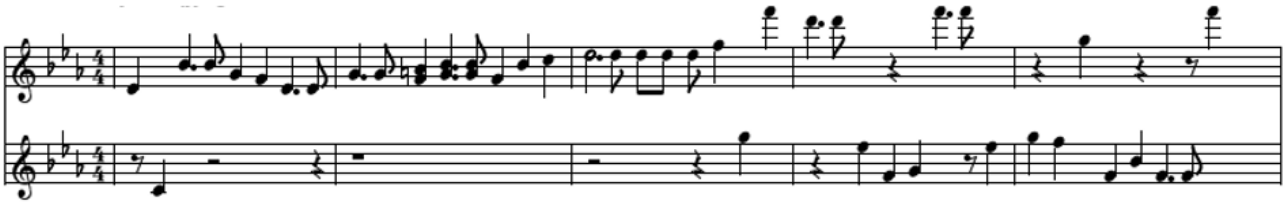

FIG. 3. 150 Epoch output for group 1, sequence length of 100

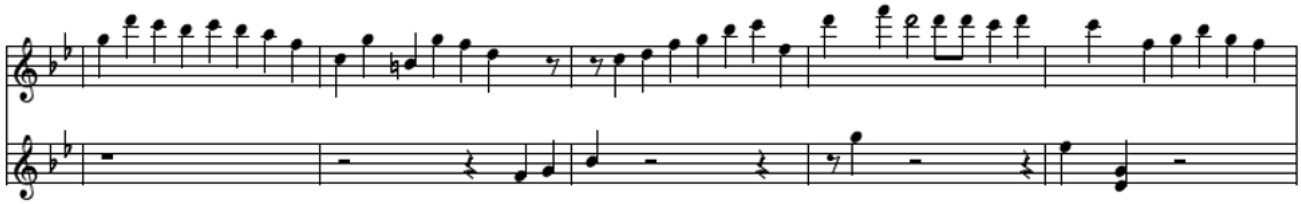

FIG. 4. 200 Epoch output for group 1, sequence length of 100

For the complex group (Group 2) we did not have much success after 50 epochs. At 50 epochs, the network was only able to pick up an alternating pattern with two notes and no interesting rhythm at all, as seen in Figure 5. For all the outputs afterwards, we got some variation of dissonant notes with good rhythmic structure. The only real progress that it made was understanding the complex beat structure. This suggests that given our sequence length, the neural net was unable to understand the melody and harmony in the more complicated pieces due to the more complex beat structure, as seen in Figure 6.

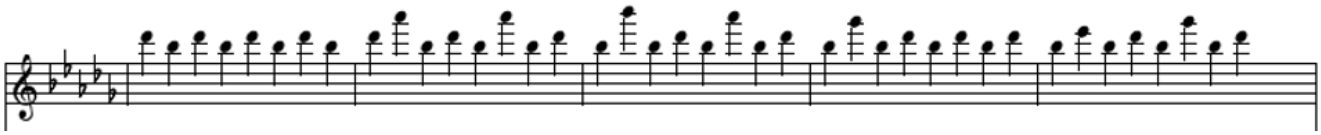

FIG. 5. 50 Epoch output for group 2, sequence length of 100

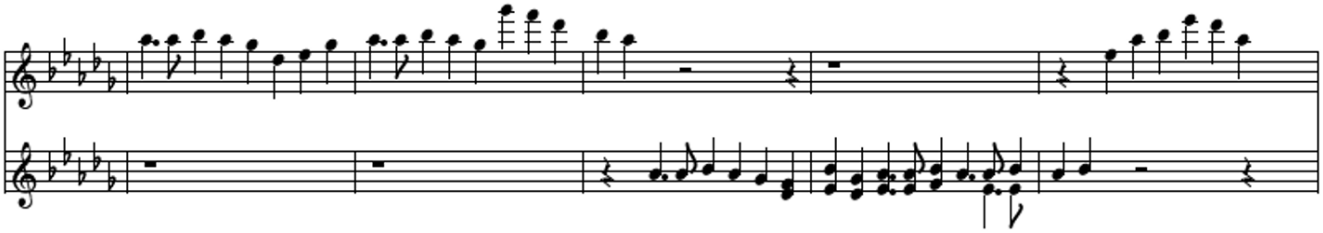

FIG. 6. 200 Epoch output for group 2, sequence length of 100

This behavior can be quantified by our model loss per epoch, shown in Figure 7. There is a spike that occurs right before 50 epochs where we see the network figure out the note repetition pattern, but no other significant spikes occur, which means that the network is overtraining within the same pattern that it picked up on within the 50 epoch regime. This is why the more complicated training data only produced outputs that contained only rhythmic development with no harmonic structure.

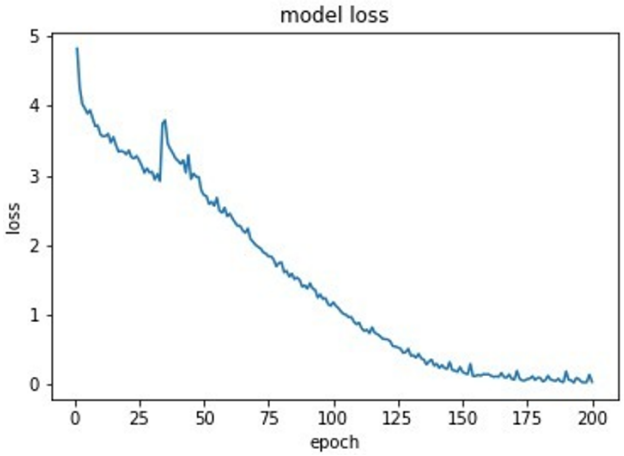

FIG. 7. Model Loss for Group 2

We can compare this to the model loss for group 1, where we see several spikes in loss right before the network figures out the key features that were discussed above, as seen in Figure 8.

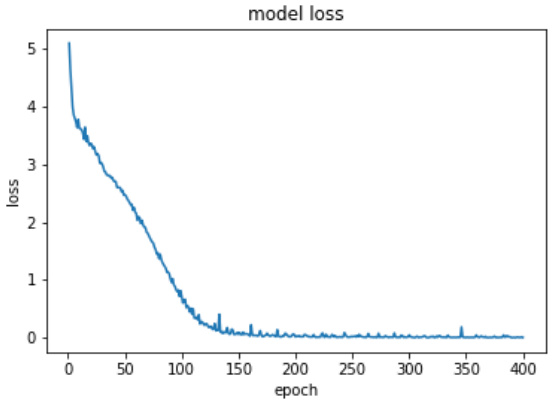

FIG. 8. Model Loss for Group 1

The network is continually decreasing in loss at the beginning, but not really experimenting with features until right before epoch 150. This corresponds to the behavior that we observed, since the network did not produce recognizable outputs until then. Beyond that point, we see some small spikes right before the network learned a new important feature of music, but the overall loss was very low. This is in agreement with he better quality outputs we got for epochs greater than 200.

VI. CONCLUSION

The project gave us thorough introduction into neural networks and their application. Based on our results and difficulties, we were able to understand the intricacies of LSTM networks, and their advantages and pitfalls. Our model was able to understand different pitches, basic rhythmic structure, but not how to transition between different parts of music. The network falls behind on its understanding of harmony because it still has issues with

transitions and dissonant chords. We believe that the current model is an extremely promising first step, but needs a lot of improvement. Fortunately, we recognize the shortcomings of our process and network, which can be addressed in future iterations of the model.

Music is an art form, and therefore subjective, but that makes our project all the more fascinating. If a human fed in preferred music to a model like ours, it will be able to make new music which is pleasureable, and possible useful in certain applications, as discussed in I. There is a lot of scope for AI in music generation and much more if we are able to generalize the models capabilities to continuous audio waveforms.

A. Future Exploration

There is much to discuss about how to extend this project further, but first we must address the issues that we faced. Given more time, we would have been able to train on larger data set, and test out more epochs and sequence lengths, which would give a better understanding of the performance of the LSTM network. We were unable to do so on this project due to the large computational overhead and the time constraints of the shortened semester. We could then take the network forward and retrain on successively more complicated musics using initial weights from previous training, which might improve the overall performance of the model.

The next logical extension would be adapting the network to handle multiple instruments. Most music obeys a hierarchical structure where one or a few instruments are the "focus" at any given time, and is supported by the other instruments via harmony or percussion. Being able to generate music with this structure would be a huge step forward because that would make the model competitive with modern instrumental genres like Jazz and EDM. A simple extension from this point is also the inclusion of vocals, but that comes with the additional complexity of speech process. It is still interesting to consider the possibility. The final extension we considered is training across subsets of different genres and generating new music that is a fusion of the training data, which could revolutionize the music industry. This is an extreme reach goal that is very far off based on the current state of the field. Overall, we believe that this model is a good first step as the field as a whole progresses towards creating an AI that is capable of creative output on par with a human.

VII. APPENDIX

Article by Tyler Doll:
https://towardsdatascience.com/making-music-with-machine-learning-908ff1b57636