Classic Monophonic Melody Generation with LSTM and Reinforcement Learning

Honghao Qiu (honq), Chi Wang (chiwang)

Abstract – Our project targets to use LSTM and Reinforcement Learning to build a deep generative model that can create monophonic music melodies. To achieve this, in the first step, we feed music training samples to train a LSTM model to predict sequential notes and chords, and then use reinforcement learning (Q learning) to generate music melody by combining LSTM note prediction and rules of music theory, so the music we generated not only keep the knowledge we learned in LSTM model, but also consistent with music theory.
Keywords: Music Generation, LSTM, Reinforcement Learning, Q Learning, Deep Generative Models

I. Introduction

Our project aims at producing AI generated music that is smooth and closely resembles music style of the samples. We plan to use different ML methods to approach the music generation problem, such as LSTM to get close to state-of-art result, and using Reinforcement Learning to improve the performance by considering music rule alignment as rewards. We expect the music generated with Q-learning enhancement will be more structured, fluent and less repeat notes.

The problem motivation here is obvious - music composition has long been considered unique ability of human and requires high level creativity, it is such a key component of music industry that if we can build AI to generate high quality music, it can assist music composer/producers greatly and boost the entire music industry. There are many inherent challenges in tackling this problem, researchers have been attempting this and achieved remarkable results in recent years using sequence (LSTM, GRU) and generative models (GAN,VAE). However, AI music composition remains to be challenging since it is hard to create music that aligns well with music theory. Motivated by this problem and inspired by Google Magenta Project, we use RL to provide feedback (based on music rules) to LSTM network for better music generation result.

II. Task Definition

Our melody generation system targets at taking classic monophonic melody files as input and outputting two AI generated monophonic melodies separately, using a trained LSTM music generation AI and a RL based music generation AI. By creating AI that learns to generate monophonic melodies, we can help music artist compose music easier by providing inspirational music melodies or auto-complete entire melody based on some starting notes/chords.

In order to measure success of the system, we collected 100 output samples from each of the LSTM and RL music generator trained on 370 input melody midi files, and designed two sets of evaluation metrics: the first set of metrics are music evaluation metrics based on music rules (such as repeated notes, notes not in key, notes in motif, and notes in repeated motif, etc.), and another metrics called Turing test score based on human evaluation by having people guessing whether the music is composed by AI or human.

III. Literature Review

There has been many attempts to generate music melodies based on AI, where many neural based approaches have been proposed.

Due to the temporal structure of music melodies, LSTM seems to be good at capturing this complexity to a high extent. There is a number of LSTM architectures that try to tackle music generation problem such as multi-layered LSTM and highway network cell. Some early approaches exploiting Recurrent Neural Networks (RNNs) are Bharucha Todd (1989, Mozer (1996), Chen Miikkulainen (2001) and Eck Schmidhuber (2002)), using single track melody and produce one note at a time. Later, Boulanger-lewandowski et al. (2012) created RNNs that generates simultaneous independent melodies. On the other hand, real-time music generation is also explored for gaming (Engels et al (2015).

Some other early approaches try to incorporate music theory prior knowledge in melody generation, by using rules of how music segments can be stitched together in a plausible way, e.g., Chan et al. (2006). Mozer (1996) used a RNN that produced pitch, duration and chord at each time step. Unlike most other neural network approaches, this work encodes music knowledge into the representation. Historically, Eck Schmidhuber (2002) first introduced LSTM approach in music melody note and chord generation. LSTM captured more global music structure compared to Mozer's work (1996). Boulanger-lewandowski

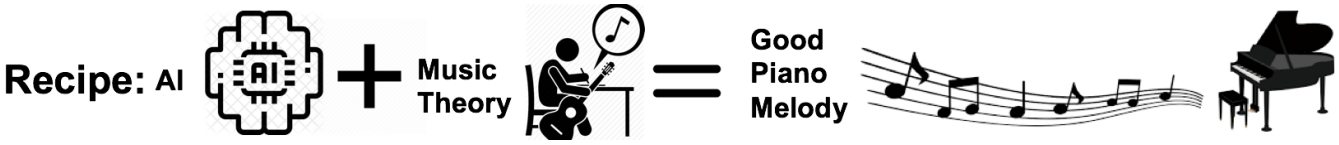

Figure 1: Melody Generation Recipe

et al (2012) propose to learn complex musical structure which has multiple notes and chords at the same time. The model is single-track and only produces melody. More recently, Huang Wu (2016) proposed a 2-layer LSTM which is able to produce music melody that is more complex than a single note sequence with chords.
Some other powerful tool proved to be effective for music generation is variational auto-encoder (VAE). Based on assumptions about latent variables distribution, VAE uses variational approach for latent representation learning. This leads to an additional loss component and the Stochastic Gradient Variational Bayes (SGVB) algorithm for training. VAE-based generative model can generate realistic music examples which are similar to the samples drawn from input data distribution.
More recently, Google Brain Magenta Team has been applying RNN along with Deep Reinforcement Learning approach to tackle music and art generation problem. They incorporate music theory into the RL tuner reward function along with LSTM entrophy reward to adjust the trained LSTM model weights, and is capable of generating music melody is smooth and more consistent with rules.
We believe that LSTM and RL approaches are complementary to each other and tries to tackle classic monophonic melody generation with similar approach. In this paper, we aim to generate classic monophonic piano music, where the melody contains both notes and chords. We draw many of the inspiration from prior arts in LSTM music generation and Google Magenta project.

IV. Infrastructure

We collected classic monophonic piano melodies midi file dataset for training. In LSTM modeling, we run the model on local laptop (Mac) for about a day. In RL modeling, we use GCP to run RL tuner faster and creates feedback providing agent (i.e. music feedback reward functions defined based on music rule alignments).
We would discuss the system in details in the following sections.

V. Approach

We use two step learning approach to generate music melody:
1. Train a LSTM network as base model to generate note sequence.
2. Do deep reinforcement learning (DQN) on rewards of LSTM prediction and music theory alignments.

For LSTM: LSTM network architecture is illustrated as follows: 3 LSTM layers each with 1024 hidden nodes connected to 3 dropout layers with 40% drop out ratio, followed by a Fully Connected dense layer, and then output to Softmax activation layer for prediction (next node/chord likelihood).

For Reinforcement Learning (DQN): Train DQN model consists of Q network and target Q network both initiated with trained LSTM model parameters, and use adjusted reward based on log likelihood from LSTM and music theory alignment score, gradient descent on DQN loss to improve melody generation.

There are two problems need to be solved in order to produce a pleasing music:
The first challenge is creating a long-term structure capable of capturing temporal relationships across chords. Long-term structure comes very naturally to people, but it's very hard for machines. Basic machine learning systems can generate a short melody that stays in key, but they have trouble generating a longer melody that follows a chord progression, or follows a multi-bar song structure of verses and choruses. We will try to use LSTM to solve it.
The second problem is to let music produced by the model follow certain music rules and form (ex, avoid excessively repeated notes, stay in key, etc), and use them as feedback to improve generated music so it aligns with musical theory better. We plan to tackle this by defining some rules, and then convert this to Q-learning reward function and use it in the RL model training.

Before addressing the music generation problem with LSTM and Reinforcement Learning, we firstly got the following baseline and oracle results:
Baseline: a music generated by top 10 most frequent notes in random sequence, (and we also try plain RNN without memory as we expect the performance to be poor without memory). Survey shows that the former has Turing test score is close to 0 while plain RNN has Turing test score about 1-2 on average.
Oracle: a music generated by mixing original music snippet (and we also try state of art LSTM as a proxy since Oracle

in this case is hard to define due to the subjective nature of the problem.) Evaluation metric from generated music is close to the training music and user study shows 80% people like the music and think it's classical style. The gap of baseline and oracle mainly comes from the ability to capture memory across melody and degree of alignment with music rules. We expect to get significantly better than baseline result and close to oracle result with our mixed approach.

The following sections describe our modeling approach in details.

VI. Dataset and Preparation

A. Data Collection and Cleaning

We are currently using 370 midi files from classical piano midi page (http://www.piano-midi.de/) written by a range of composers including Bach, Mozart, Schubert, etc. We pick this piano music source due to the very distinct and beautiful melodies that the majority of the pieces have and the sheer amount of pieces that exist. Each file is classic monophonic melody that is approximately 3 minutes in length. We first load the music midi file with music21 package in python, and truncate each file to 30 seconds' music melody as our training samples.

B. Note and Chord Extraction

According to the MIT music21 project introduction (http://web.mit.edu/music21/), notes and chords are the key to music melody, and we need to firstly process the training sample to extract note and chord information from music data. Note consists of three key pieces of information - pitch, octave, and offset. a Note stores a single note (that is, not a rest or an unpitched element) that can be represented by one or more notational units – so for instance a C quarter-note and a D eighth-tied-to-32nd are both a single Note object. A Note knows both its total duration and how to express itself as a set of tied notes of different lengths. For instance, a note of 2.5 quarters in length could be half tied to eighth or dotted quarter tied to quarter. And Chord is a set of notes played at the same time. Here are the example note and chord extracted and printed out from our input music sample:
<music21.note.Note C> 86.0
<music21.chord.Chord E3 A3> 86.5
Note that the last number associated to the extracted note and chord is offset, which gives us idea on what's the time interval between notes and chords.

VII. Music Generation with LSTM

A. Data and Processing

To train the model, We firstly need to prepare the possible range of notes (and chords) that our model can output to, by putting all the notes and chords into a sequential list, and the set of distinct notes and chords (in this case the number is 432) is our target output range. For the input sequence of notes and chords, we perform a transformation to map it from string-based categorical data to integer-based numerical data. For example, [C(note) A2(note) B3(note) A2(note) A2+B3(chord)] becomes [0 1 2 1 3].
We also need to process the data to create 'label' to be used for training. We prepare the output as follows: the model would predict the next note/chord in the sequence with the previous 40 notes/chords. We expect to tune this hyper-parameter going forward to see how we can generate better piano melody.
Finally, we standardize input by normalization and make the output label one-hot encoded.

B. Model and Network Architecture

We firstly apply LSTM sequence model to music generation. Our model pipeline consists of the following components:
1. Apply LSTM layer, a Recurrent Neural Net (RNN) that takes sequence input and train weights to predict output based on its memory mechanism consists of forget gate, input gate, and update gate to protect and control the cell state.
2. Use Dropout for regularization by setting certain portion (set by a parameter) of input units to 0 at each update during the training to prevent over-fitting.
3. Then we use Fully Connected layers where all input nodes connected to all output nodes.
4. Finally, feed into Softmax layer to predict the output (a node/chord within the list of all possible nodes/chords).
We train the network with 200 epochs and batch size 64. Our network architecture is illustrated as follows: 3 LSTM layers each with 1024 hidden nodes connected to 3 dropout layers with 40% drop out ratio, and a Fully Connected dense layer, then output to Softmax activation layer to make prediction.

C. Loss Function

The Loss function for training the network is (categorical) cross entropy loss:

$$H_y(y) = -\sum_i y_i' log(y_i)$$

where y_i is the predicted probability value for class i and y_i' is the true probability for that class.

Figure 2: Data Processing

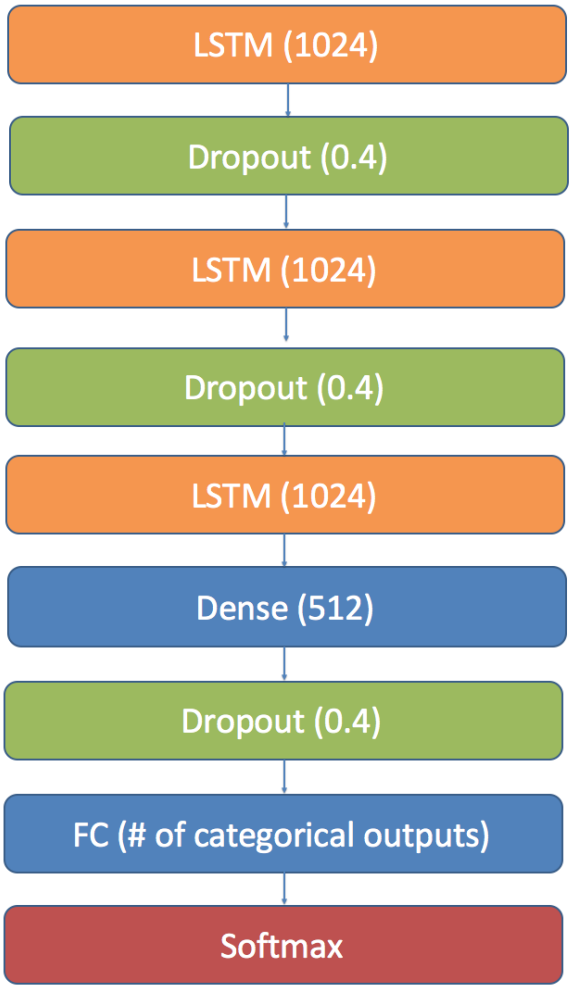

Figure 3: LSTM Network Architecture

D. Prediction/Music Generation

Finally, we can use the trained model to generate 30 second monophonic piano melody, by using the most likely node/chord based on the Softmax layer prediction output, and make prediction sequentially (outputting 125 nodes/chords in total for 30 second music). Note that since we have map node/chord from categorical string to integers and we are ignoring offset in model training, and only predicting nodes and chords, we need to decode the output from integers to notes/chords, and then we insert 0.25 second time interval between each outputted node/chord to fill and create a 30 second music.

E. Experimentation Result

We output 10 music melodies for metric evaluation. The notes/chords repeated has a relatively high ratio (63.3%), and at times there are some weird notes/chords outputted in sequence that does not fit with music theory. We mix the ten generated melodies along with other 10 human performed melodies (by making these both 30 seconds and with 0.25 offset across notes/chords), and give it to 10 of our friends as Turing test to see how many of the generated music sample can pass. The current preliminary result is relatively low (at 20%), this is probably due to the shallow network we are using, and lack of parameter tuning. And more importantly, we did not introduce music rules and theories as reward function to conduct reinforcement learning to tune the network. We expect the performance to be improved later on.

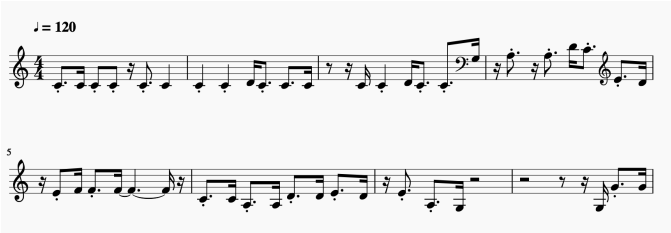

Figure 4: one LSTM generated melody

VIII. Music Generation with Reinforcement Learning

A. Data and Processing

By review those sample melodies, we find these music pieces generated by LSTM model tend to wander and lack musical structure, see figure 5, this is a typical LSTM generated melody which contains lots duplication notes and repeated patterns. To overcome this deficiency, we introduce reinforcement learning to impose structure on RNN trained data.

The input of our reinforcement tuning model is a trained LSTM model which described in previous section, it will produce one note at a time. In order to formulate musical composition as an RL problem, we treat placing the next note in the composition as taking an action.

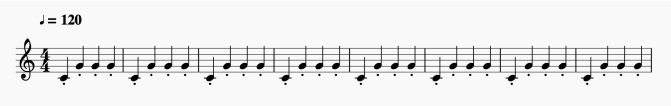

Figure 5: one LSTM generated melody with lots duplicated notes and repeated pattern

B. Reinforcement Learning Tuning Design

Our goal is to compose music which is not only maintaining the information of musical compositions originally learned from data but also follow music theory in certain degree. To accomplish this We propose the following RL Turning system (Figure 6).
To train this Deep Q network, We use a trained LSTM model to supply the initial weights/bias for three networks in RL Tuner: the Q-network and Target Q-network in the DQN algorithm and a Reward RNN. The Reward RNN is held fixed and used to supply part of the reward value used in training.
The state of the environment s consists of the notes placed in the composition so far, Q-network internal state and the Reward RNN. To calculate the reward, we combine probabilities learned from the training data with knowledge of music theory. We define a set of music-theory based rules (described in Section VIII.E) to impose constraints on the melody that the model is composing through a reward signal r_{MT} (a, s).
For example, if a note is in the wrong key, then the model receives a negative reward. However, it is necessary that the model still be "creative," rather than learning a simple composition that can easily exploit these rewards. Therefore, we use the Reward RNN — or equivalently the trained Note RNN to compute $log_p(a|s)$, the log probability of a note a given a composition s, and incorporate this into the reward function. Figure 6 illustrates these ideas.

C. RL Model

The state of the environment s consists of the notes placed in the composition so far, Q-network internal state and the Reward RNN.
We treat choosing next note in composition as an action. Our Q Value function is:

$$Q^{\pi}(s_t, a_t) = \mathbb{E} \parallel \sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'}) \parallel$$

Q Optimal function is:

$$Q_{opt}(s_t, a_t) = r(s_t, a_t) + \gamma \mathop{\mathbb{E}}_{p(s_{t+1}|s_t, a_t)} [\max_{a_t|1} Q(s_{t+1}, a_{t+1})]$$

Reward function at time t is:

$$r(s, a) = logp(a|s) + r_{MT}(a, s)/c$$

where c is a constant controlling the emphasis placed on the music theory reward.

D. Loss Function

Given the DQN loss function below and reward function above, our loss function and learned policy for RL are:

$$L(\theta) = \mathop{\mathbb{E}}_{\beta}[(log(p(a|s)) + r_{MT}(a, s)/c + \sigma \max_{a'} Q(s', a') - Q(s, a))^2]$$

$$\pi_{\theta}(a|s) = \delta(a = \arg\max_{a} Q(s, a))$$

E. Music Theory Based Reward

Currently we defined below three rules in placed:

- All notes should belong to the same key.
- The composition should begin and end with the tonic note of the key.
- Unless a rest is introduced or a note is held, a single tone should not be repeated more than four times in a row.

IX. Experiment and Analysis

A. LSTM Output and Evaluation

Metrics and Turing Test Result Metrics can be found in figure 3. Turing test score is currently 0.2 (passes 20% of the time on average).

B. RL tuning Output and Evaluation

Figure 7 is a sample 128 notes melody generated by RL tuner, We also generate note probability graph (Figure 8 and 9) for LSTM model and RL Tuner.
By comparing the note probability graph of Figure 8 and 9, we could see RL Tuner improve the music composition by choosing notes in a wider range(more diverse) and less repetitive. List the 32 note sequences generated by LSTM and RL Tuner for figure 8 and 9 below:
LSTM: Sample LSTM generated note sequence snippet: 23,16,23,0,1,0,18,0,16,0,18,0,18,0,16,0,18,0,18, 0, 22, 0, 0, 0, 23, 0, 1, 16, 23, 0, 18. (Full melody 30 seconds in length) Turing test score: 2/10 passed.

RL(+LSTM): Sample RL tuner generated note sequence snippet: 0, 1, 23, 0, 23, 26, 16, 23, 25, 0, 26, 0, 16, 1, 30, 0, 23, 26, 3, 33, 0, 28, 26, 0, 0, 21, 16, 0, 26, 0, 3. (Full melody has 30 seconds) Turing test score: 4/10 passed.

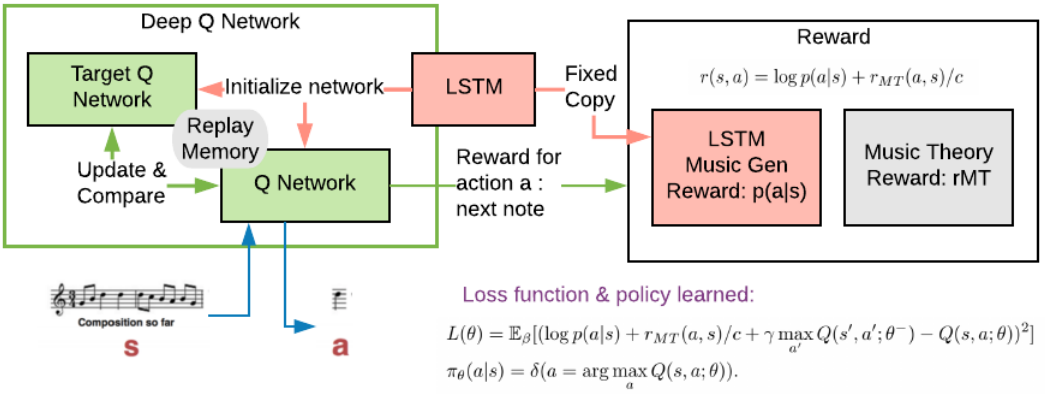

Figure 6: Melody Generation with QDN

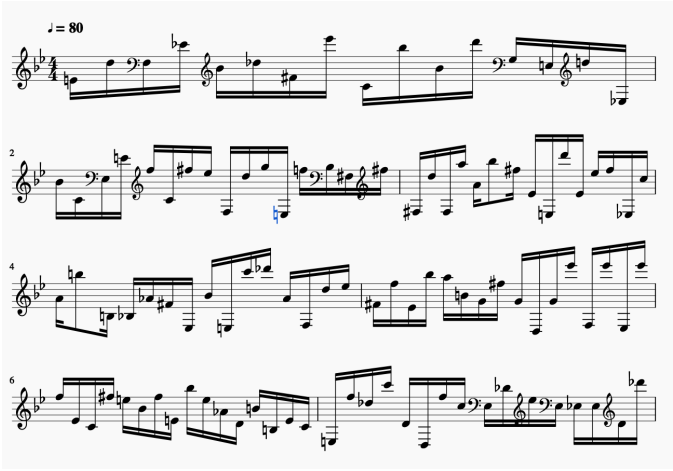

Figure 7: Sample RL Tuner 128 note sequence

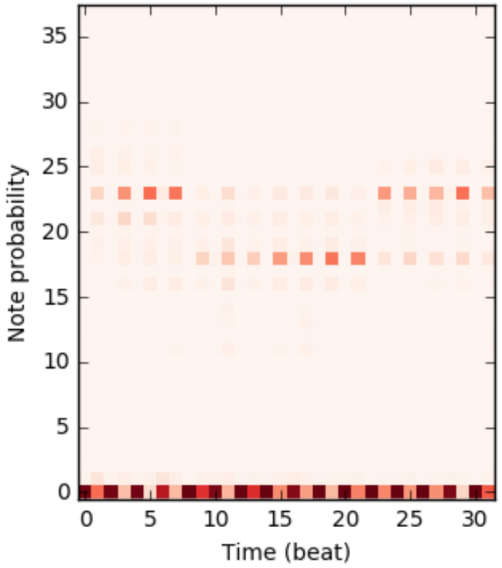

Figure 8: Note probability of LSTM

C. Metric Comparison

Below table provides quantitative results in the form of performance on the music theory rules to which we trained the model to adhere.
This result is based on 10 randomly initialized composition generated by each model.

Metric	LSTM	RL+LSTM
Notes excessively repeated	63.3%	2.5%
Notes not in key	5.0 %	3.50%
Notes in motif	10.2 %	69.50%
Notes in repeated motif	0.0 %	0.62%
Turing test score	0.2	0.4

D. Q Learning Reward

In RL Tuner, we design reward function at time t as : $r(s,a) = logp(a|s) + r_{MT}(a,s)/c$, which combine LSTM note probability and music theory rules. Figure 10 depicts reward value used in RL Tuner training, we could adjust const c to give music theory rule more weight in the reward function.

X. Error Analysis & Discussion

According to the experimentations and evaluation metrics presented above, we find that:
LSTM model is able to generate reasonably good monophonic piano melody based on the input music samples, many of the generated notes and chords are consistent with each other, but still it has many weird patterns in the generated melody that is repetitive but not consistent with music theory, and with less repeated motif which is important to good melody. The Turing test score is also

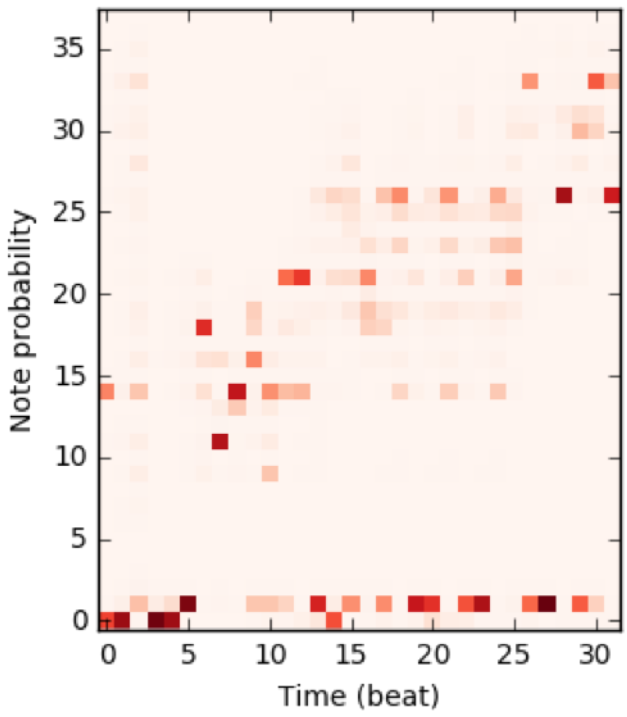

Figure 9: Note probability of LSTM + RL Tunner

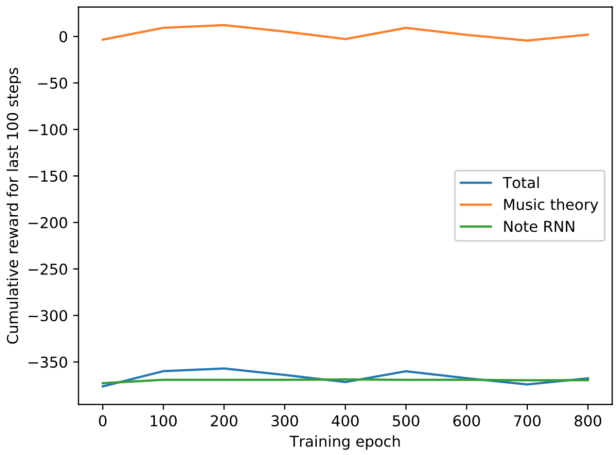

Figure 10: RL Q Training Rewards

relatively low.

Reinforcement learning is able to provide judgment by giving rewards to generated music sample based on how well does it align with the music theory. With DQN adding music theory alignment reward, we improve the generated melody to make it more fluent, more consistent and less repetitive by combing Q learning with trained LSTM model. Turing test score (passed human judgement) is also much higher.

Combining RNN LSTM with RL tuner based on music-rule reward seems to be an effective way of creating music melody that is fluent and consistent with musical theory.

XI. Conclusion

Our experiments have shown that LSTM model is able to generate reasonably good monophonic piano melody based on the input music samples, many of the generated notes and chords are consistent with each other, but still it has many weird patterns in the generated melody that is not consistent with music rule and music theory. However, we found that reinforcement learning is able to provide judgment by giving rewards to generated music sample based on how well does it align with the music theory. Going forward, we can improve the generated melody to make it more fluent, more consistent and less repetitive based on combing Q learning with trained LSTM model.

XII. Future Work

1. Improve LSTM: We could improve LSTM by tuning parameters, changing network architecture such as creating deeper model, input more samples, and using attention mechanisms or other modeling approaches (GRU, etc.).

2. Improve RL: we could improve RL algotithm by using different reward functions (adding more music rules) to get better reward from reinforcement learning based on music rule and music theory alignment to adjust weight and improve the generative model. We could also try different network architectures (DDQN) and perform more hyper-tuning, with larger input dataset.

3. Data clustering: we could make better prediction on each music style/categories by performing K-means clustering to classify music melody by genre/composer, and to learn models that can generate music for the specific category of music.

4. Deep Generative Models: We could also try using Generative Models (e.g. GAN/VAE) based approach to have a discriminator to tell which music is generated vs human-composed, and retrain generative network to create better music sample.

XIII. References

We primarily take references to Google Magenta Project (https://magenta.tensorflow.org), Sequence Model - LSTM on coursera (https://deeplearning.ai), and Reinforcement Learning course at UC Berkeley (http://rll.berkeley.edu/deeprlcourse/).

[1] Google Magenta Project: RL Tuner: https://magenta.tensorflow.org

[2] LONG SHORT-TERM MEMORY, Sepp et al, Neural Computation 9(8): 1997

[3] Medium: How-To-Generate-Music-using-a-Lstm-Neural-Network-in-Keras (https://medium.com/m/signin?redirect=https

[4] Untersuchungen zu dynamischen neuronalen Netzen, Hochreiter

[5] Long Short-Term Memory, Hochreiter, Sepp; Schmidhuber

[6] Long Short-Term Memory in Recurrent Neural Networks, Felix Gers

[7] Supervised Sequence Labelling with Recurrent Neural Networks, Alex Graves

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

[9] http://www.deeplearningbook.org/

[10] Reinforcement Learning: An Introduction, Richard S. Sutton and Andrew G. Barto

[11] I. Arel, "Deep Reinforcement Learning as Foundation for Artificial General Intelligence," in Theoretical Foundations of Artificial General Intelligence, ed: Springer, 2012, pp. 89-102.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, et al., "Playing Atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, et al., "Human-level control through deep reinforcement learning," Nature, vol. 518(7540), pp. 529-533, 2015.

[14] Jamshed J. Bharucha and Peter M. Todd. Modeling the perception of tonal structure with neural nets. Computer Music Journal, 13(4):44–53, 1989.

[15] Nicolas Boulanger-lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In ICML, 2012.

[16] Michael Chan, John Potter, and Emery Schubert. Improving algorithmic music composition with machine learning. In 9th International Conference on Music Perception and Cognition, 2006.

[17] Chun-Chi J. Chen and Risto Miikkulainen. Creating melodies with evolving Douglas Eck and Juergen Schmidhuber. A first look at music composition using lstm recurrent neural networks. 2002.

[18] Steve Engels, Fabian Chan, and Tiffany Tong. Automatic real-time music generation for games. In AIIDE Workshop, 2015.

[19] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. volume 315, pp. 972–976, 2007.

[20] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. In arXiv:1508.06576, 2015.

[21] Allen Huang and Raymond Wu. Deep learning for music. arXiv preprint arXiv:1606.04930, 2016.

[22] Daniel Johnson. Composing music with recurrent neural networks. https://goo.gl/YP9QyR.

[23] Semin Kang, Soo-Yol Ok, and Young-Min Kang. Automatic Music Generation and Machine Learning Based Evaluation, pp. 436–443. Springer Berlin Heidelberg, 2012.

[24] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. In ICLR 2016 Workshop, 2016.

[25] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In NIPS, 2015.

[26] David Macdonald. Song from . https://youtu.be/OMq9he-5HUU.

[27] Reddit midi man. Midi collection. https://goo.gl/4moEZ3.

[28] Felix Sun. Deephear - composing and harmonizing music with neural networks. https://goo.gl/7OTZZL.

[29] Elliot Waite, Douglas Eck, Adam Roberts, and Dan Abolafia. Project magenta. https://magenta.tensorflow.org/.