Algorithmic Music Composition using Recurrent Neural Networking

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 paper, we 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 of any duration. This is useful in a game situation where the player 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 generally positive, with the exception of 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 backpropagation. 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 use human evaluation and judgement for what makes a good music piece. They also mention that adding dynamics such as soft to loud transitions 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 by focusing on creating accompanying music. Although it seems that RNN's may perform better, we may draw inspiration from how the Deep Belief Net works with harmonies.

3. GOAL

Our goal can be divided into three points. First, we want to generate music pieces that sound structurally correct (i.e.) respects beats in time signatures. Second, we want to create music for a targeted music genre. Thirdly, we want 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.

4. METHODOLOGY

Several attempts have been made to build an automatic music generator with little achievement. Among previous approaches, neural networks are most often proposed to tackle this problem due to its similarity in modeling the 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.

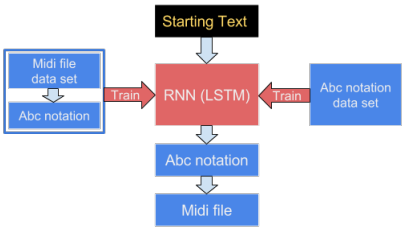

Figure 1: Overview of music generator pipeline

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, we will leverage this work by transforming the automatic music generation problem into a text learning setup. More specifically, we will use 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. Finally, we can convert the automatic generated text music into midi files. 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.

5. DATA ACQUISITION

From our discussion in the previous sections, we have decided to train our RNN model on music text notations. Thus, to generate the training data set, we downloaded 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 are provided in [10] and [11].

6. BASELINE MODEL

The training files were originally in 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 white spaces. A token usually consists of a single note, a set of notes connected by a beam, or a rest.

For our initial training set, we use 50 Baroque English Suite pieces, composed by J.S. Bach. We then generate a

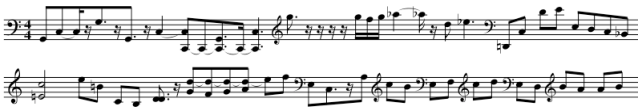

Figure 2: A music sample composed by our baseline implementation

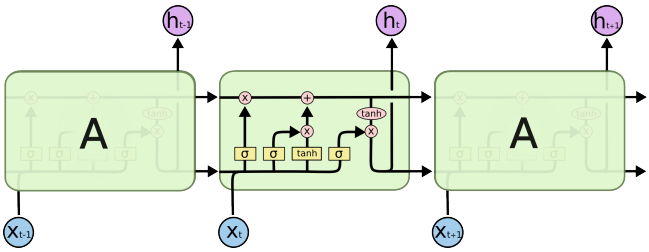

Figure 3: The design of our LSTM RNN

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.

7. 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, we 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 well-formed music pieces, also in ABC text format.

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).

7.1 Basic Neural Network

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

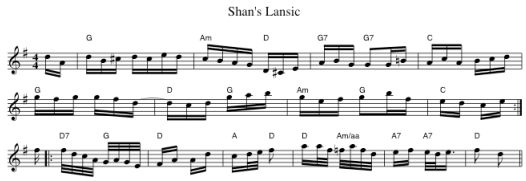

Figure 4: A music sample composed by our RNN-LSTM implementation

function. We set the learning rate = 0.002 and use a per-parameter adaptive learning rate method, RMSProp, with decay rate = 0.97. [15] Characters were fed into the neural net in 2500 size batches for 50 epochs. These parameters were taken from Manuel Araoz's neural network that composed folk music. [16]

We initially trained our basic neural network on 1000 pieces of traditional folk music, a genre that only comprises of a basic melody. From our experiment (discussed in detail in the evaluation section), the generated music we obtained was largely indistinguishable to music composed by a human. 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 [19].

7.2 Challenges With Basic Neural Network

Our basic neural network started falling short when we trained it on Bach's music, work that comprised of both melody and harmony. Our Bach training set comprised of 150 harpsichord and chamber works. Despite our training set being of equal size (in total bytes) to our folk music set, our neural network only produced a melody. In addition, the melody sounded nothing like Bach's work. The result was unsatisfying because the model did not learn the relationship between chords and melody well enough.

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.

7.3 Naive Interleaving Approach

We started with a naive approach to generate a song that had both melody and harmony in it. Going off of our hypothesis that the neural network had trouble generating both a melody and harmony because abc-notation did not represent them in close proximity, we decided to split them up by bar and interleave them.

Then, we fed our interleaved songs into our char-by-char neural net, leaving the parameters unchanged from our basic neural net. We took the generated output and separated every other bar into the melody and harmony. The results were slightly better than our basic neural network because there were small portions of the music that sounded harmonious. However, the naive implementation suffered from asymmetric bars as the summation of time in the notes and rests were not equal between melody and harmony. Also, the music was largely aharmonic.

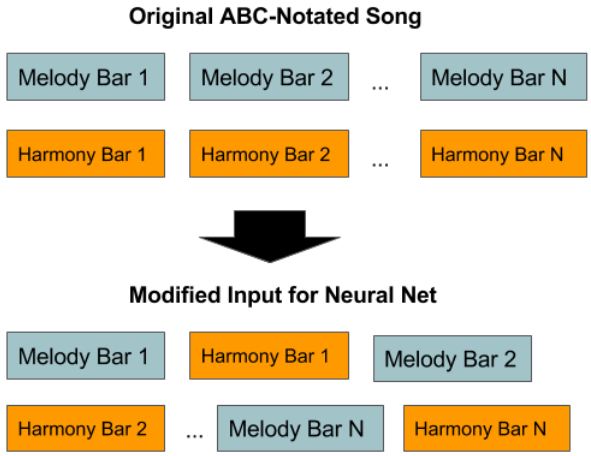

Figure 5: In our naive interleaving implementation, the original abc-text format is modified for input to our neural net. Also, the neural net's output is assumed to be in interleaved form, which we revert back to abc-text format when sampling music.

7.4 Dual Input Approach

In the Related Works section, we mentioned the biaxial RNN that has generated impressive results. The outputted music generally obeys the rules of classical music composition, and sounds mostly harmonious with the exception of prolonged repetition at some parts. The biaxial RNN has an LSTM for each musical note, and for each timestep, t, it gets its output at $t-1$ in addition to the outputs of all the adjacent note LSTM's +/- 1 octave away. Due to the time constraints of the course, we've decided not to implement the biaxial RNN. Instead, we've invented a simpler approach inspired by the biaxial method, which we call the 'Dual Input Approach'.

In our proposed model, we have 2 LSTM's, one for the melody stream and one for the harmony. Instead of our original char-by-char network, we use a bar-by-bar network. The song is chunked by bars, with the melody and harmony LSTM's receiving their respective bars. At each timestep, t, each neural network gets input from the bar produced by the other network and the bar it produced at $t-1$. We decided to implement a bar-by-bar network instead of char-by-char because we wanted a neural net's input to contain notes played in close temporal proximity. Because each bar can have a variable number of chars with the melody stream generally having more chars per bar, a char-by-char implementation does not guarantee the char inputs are in close temporal proximity of the char to be outputted.

In addition to chunking the bars, we also chunk each header. Each header field is fed in its entirety to both the LSTM's before they receive their bars. After we train the model, we seed the LSTM's with the initial header, 'X: 1'. Then, we concatenate the two melody and harmony outputs. Since there can only be one set of headers, the melody's header takes precedent over the harmony's if they differ.

Since we are using a bar-by-bar implementation, the output we got from our network using parameters from our original char-by-char network was not coherent. We changed the batch and sequence size, which determines the size of the

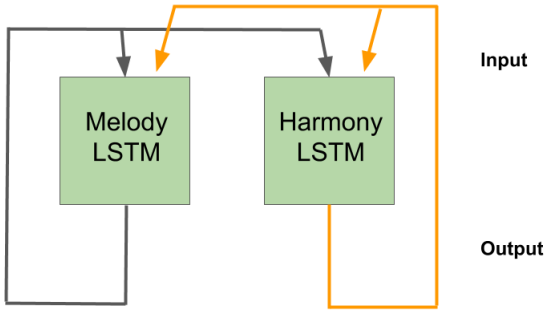

Figure 6: Neural Network architecture in the dual input approach.

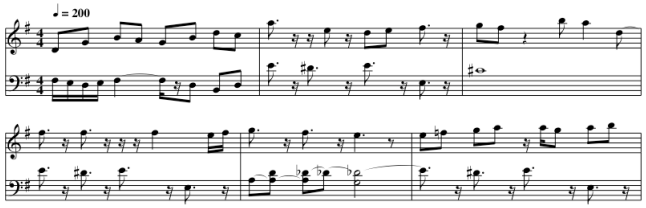

Figure 7: Harmony and melody composed by our dual input neural network trained on Bach's chamber music.

chunk that is fed into the network as input. Using binary search, we discovered that a batch size of 250 bars produces optimal results.

8. ORACLE

We use Google Magenta as our oracle to represent a state-of-the-art implementation. Developed by Google Brain, Magenta is a deep learning project that creates compelling art and music. While Google's WaveNet also produces impressive results, we decided to use Magenta instead because WaveNet takes sound waves as input to its neural net, whereas Magenta uses text input to generate music. Since our project pipeline is more comparable to Magenta's, we have decided to use it as our oracle.

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 create a melody starting in 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]"
```

9. EVALUATION

The evaluation metric of our project consists of having the test subject listen to multiple pieces of music, either composed by a human or a machine. In each trial of the test, the listener is asked to choose whether or not a song was composed by a human on a scale of 0 to 5, where 0 is cannot possibly be a human and 5 is definitely human. If our test subjects label our music as 'more human' than the actual human-created music, we can conclude that the music generated from our neural network is indistinguishable to a human and is successful.

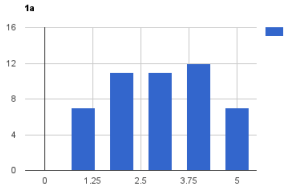

Figure 8: 1a Scores for Our Basic Neural Network Folk Song

We used Amazon's Mechanical Turk platform and our fellow peers to get ratings among 7 different songs split between folk and classical. Question 1 included three different versions of folk music. The first was created by our basic neural network, the second was by a human, and the third was by Google's Magenta. Question 2 had three different versions of classical music where the first was our dual input neural network, the second was human (Bach), and the third was our naive interleaved neural network. Our last question, which we did not show below, was a filter question to make sure that our reviewers were actually listening to the songs.

For our evaluation we received a total of 71 responses where 41 were Mechanical Turks and 30 were our peers. After filtering out with our last question to see which were credible responses, we had a total of 52 responses, of which 27 were Mechanical Turks and 25 were our peers.

	Base NN (1a)	Human (1b)	Magenta (1c)
Average Score	3.02	2.15	**3.27**
Standard Dev	1.30	1.50	1.32

Table 1: Folk Music Ratings.

	Dual Input NN (2a)	Human (2b)	Naive NN (2c)
Average Score	2.85	**3.65**	1.98
Standard Dev	1.57	1.51	1.49

Table 2: Classical Music Ratings.

10. RESULTS AND DISCUSSION

In our first table, among folk music, Google's Magenta was rated the most human, followed by our model, and finally the real human. Figure 5 shows the distribution of scores for our basic neural network produced song which is quite spread out across. This suggests that most people seem to be unsure about whether or not our composition is human with scores appearing to be around 3. Compared with Magenta in Figure 7, the scores are leaning more towards the right, around 3.27.

With classical music, the human composer (J.S. Bach) was rated the most human, followed by the dual input neural

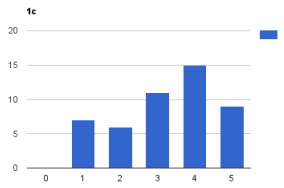

Figure 9: 1b Scores for Google Magenta Folk Song

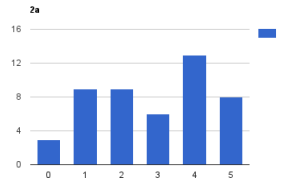

Figure 10: 2a Scores for Our Dual Input Neural Network Classical Song

net, and finally our naive interleaved neural net. The results from the music generated by our dual input neural network suggests that people were split between the two extremes of either 'very much a machine' or 'very much a human composition'. We can see this in Figure 9, that the results give a bi-modal shape. Although we did not perform as well as the actual human who created the musical composition, it is worthwhile to note that at least half of the reviews strongly felt that our composition was human-made.

For reference of our experiment and results, the survey and samples of our music can also be heard at [18] and [19] respectively.

11. FUTURE WORK

We plan to add some additional optimizations to improve creating harmonies and not only melodies. As mentioned in our Related Work section, we will look into improving Daniel Johnson's biaxial RNN. Johnson's RNN is able to deal with both time and creating nice chords by using two axes along with a pseudo-axis for the direction-of-computation. This structure allows for patterns in both time and note space without giving up invariance.

We can also improve upon the parameters we are using for decay and learning. A potential approach is to iterate 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' [20] 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]

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/ tijmen/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] https://goo.gl/forms/p66BeL5ZYqBf1s3B3

[19] goo.gl/jQOfvO

[20] 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.