Artificial Intelligence Project Milestone 2: Automatic Music Composition Machine

Kai-Chieh Huang
kaichieh@stanford.edu
Stanford University
Dept. of Electrical Engineering

Quinlan Jung
quinlanj@stanford.edu
Stanford University
Dept. of Computer Science

Jennifer Lu
jenylu@stanford.edu
Stanford University
Dept. of Computer Science

1. MOTIVATION

Music has been composed for centuries, and different genres have emerged over the years. Throughout history, music has for the most part, been composed by humans. Unfortunately, human labor is costly, and composing a piece of music can be a lengthy process. Also, psychological conditions such as the Writer's Block [1] slow the composition process.

In this project, we will leverage different machine learning and mathematical models to generate a piece of music of a certain genre. By creating a model that can reliably generate a piece of music, we lower the cost of creating music by removing the bottleneck of human labor. People who wish to listen to new music or advertising companies who want background music or jingles will be able to do so at an extremely low cost. Furthermore, with an automatic music generator, we can easily generate diverse music material with any duration in a game situation where the play can stay in a particular scene for an undetermined amount of time without looping the same background music again and again.

Also, we hope to generate music that can serve as inspiration to different human composers trying to make their own songs. Fostering creativity in humans has been a well researched subject, but there is no golden standard in place to improve it. However, there is general consensus that humans draw upon experiences and concepts they have previously been exposed to in order to generate novel work. With a reliable music generator, we hope to improve creativity by exposing humans to different compositional variations that they may not have seen before. Work by Nickerson et al [2] suggests that exposing humans to pieces of music composed in novel ways can improve creativity.

2. RELATED WORK

There have been some past works to create music using neural networks. To begin with, Daniel Johnson's "Composing music with recurrent neural networks" [3] describes how he used a model he calls a "biaxial RNN". This structure has two axes - one for time and another for the note. There is also a psuedo-axis for the direction-of-computation. This allows patterns in both time and in note space without giving up invariance. His results were overall pretty positive however there are some defects such as repetition of the same tune for a long period of time.

"Music Composition with Recurrent Neural Network" [4] describes a framework for resilient propagation(RProp) and long short term memory (LSTM) recurrent neural network in order to compose computer music. The LSTM network is able to learn the characteristics of music pieces and recreate music. RProp is also able to predict existing music quicker than Back propagation. Some downsides to the paper were a lack of accurate evaluation for what is considered a good music piece. The evaluation method they used did not always match with a human perspective. Our approach in contrast will most likely use human evaluation and judgement for what makes a good music piece. They also mention that adding dynamics such as soft to loud is something that was lacking in their results.

Another paper "DeepHear - Composing and harmonizing music with neural networks" [5] trains a Deep Belief network on ragtime music to create harmonies to given melodies. This paper took an interesting twist with focusing on creating accompanying music. Although it seems that RNN's may perform better we may draw on inspiration with how the Deep Belief Net works with harmonies.

3. METHODOLOGY

Several attempts have been made to build a automatic music generator with little achievement. Among previous approaches, neural network is most often proposed to tackle this problem due to its similarity in modeling human brain process. It also has an advantage in predicting possible outcomes and pattern recognition. If we were to formulate the automatic music composition problem, it is equivalent to a prediction problem where the model tries to predict the note that should be played at time t given the notes played before.

However, the normal multi-layered feed-forward neural network used in prediction or pattern recognition is limited by its ability to capture the overall rhythmic pattern and music structure since it does not have a mechanism to keep track of the notes played in the past. On the other hand, the Recurrent Neural Network (RNN) with Long Short Term Memory (LSTM) is an ideal model for this task as the feedback connections in RNN enable it to maintain an internal state to keep track of temporal relationship of the inputs and learn the short-term and long-term dependencies in a music piece [4].

Since much research has been done for training RNNs on text to generate paragraphs similar to the training data,

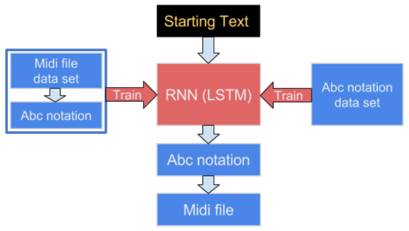

Figure 1: Overview of music generator pipeline

we will leverage this work by transforming the automatic music generation problem into a text learning setup. More specifically, we will use the abc notation[6], a text-based note format to represent each music piece and train our RNN model to generate similar text music in abc notation format. Finally, we can convert the automatic generated text music into midi file. The overall project pipeline is shown in figure 1. We will use the open source library [7] for our RNN model training. After training the RNN model, we can send in some starting text and then feed the output back recursively to generate a piece of music.

4. DATA ACQUISITION

From our discussion in the previous sections, we've decided to train our RNN model on music text notations. Thus, to generate the training data set, we plan on downloading music written in abc notation from the abc notation database[8], where they have over 1000 songs in the text format. Furthermore, we will also leverage the abundant resources in midi music representation by converting midi files into abc notation using [9]. Some midi music libraries examples is provided as in [10] and [11].

5. BASELINE MODEL

The training files were originally in a midi format - a file carrying event messages that specify notation, pitch and velocity. In order to process each midi file to be ingested by our baseline model, we convert it to abc format, a text-based music notation system used as the de facto standard for folk and traditional music.

In each abc file, a 'token' is a small set of notes delimited by whitespaces. A token usually consists of a single note, a set of notes connected by a beam, or a rest.

For our training set, we use 50 Baroque English Suite pieces, composed by J.S. Bach. We then generate a weighted probability vector of each token and its frequency. Our baseline model outputs a piece consisting of 100 tokens. To generate a single token, we use our vector to create weighted probabilities in order to choose a random token. We repeat this process until we have chosen 100 tokens.

A piece's key and time signature is also chosen using a weighted probability vector generated at training time.

While the generated music pieces are syntactically correct, they sound very different from the music seen during training. There is a more contemporary/aharmonic theme present in our baseline music, a completely different type of sound than was heard in the work J.S. Bach ever composed.

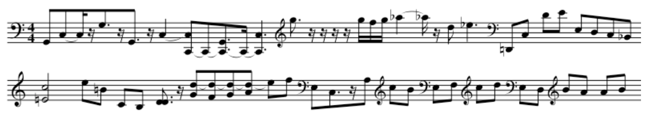

Figure 2: A music sample composed by our baseline implementation

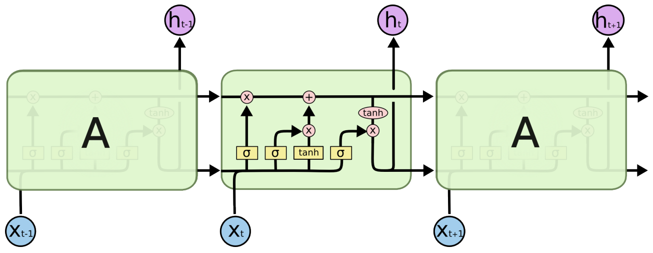

Figure 3: The design of our LSTM RNN

We expand more on the evaluation of our baseline model in the Evaluation section.

6. NEURAL NETWORK MODEL

In our baseline model, we used a probability model to generate notes. However, there is no information incorporated in the model that accounts for the sequence relationships between notes. While other attempts have been made to tackle algorithmic music generation (ie) Markov models and genetic algorithms, LSTM RNNs have been deemed the most successful. [13] Thus, we use the RNN model with LTSM to improve our results.

Our implementation is inspired by Andrej Karpathy's char-rnn, which was originally used to generate Shakespeare-like prose. [14] For our input, we concatenated all our abc notated training songs into a large text file. Then, we train the RNN model on this input text file by feeding it into our network one character at a time per timestep. To obtain a piece of music, sample our neural network, seeding with the beginning header of an ABC file, 'X'. The output of the neural network (with enough samples) is a stream of whole music pieces, also in ABC text format.

Our setup of the neural network is shown in Fig. 3. We leverage Tensorflow to build a MultiRnnCell comprising of 2 128-unit LSTM cells. We use the default *tanh* activation function. We set the learning rate = 0.002 and use a per-parameter adaptive learning rate method, RMSProp, with decay rate = 0.97. [15] These parameters were taken from Manuel Araoz's neural network that composed folk music. [16]

We ran our neural network on Amazon's EC2 g2.2xlarge instance, a machine optimized for graphics-intensive applications with 1 NVIDIA GPU (1,536 CUDA cores and 4GB of video memory).

7. INITIAL RESULTS

In the beginning, we used 1000 classical songs for our training data. Each song's abc notation contained both the melody and chords that played at the same time. After training our model, the result was not satisfying because the model did not learn the relationship between chords and melody well enough.

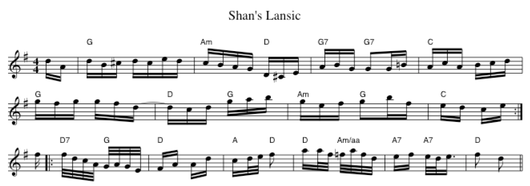

Figure 4: A music sample composed by our RNN-LSTM implementation

One possible explanation is that in the abc notation, it represents the entire melody line of the song first, with the chords that accompany the melody on the next line. Since the RNN model can only relate texts that are close to each other in a certain range, the RNN model can lose track of the relationship of the melody and chord that is supposed to be played in the same bar if the text file represents them a line apart.

In response to this observation, we decided to take a step back and train our model with 1000 single melody folk songs without the chords to see if we could get a better result. With the single melody training experiment, the result was better, as the RNN model was capable of generating arpeggios (broken chords) for the melody, along with random titles for the songs. An example of the resulting music score is shown in figure 4 and a demo sound file is presented in [18].

For the next step, we plan to modify the model so that it can learn the relationship between melodies and chords. More specifically, we propose to separate the melody line and chords in the music text representation for each song and use two RNN models, one for training the melody line as before, the other to train the relationship between melody and chords.

In the proposed model, the input to the first RNN model and the output target are both the melody as before, while the input to the second RNN model will be the melody line and the chord from the prior timestep. Its target output will be the corresponding chords. After we train the model, we can generate random melodies through the first RNN model and feed the randomly generated melody into the second RNN model to produce corresponding chords. Some other desirable optimizations are describe in the next section.

8. FUTURE OPTIMIZATIONS

We plan to add some additional optimizations to improve creating harmonies and not only melodies. As mentioned in our Related Work section, we will try looking at Daniel Johnson's "Composing music with recurrent neural networks" [3] where he is using a model called the "biaxial RNN". The "biaxial RNN" is able to deal with both time and creating nice chords by using two axes along with a psuedo-axis for the direction-of-computation This structure allows for patterns in both time and note space without giving up invariance.

Another thing we can improve upon is the parameters we are using for decay and learning. We can try iterating through a range of parameters to narrow down and find the best one to use. "Tuning a Neural Network for Harmonizing Melodies in Real-Time" [19] talks about such a method called wide-search for decay parameters where they try out different pairs ranging from 0 to 1. Each time it will update the rule for chords and melodies depending on the values that do best.

We will also explore ways to tune all the other parameters in our neural network, such as the number of hidden layers and the number of neurons in each layer. These features change upon the specific application, and there doesnt seem to be any hard and fast rule for choosing these parameters. We will start by applying genetic algorithms to find the optimum combination of effective factors. [17]

9. ORACLE

We use Google Magenta as our oracle to represent as the gold standard. Developed by Google Brain, Magenta is a deep learning project that creates compelling art and music. We leverage Magenta's recurrent neural network and train it with the files we used on our models. Then, we use their MIDI interface to generate Magenta's stream of music. To generate some music we run something like the following command to generate a melody starting with a middle C:

```
melody_rnn_generate \
--config=${CONFIG} \
--bundle_file=${BUNDLE_PATH} \
--output_dir=/tmp/melody_rnn/generated \
--num_outputs=10 \
--num_steps=128 \
--primer_melody="[60]"
```

During our evaluation phase, we expect our test subject to be much more likely to choose the Google Magenta's music over our generated music.

10. GOAL

Our goal can be divided into three milestones: first, is to generate music pieces that sounds is structurally correct (ie) respects beats in time signatures, second, is to create music for a targeted music genre, and third to train our music generator model to produce music styles from different composers. More specifically, we will input a large amount of music of a specific genre in abc notation as training data for our model and output a midi file of that specified genre. We hope to create new midi files that will be close to indistinguishable from human created music.

11. EVALUATION METRICS

The evaluation metrics of our project consists of two parts. First of all, we will utilize a state-of-the-art music genre classifier to verify if the music composed from our model is in the targeted genre. Secondly, we will conduct a double blind A/B test by having a test subject listen to two pieces of music: music composed by a human, and music composed by our generator. In each trial of the test, We will ask the listener to choose which one to label as composed by a human. If our test subjects can only labeled the music generated by human correctly about 50% of the time, we can conclude that the music generated from our generator is indistinguishable to human and is successful.

12. REFERENCES

[1] Clark, Irene. "Invention." Concepts in Composition: Theory and Practice in the Teaching of Writing. 2nd ed. New York: Routledge, 2012. Print.

[2] Nickerson, R. S. (1999). "Enhancing creativity". In R. J. Sternberg. Handbook of Creativity. Cambridge University Press.

[3] http://www.hexahedria.com/2015/08/03/composing-music-with-recurrent-neural-networks/

[4] I-Ting Liu, Bhiksha Ramakrishnan, "Music Composition with Recurrent Neural Network", Carnegie Mellon University, 2015.

[5] http://web.mit.edu/felixsun/www/neural-music.html

[6] http://abcnotation.com/

[7] https://github.com/sherjilozair/char-rnn-tensorflow

[8] http://abc.sourceforge.net/NMD/

[9] http://abc.sourceforge.net/abcMIDI/

[10] https://freemidi.org/

[11] http://www.piano-midi.de/

[12] http://freemusicarchive.org/

[13] J. D. Fernndez and F. J. Vico. 2013. AI methods in algorithmic composition: A comprehensive survey. Journal of Artificial Intelligence Research 48 (2013), 513–582.

[14] https://github.com/karpathy/char-rnn

[15] http://www.cs.toronto.edu/ tij-men/csc321/slides/lecture_slides_lec6.pdf

[16] https://maraoz.com/2016/02/02/abc-rnn/

[17] M. Bashiri, a. Farshbaf Geranmayeh Tuning the parameters of an artificial neural network using central composite design and genetic algorithm Sci Iran, 18 (6) (2011), pp. 16001608

[18] goo.gl/jQOfvO

[19] Gang, Dan, D. Lehman, and Naftali Wagner. "Tuning a neural network for harmonizing melodies in real-time." Proceedings of the International Computer Music Conference, Ann Arbor, Michigan. 1998.