A Hierarchical Recurrent Neural Network for Symbolic Melody Generation

Jian Wu, Changran Hu, Yulong Wang, Xiaolin Hu, and Jun Zhu

Abstract

In recent years, neural networks have been used to generate symbolic melodies. However, the long-term structure in the melody has posed great difficulty for designing a good model. In this paper, we present a hierarchical recurrent neural network for melody generation, which consists of three Long-Short-Term-Memory (LSTM) subnetworks working in a coarse-to-fine manner along time. Specifically, the three subnetworks generate bar profiles, beat profiles and notes in turn, and the output of the high-level subnetworks are fed into the low-level subnetworks, serving as guidance for generating the finer time-scale melody components in low-level subnetworks. Two human behavior experiments demonstrate the advantage of this structure over the single-layer LSTM which attempts to learn all hidden structures in melodies. Compared with the state-of-the-art models MidiNet (Yang, Chou, and Yang 2017) and MusicVAE (Roberts et al. 2018), the hierarchical recurrent neural network produces better melodies evaluated by humans.

Introduction

Automatic music generation using neural networks has attracted much attention. There are two classes of music generation approaches, symbolic music generation (Hadjeres, Pachet, and Nielsen 2017)(Waite et al. 2016)(Yang, Chou, and Yang 2017) and audio music generation (van den Oord et al.)(Mehri et al. 2016). In this study, we focus on symbolic melody generation, which requires learning from sheet music.

Many music genres such as pop music consist of melody and harmony. Since usually beautiful harmonies can be ensured by using legitimate chord progressions which have been summarized by musicians, we only focus on melody generation, similar to some recent studies (Waite et al. 2016)(Yang, Chou, and Yang 2017)(Colombo, Seeholzer, and Gerstner 2017)(Roberts et al. 2018). This greatly simplifies the melody generation problem.

Melody is a linear succession of musical notes along time. It has both short time scale such as notes and long time scale such as phrases and movements, which makes the melody generation a challenging task. Existing methods generate pitches and rhythm simultaneously (Waite et al. 2016) or sequentially (Chu, Urtasun, and Fidler 2016) using Recurrent Neural Networks (RNNs), but they usually work on the note scale without explicitly modeling the larger time-scale components such as rhythmic patterns. It is difficult for them to learn long-term dependency or structure in melody.

Theoretically, an RNN can learn the temporal structure of any length in the input sequence, but in reality, as the sequence gets longer it is very hard to learn long-term structure. Different RNNs have different learning capability, e.g., LSTM (Hochreiter and Schmidhuber 1997) performs much better than the simple Elman network. But any model has a limit for the length of learnable structure, and this limit depends on the complexity of the sequence to be learned. To enhance the learning capability of an RNN, one approach is to invent a new structure. In this work we take another approach: increase the granularity of the input. Since each symbol in the sequence corresponds to longer segment than the original representation, the same model would learn longer temporal structure.

To implement this idea, we propose a Hierarchical Recurrent Neural Network (HRNN) for learning melody. It consists of three LSTM-based sequence generators — Bar Layer, Beat Layer and Note Layer. The Bar Layer and Beat Layer are trained to generate bar profiles and beat profiles, which are designed to represent the high-level temporal features of melody. The Note Layer is trained to generate melody conditioned on the bar profile sequence and beat profile sequence output by the Bar Layer and Beat Layer. By learning on different time scales, the HRNN can grasp the general regular patterns of human composed melodies in different granularities, and generate melody with realistic long-term structures. This method follows the general idea of granular computing (Bargiela and Pedrycz 2012), in which different resolutions of knowledge or information is extracted and represented for problem solving. With the shorter profile sequences to guide the generation of note sequence, the difficulty of generating note sequence with well-organized structure is alleviated.

Related Work

Melody Generation with Neural Networks

There is a long history of generating melody with RNNs. A recurrent autopredictive connectionist network called CONCERT is used to compose music (Mozer 1994). With a set of composition rules as constraints to evaluate

melodies, an evolving neural network is employed to create melodies (Chen and Miikkulainen 2001). As an important form of RNN, LSTM (Hochreiter and Schmidhuber 1997) is used to capture the global music structure and improve the quality of the generated music (Eck and Schmidhuber 2002). Boulanger-Lewandowski, Bengio, and Vincent explore complex polyphonic music generation with an RNN-RBM model (Boulanger-Lewandowski, Bengio, and Vincent 2012). Lookback RNN and Attention RNN are proposed to tackle the problem of creating melody's long-term structure (Waite et al. 2016). The Lookback RNN introduces a handcrafted lookback feature that makes the model repeat sequences easier while the Attention RNN leverages an attention mechanism to learn longer-term structures. Inspired by convolution, two variants of RNN are employed to attain transposition invariance (Johnson 2017). To model the relation between rhythm and melody flow, a melody is divided into pitch sequence and duration sequence and these two sequences are processed in parallel (Colombo et al. 2016). This approach is further extended in (Colombo, Seeholzer, and Gerstner 2017). A hierarchical VAE is employed to learn the distribution of melody pieces in (Roberts et al. 2018), the decoder of which is similar to our model. The major difference is that the higher layer of its decoder uses the automatically learned representation of bars, while our higher layers use predefined representation of bars and beats which makes the learning problem easier. Generative Adversarial Networks (GANs) have also been used to generate melodies. For example, RNN-based GAN (Mogren 2016) and CNN-based GAN (Yang, Chou, and Yang 2017) are employed to generate melodies, respectively. However, the generated melodies also lack realistic long-term structures.

Some models are proposed to generate multi-track music. A 4-layer LSTM is employed to produce the key, press, chord and drum of pop music seperately (Chu, Urtasun, and Fidler 2016). With pseudo-Gibbs sampling, a model can generate highly convincing chorales in the style of Bach (Colombo, Seeholzer, and Gerstner 2017). Three GANs for symbolic-domain multi-track music generation were proposed (Dong et al. 2018). An end-to-end melody and arrangement generation framework XiaoIce Band was proposed to generate a melody track with accompany tracks with RNN (Zhu et al. 2018).

Hierarchical and Multiple Time Scales Networks

The idea of hierarchical or multiple time scales has been used in neural network design, especially in the area of natural language processing. The Multiple Timescale Recurrent Neural Network (MTRNN) realizes the self-organization of a functional hierarchy with two types of neurons "fast" unit and "slow" unit (Yamashita and Tani 2008). Then it is shown that the MTRNN can acquire the capabilities to recognize, generate, and correct sentences in a hierarchical way: characters grouped into words, and words into sentences (Hinoshita et al. 2011). An LSTM auto-encoder is trained to preserve and reconstruct paragraphs by hierarchically building embeddings of words, sentences and paragraphs (Li, Luong, and Jurafsky 2015). To process inputs at multiple time scales, the Clockwork RNN is proposed, which partitions the hidden layers of RNN into separate modules (Koutnik et al. 2014). Different from the Clockwork RNN, we integrate the prior knowledge of music in constructing the hierarchical model and feed multiple time scales of features to different layers.

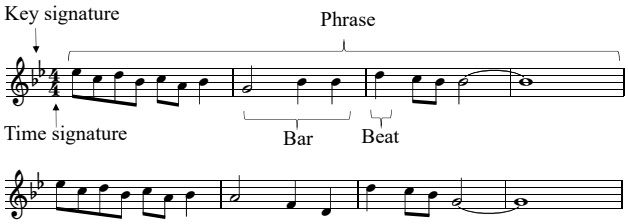

Figure 1: A typical form of melody. The time signature of this musical piece is 4/4. The numerator means a bar contains 4 beats, and the denominator means the time length of 1 beat is a quarter note.

Music Concepts and Representation

We first briefly introduce some basic music concepts and their properties to familiarize the readers who do not have a music background, then explain how the concepts are represented in the model.

Basic Music Concepts

As shown in Fig. 1, melody, often known as tune, voice, or line, is a linear succession of musical notes, and each note represents the pitch and duration of a sound. Several combined notes form a beat that determines the rhythm based on which listeners would tap their fingers when listening to music. A bar contains a certain number of beats in each musical piece. Time signature (e.g., 3/4) specifies which note value is to be given in each beat by the denominator and the number of beats in each bar by the numerator. Each musical piece has a key chosen from 12 notes in an octave. Key signature, such as C♯ or B♭, designates which key the current musical piece is. The musical piece can be transformed to different keys while maintaining the general tone structure. Therefore we can transpose all of the musical pieces to key C, while maintaining the relative relationship between notes. Shifting all musical pieces to the same key makes it easier for the model to learn the relative relationship between notes. The generated pieces can be transposed to any key.

Melody Representation

To simplify the problem, we only chose musical pieces with the time signature 4/4. This is a widely-used time signature. According to the statistics on the Wikifonia dataset described in Section , about 99.83% of notes have pitches between C2 and C5. Thus, all notes are octave-shifted to this range. Then there are 36 options for a pitch of a note (3 octaves and each octave has 12 notes). To represent duration, we use event messages in the Midi standard. When a

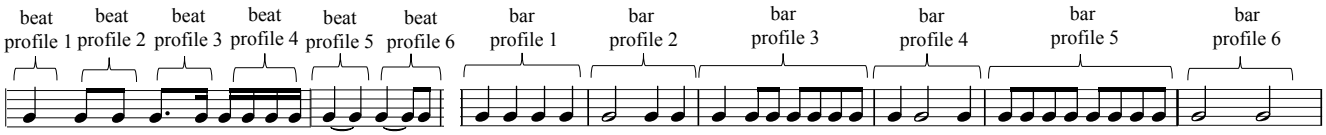

Figure 2: Samples of beat profiles and bar profiles. Here we use notes with same pitch to illustrate rhythm in beat and bar. The rhythm represented by beat profile 5 and 6 are related with the rhythm of the previous beat so they are shown with two beats where the first beats are all quarter notes.

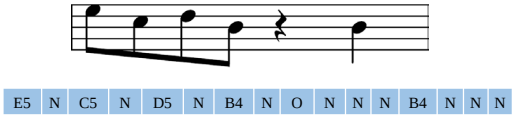

Figure 3: An example of melody representation. **Top**: A melody with the length of one bar. **Bottom**: Representation of the melody, in which the N means a no-event and the O means a note-off event. Since the fourth note is not immediately followed by any note, a note-off event is necessary here.

note is pressed, a note-on event with the corresponding pitch happens; and when the note is released, a note-off event happens. For a monophonic melody, if two notes are adjacent, the note-on event of the latter indicates the note-off event of the former, and the note-off event of the former is therefore not needed. In this study, every bar was discretized into 16 time steps. At every time step, there are 38 kinds of events (36 note-on events, one note-off event and one no-event), which are exclusive. One example is shown in Fig. 3. In this way, note-on events mainly determine the pitches in the melody and no-events mainly determine the rhythm as they determine the duration of the notes. So a 38-dimensional one-hot vector is used to represent the melody at every time step.

Rhythmic Patterns and Profiles

Rhythmic patterns are successions of durations of notes which occur periodically in a musical piece. It is a concept on a larger time scale than the note scale and is important for melodies' long-term structure. Notice that in this model we do not encode the melody flow because it is hard to find an appropriate high-level representation of it.

Two features named *beat profile* and *bar profile* are designed, which are high-level representations of a whole bar and beat, respectively. Compared with individual notes, the two profiles provide coarser representations of the melody. To construct the beat profile set, all melodies are cut into melody clips with a width of one beat and binarized at each time step with 1 for an event (note-on events and note-off event) and 0 for no-event at this step. Then we cluster all these melody clips into several clusters via the K-Means algorithm and use the cluster centers as our beat profiles. Given a one beat melody piece, we can binarize it in the same manner and choose the closest beat profile as its representation. The computation of bar profile is similar, except that the width of melody clip is changed to one bar. Based on the well-known elbow method, the numbers of clusters for beat profiles and bar profiles are set to be 8 and 16 respectively. In Fig. 2, some frequently appeared profiles are shown with notes.

Hierarchical RNN for Melody Generation

Model Architecture

HRNN consists of three event sequence generators: Bar Layer, Beat Layer and Note Layer, as illustrated in Fig. 4. These layers are used to generate bar profile sequence, beat profile sequence and note sequence, respectively.

The lower-level generators generate sequences conditioned on the sequence output by the higher-level generators. So to generate a melody, one needs to first generate a bar profile sequence and a beat profile sequence in turn. Suppose that we want to generate a melody piece with the length of one bar, which is represented as $n_t, ..., n_{t+15}$(see Fig. 4). First, the Bar Layer generates a bar profile B_t with the last bar profile B_{t-16} as input. Then the Beat Layer generates 4 beat profiles $b_t, ..., b_{t+12}$ with b_{t-4} as input conditioned on the bar profile B_t. To generate the notes $n_t, n_{t+1}, ...n_{t+3}$, the Note Layer is conditioned on both B_t and b_t; to generate the notes $n_{t+4}, ..., n_{t+7}$, the Note Layer is conditioned on both B_t and b_{t+4}; and so on. In this way, each bar profile is a condition for the 16 generated notes and each beat profile is a condition for the 4 generated notes.

All of the three layers use LSTM but the time scales of the input are different. Theoretically, the Beat Layer and Bar Layer can learn 4 and 16 times longer temporal structure than the Note Layer, respectively. Note that it is difficult to quantify the length of temporal structure learned in a model, since "temporal structure" is an abstract concept and its characterization is still an open problem. We could only probe the difference in length produced by different models indirectly by measuring the quality of the generated sequences using behavior experiments (see Section 5).

To explicitly help RNN memorize recent events and potentially repeat them, a Lookback feature was proposed for *the Lookback RNN* (Waite et al. 2016). A user study suggested that the RNN with Lookback feature outperforms basic RNN (Yang, Chou, and Yang 2017) so we also use it in our model[1]. The lookback distance is 2 and 4 for the Bar Layer, 4 and 8 for the Beat Layer, 4 and 8 for the Note Layer. Therefore, the Note Layer without the condition of the Beat layer and Bar layer is equivalent to *the Lookback RNN*.

[1]For fair comparison in experiments, all models were equipped with this feature.

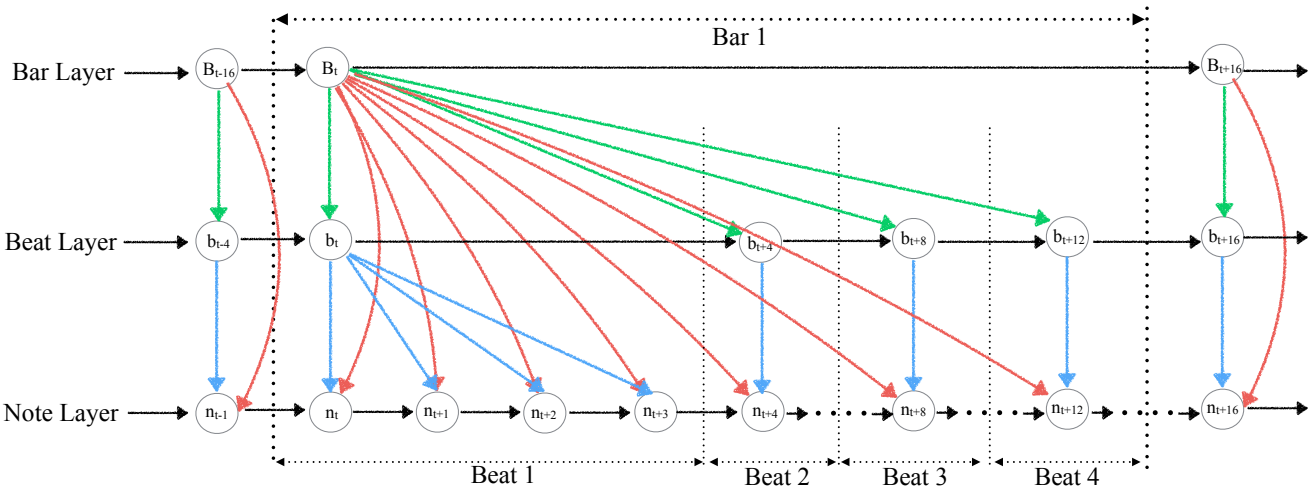

Figure 4: Architecture of HRNN. From top to bottom are Bar Layer, Beat Layer and Note Layer respectively. Inner layer connections along time are shown with black lines. Connections between layers are shown with green lines, blue lines and red lines.

LSTM-Based Event Sequence Generator

Bar profiles, beat profiles and notes can be abstracted as events, which can be generated by RNN. It might be better to use different models for generating different types of events, but for simplicity we use the same LSTM-based event sequence generator for the Bar Layer, Beat Layer and Note Layer.

The event sequence generator G_θ is trained by solving the following optimization problem:

$$\max_{\theta} \sum_{y \in \mathcal{Y}} \sum_{t=1}^{\text{len}(y)} \log p(y_t | y_0, ..., y_{t-1}, c_t) \tag{1}$$

where θ are the parameters of the generator, y is a sequence sampled in the event sequences dataset $\mathcal{Y}$. And y_t denotes the t-th event in y, c_t denotes the condition for y_t.

LSTM is used to predict the conditional probability in Eq. (1), which is characterized by input gates i_t, output gates o_t and forgetting gates f_t (Hochreiter and Schmidhuber 1997):

$$\begin{aligned} i_t &= \sigma(W_{ix} x_t + W_{im} m_{t-1}) \\ f_t &= \sigma(W_{fx} x_t + W_{fm} m_{t-1}) \\ o_t &= \sigma(W_{ox} x_t + W_{om} m_{t-1}) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_{cx} x_t + W_{cm} m_{t-1}) \\ m_t &= o_t \odot c_t \\ p_t &= \text{Softmax}(m_t) \end{aligned} \tag{2}$$

where W_{ix}, W_{im}, W_{fm}, W_{fm}, W_{ox} and W_{om} are trainable parameters, $\odot$ denotes the element-wise multiplication and $\sigma(\cdot)$ denotes the sigmoid function. The y_{t-1} is used as input x_t.

The lookback feature is added to the Bar Layer, Beat Layer and Note Layer, to help the model memorize recent events and potentially repeat them. The lookback distance is 2 and 4 for the Bar Layer, 4 and 8 for the Beat Layer, 4 and 8 for the Note Layer.

During generation, given a primer sequence as an initial input sequence, the LSTM network generates the distribution p_0 over all candidate events. The next event was chosen by sampling over p_0. The successive events are generated according to $p(y_t | y_0, ..., y_{t-1}, c_t)$.

Experiments

Evaluating the performance of the models for melody generation is difficult. The main reason is that measuring the quality of the generated melodies is subjective and it is hard to find an objective metric.

We evaluated three generative models, HRNN-1L, HRNN-2L and HRNN-3L mainly based on behavioral experiments. HRNN-3L is the model we described in the previous section. HRNN-2L is the HRNN-3L without the Bar Layer while HRNN-1L is the HRNN-3L without the Bar Layer and the Beat Layer. Note that HRNN-1L is actually the *Lookback RNN* developed by Google Magenta (Waite et al. 2016). The music pieces generated by the models were not post-processed.

All melodies used in experiments were publicly available [2].

Implementation Details

All LSTM networks used in experiments had two hidden layers and each hidden layer had 256 hidden neurons. They were trained with Adam algorithm (Kinga and 2015) and the initial learning rate was 0.001. The minibatch size was 64. The β_1, β_2 and ϵ of Adam optimizer were set to 0.9, 0.999, 1e-8. To avoid over-fitting, dropout with ratio 0.5

[2] https://www.dropbox.com/s/vnd6hoq9olrpb5g/SM.zip?dl=0

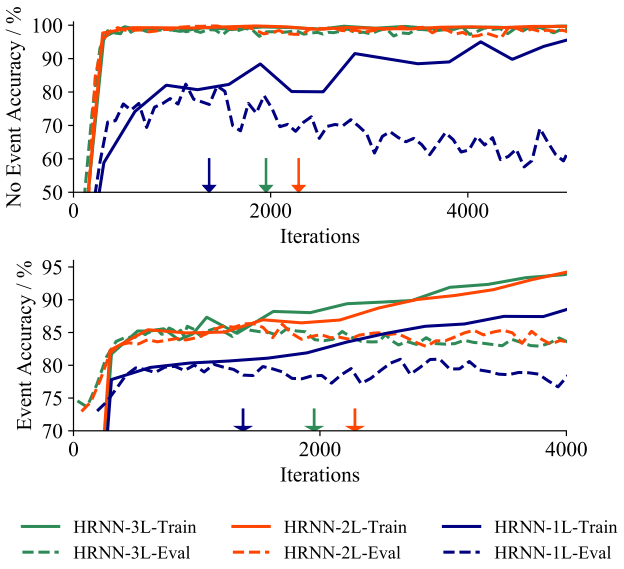

Figure 5: The accuracy curves of Note Layer for training and validation dataset. **Left**: no-event accuracy curves. **Right**: event (both note-on event and note-off event) accuracy curves. Arrows indicate iterations at which training stopped to prevent over-fitting.

was adopted for every hidden layer of LSTM and validation-based early stopping (see Fig. 5) was employed so that the training was stopped as soon as the loss on the validation set increased for 5 times in a row (the model is evaluated on validation set every 20 training iterations).

In each generation trial, primer sequences (both profiles and melodies) were randomly picked from the validation dataset. For the Bar Layer and Beat Layer, one profile is given as the primer. For the Note Layer, the length of the primer sequence is 1 beat. Beam search with a beam of size 3 was used in all experiments.

Dataset

We collected 3,859 lead sheets with the time signature of 4/4 in MusicXML format from `http://www.wikifonia.org`. We have made these lead sheets publicly available[3] 90% of the lead sheets were used as training set and the other 10% were used as validation set. The speed of most music pieces in the dataset is 120 beats per minute. To guarantee the correct segmentation of melodies, all melodies started with weak beats were removed so that we can take bar as a basic unit.

Guiding Effect of Profiles

To verify whether the beat and bar profiles can guide the generation of melody, we plotted the Note Layer's accuracy curves of in Fig. 5. Here both event accuracy (accuracy of the note-on event and note-off event; chance level is $1/37$) and no-event accuracy (accuracy of the no-event; chance level is $1/2$) are plotted.

With beat and bar profiles, the Note Layer learned the pattern of no-event quickly and easily. For models with profiles, the accuracy of no-event increased to nearly 100% at about 200 iterations while the model without profile converged slowly and over-fitting started after about 2000 iterations. Since rhythm is encoded by no-event (see Section), this showed that the Note Layer successfully utilized the rhythm provided by the beat and bar profiles. The accuracy of note-on and note-off events also improved, which means models with profiles not only did a good job in predicting rhythm, but also in predicting pitch.

With given profile sequences, the Note Layer will generate melodies with rhythm represented by profile sequence. To show this, we used handcrafted profile sequences to guide

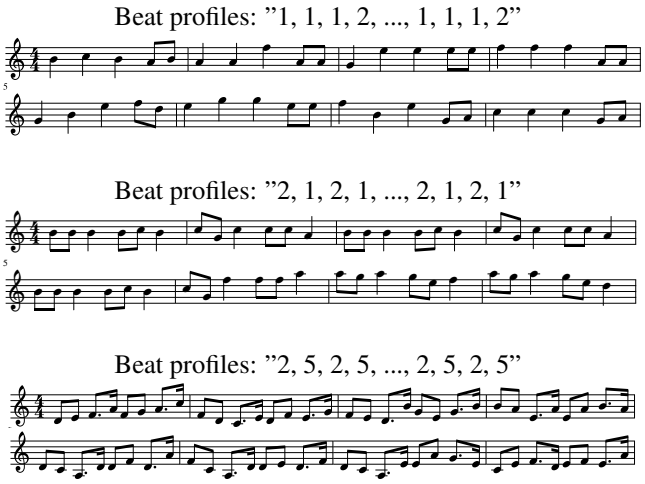

Figure 6: Melodies generated with given beat profile sequences.

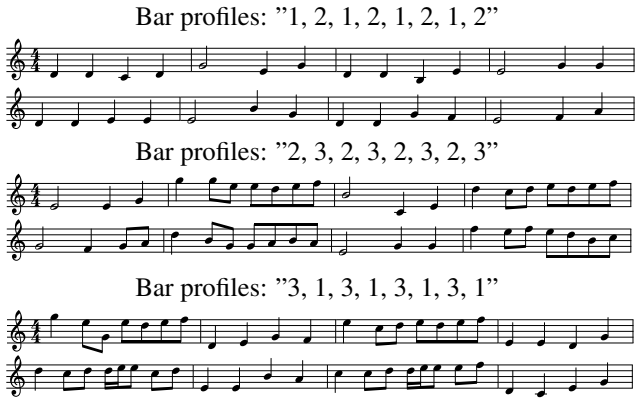

Figure 7: Melodies generated with given bar profile sequences.

[3] `https://www.dropbox.com/s/x5yc5cwjcx2zuvf/Wikifonia.zip?dl=0`

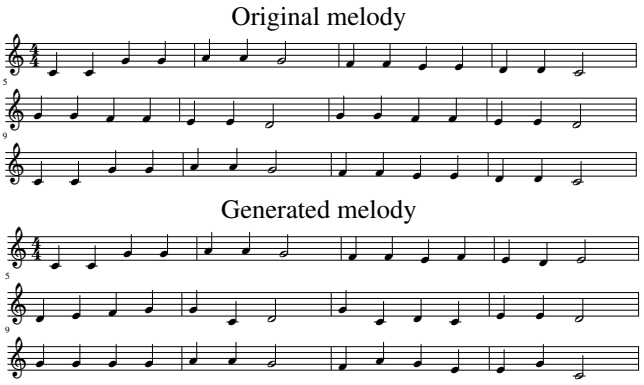

Figure 8: Original melody of Twinkle, Twinkle Little Star and generated melody by Note Layer of HRNN-3L, given profiles of the original melody.

the generation of the Note Layer. Fig. 6 and Fig. 7 show generated melodies given beat profile sequences to HRNN-2L and bar profile sequences to HRNN-3L (profile index in Fig. 2). The results verified that the generated melodies are strongly constrained by the given profile sequence patterns. The same conclusion can be obtained using fixed beat profiles and bar profiles extracted from existing melodies. We extracted beat profiles and bar profiles of children's rhymes *Twinkle, Twinkle Little Star* and generated melody conditioned on these profiles. The result is shown in Fig. 8. The audio files can be found in **Supplementary Materials**. The rhythm of the generated melody is unison with the original melody, which suggests that the beat profiles and bar profiles effectively guided the generation of melody.

Qualitative Comparison

The strong guiding effect of profiles implies that the Note Layer could output good melodies if higher layers could generate good profile sequences. Since note sequences are much longer than their profile sequences, learning the latter should be easier than learning the former using the same type of model. Thus, compared to HRNN-1L, melodies generated by HRNN-2L and 3L model should be more well-organized and keep better long-term structures. The qualitative comparison verified this point. Three typical music pieces generated by HRNN-1L, HRNN-2L, HRNN-3L with the same primer note were shown in Fig. 9. The melody generated by HRNN-1L has basic rhythm, but also irregular rhythmic patterns. And the melodies generated by HRNN-2L and HRNN-3L contain less irregular rhythmic patterns.

Comparison of Different Number of Layers

Three human behavior experiments were conducted to evaluate melodies generated by models. For this propose, we built an on-line website where people could listen to melodies and give their feedback. To model real piano playing scenario, sustain pedal effect was added to all model generated and human composed musical pieces evaluated in these experiments. This was achieved by extending all notes' duration so that they ended at the end of the corresponding bars.

Two-Alternative Forced Choice Experiment We randomly provided subjects pairs of melodies with the length of 16 bars (about 32 seconds) and asked them to vote (press one of two buttons in the experiment interface) which melody sounded better in every pair. This is the two-alternative forced choice (2AFC) setting. Subjects had infinite time for pressing the buttons after they heard the melodies. Pressing the button started a new trial.

Three types of pairs were compared: HRNN-3L versus HRNN-1L, HRNN-2L versus HRNN-1L and HRNN-3L versus HRNN-2L. Each model generated a set of 15 melodies and in every trial two melodies were randomly sampled from the two corresponding sets. Different types of pairs were mixed and randomized in the experiment.

Call for participants advertisement was spread in a social media. 1637 trials were collected from 103 IP addresses (Note that one IP address may not necessarily correspond to one subject). The results are shown in Fig. 10. In nearly two-thirds of trials, melodies generated by hierarchical models were favored (Pearson's chi-squared test, $p = 3.96 \times 10^{-10}$ for HRNN-3L versus HRNN-1L and $p = 2.70 \times 10^{-8}$ for HRNN-2L versus HRNN-1L). In addition, subjects voted more for melodies generated by HRNN-3L than by HRNN-2L ($p = 3.38 \times 10^{-6}$)

Melody Score Experiment To quantitatively measure the quality of melodies generated by different models and verify the conclusion obtained in the online experiment, we invited 18 subjects between ages of 18 and 25 to score these melodies.

Every subject was asked to score every melody used in 2AFC experiment with 5 levels: 5 the best and 1 the worst. It took each subject about 24 minutes to finish the experiment.

We calculated the average score of every melody in Fig. 10. The results verified that the two additional layers improved the quality of melodies generated by the single-layer model (two-tailed test, $p = 0.00576$ for HRNN-3L versus HRNN-1L).

Control the Number of Parameters In the above experiment, the number of parameters of HRNN-3L (2.84M) was three times that of HRNN-1L (0.94M). That might be the reason why HRNN-3L performed better. So we trained a HRNN-1L model with 450 hidden neurons (2.79M) and conducted another 2AFC experiment (as described in Section 5.5.1) to compare their performances. A total of 203 trials were collected from 21 IP addresses. In 127 trials (62.6%, $p = 3.44 \times 10^{-4}$), melodies generated by HRNN-3L were favored, which is similar to the result (63.3%) in comparison with HRNN-1L with fewer parameters (Fig. 10 left). The results indicate that the better performance of the hierarchical structure was not mainly due to the increased number of parameters.

Music Turing Test To compare the quality of the melodies generated by the models and the quality of melodies composed by human composers, a music "Turing test" was carried out. Only two models, HRNN-3L and HRNN-1L, were

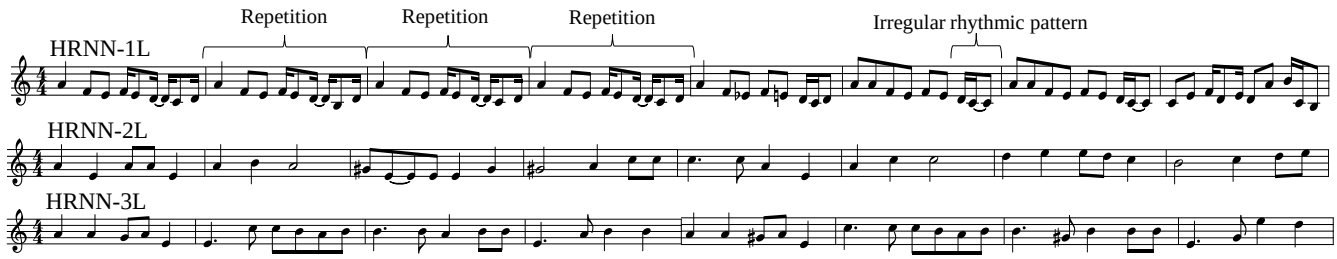

Figure 9: Melodies generated by HRNN-1L, HRNN-2L and HRNN-3L.

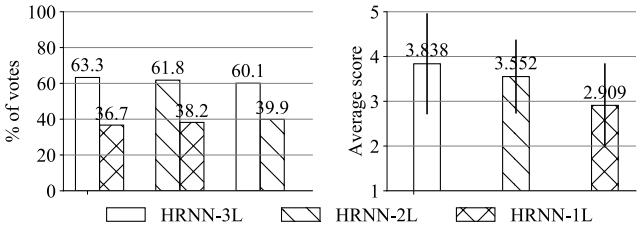

Figure 10: Results of the 2AFC experiment (**left**) and the melody score experiment (**right**).

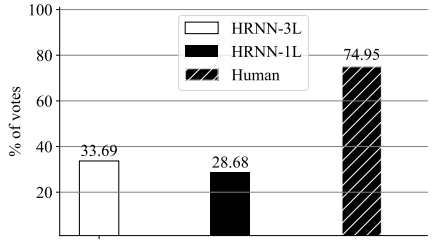

Figure 11: Results of the music turing test.

tested. We found that without chord, it was difficult for the models to generate melodies that could fool human. So chords were added as a condition of the Beat Layer and Note Layer in training and generation. Chords and primer sequences used in the generation of a melody were extracted from the same music piece in the validation set.

A total of 50 musical pieces containing 8 bars were randomly chosen from the validation set as human composed music. Then the HRNN-1L and the HRNN-3L both generated 25 melodies with the length of 8 bars. Sample We provided subjects music pieces from these 100 examples and asked them to judge if they were composed by human by pressing one of two buttons in the experiment interface. Subjects had infinite time for pressing the buttons after they heard the melodies. Pressing the button started a new trial. Feedback about the correctness of the choice was provided immediately after the subjects made the choice in every trial. Then the subjects had a chance to learn to distinguish human-composed and machine-composed melodies, which made it hard for the models to pass the Turing Test.

In this experiment, we collected 4185 trials from 659 IP addresses, among which 1018 music pieces were generated by HRNN-1L, 1003 by HRNN-3L and 2164 by human. As shown in Fig. 11, 33.69% of music pieces generated by HRNN-3L were thought to be human composed (or real), which is higher than the result of HRNN-1L, 28.68%.

It is seen that not all music pieces sampled from the original dataset were thought to be composed by humans (only 74.95% were correctly classified). This implies that some music pieces generated by the models sounded better than human composed pieces, and that the quality of the dataset is not very high.

Comparison with Other Models

Though many models have been proposed for melody generation, to the best of our knowledge, only the lookback RNN [21], attention RNN [21], MidiNet [23] and MusicVAE [19] have public available source codes. These models represent the state-of-the-art in the area of melody generation. It was reported in (Yang, Chou, and Yang 2017) that the attention RNN had similar performance to the lookback RNN. Our previous experiments have shown that HRNN-3L performed better than the lookback RNN, i.e. HRNN-1L. We then compared MidiNet and MusicVAE with HRNN-3L based on human evaluations.

MusicVAE MusicVAE is a variatiaonal autoencoder that can generate melodies with the length of 16 bars. We compared the HRNN-3L model with a MusicVAE trained on our dataset (with the same training setting in the original paper) and their pretrained MusicVAE using the 2AFC setting separately. Each model generated 15 melodies with the length of 16 bars. In each experiment, we randomly provided subjects 20 pairs of melodies and subjects were asked to vote for the better sounded melody.

In the comparison between HRNN-3L and MusicVAE that was trained on our dataset, 435 trials were collected from 17 IP addresses. In 317 trials (72.6%, $p = 1.41 \times 10^{-21}$), melodies generated by HRNN-3L were favored. We found the rhythm of melody generated by this MusicVAE is chaotic. One reason might be that the size of our dataset is too small compared with the dataset used in (Roberts et al. 2018).

We then compared HRNN-3L and Pretrained-MusicVAE. 461 trials were collected from 21 IP addresses. In 293 trials (63.5%, $p = 5.82 \times 10^{-9}$), melodies generated by HRNN-3L were favored. We generated 200 melodies with the Pretrained-MusicVAE and the statistics on 200 melodies

show that about 40% of notes generated by Pretrained-MusicVAE had pitches lower than C2, which made some melodies sound strange.

MidiNet Another 2AFC experiment was used to compare HRNN-3L with MidiNet (Yang, Chou, and Yang 2017). The MidiNet was trained on our dataset with the same training setting in the original paper. Since MidiNet required chords as an input, chords were used as a condition for MidiNet and HRNN-3L. MidiNet generated 25 melodies with the length of 8 bars conditioned on chords. The 25 melodies of HRNN-3L used in the Music Turing Test were used here for comparison.. In this 2AFC experiment, we randomly provided subjects pairs of melodies (HRNN-3L versus MidiNet) and asked them to vote for the better sounded melody in every pair. 290 trials were collected from 28 IP addresses. In 226 trials (77.9%, $p = 1.85 \times 10^{-21}$), melodies generated by HRNN-3L were favored.

Discussions

In this paper, we present a hierarchical RNN model to generate melodies. Two high-level rhythmic features, beat profile and bar profile, are designed to represent rhythm at two different time scales respectively. The human behavior experiment results show that the proposed HRNN can generate more well-organized melodies than the single-layer model. In addition, the proposed HRNN, though very simple, can generate better melodies than the well-known models MusicVAE and MidiNet.

In the Music Turing Test, only 33.69% pieces generated by the proposed model were thought to be composed by human. This proportion is still far from our expectation and there is still a long way to go for developing a perfect automatic melody generator. However, under current technology, HRNN has achieved good enough results. On one hand, one should notice that automatic generation of other forms of data is in the similar stage. For example, many state-of-the-art machine learning models trained on natural images (Zhu et al. 2017)(Isola et al. 2017) generate no more than 30% images that can fool human. On the other hand, the dataset used in this study is not good enough (only 74.95% human composed pieces were thought to be composed by human) which has hurt the performance of the model. If the model is trained on a dataset in which nearly all human composed pieces can be correctly classified, one may expect that about 44.9% (=33.69/74.95) pieces generated by the model would fool human subjects.

The proposed approach of course have many limitations which should be considered in future. First, since we quantized a bar into 16 time step, the encoding could not represent triplet or other types of rhythm. Second, we only selected musical pieces with 4/4 time signature from the original dataset for training. More time signatures should be taken into consideration to improve the capability of the model. Third, we only considered beats and bars as larger units than notes, and did not consider phrases which are often present in pop music, since they are not labeled in the dataset. With larger time scale units, the model may output pieces with longer temporal structure.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 61332007, 61621136008 and 61620106010.

References

Bargiela, A., and Pedrycz, W. 2012. *Granular Computing: An Introduction*, volume 717. Springer Science & Business Media.

Boulanger-Lewandowski, N.; Bengio, Y.; and Vincent, P. 2012. Modeling temporal dependencies in high-dimensional sequences: application to polyphonic music generation and transcription. In *Proceedings of The 29th International Conference on Machine Learning (ICML)*.

Chen, C.-C., and Miikkulainen, R. 2001. Creating melodies with evolving recurrent neural networks. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 3, 2241–2246.

Chu, H.; Urtasun, R.; and Fidler, S. 2016. Song from pi: A musically plausible network for pop music generation. *arXiv preprint arXiv:1611.03477*.

Colombo, F.; Muscinelli, S. P.; Seeholzer, A.; Brea, J.; and Gerstner, W. 2016. Algorithmic composition of melodies with deep recurrent neural networks. In *Proceedings of the 1st Conference on Computer Simulation of Musical Creativity*, number EPFL-CONF-221014.

Colombo, F.; Seeholzer, A.; and Gerstner, W. 2017. Deep artificial composer: A creative neural network model for automated melody generation. In *International Conference on Evolutionary and Biologically Inspired Music and Art*, 81–96. Springer.

Dong, H.; Hsiao, W.; Yang, L.; and Yang, Y. 2018. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*.

Eck, D., and Schmidhuber, J. 2002. A first look at music composition using LSTM recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale* 103.

Hadjeres, G.; Pachet, F.; and Nielsen, F. 2017. DeepBach: a ateerable model for Bach chorales generation. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 1362–1371.

Hinoshita, W.; Arie, H.; Tani, J.; Okuno, H. G.; and Ogata, T. 2011. Emergence of hierarchical structure mirroring linguistic composition in a recurrent neural network. *Neural Networks* 24(4):311–320.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Isola, P.; Zhu, J.-Y.; Zhou, T.; and Efros, A. A. 2017. Image-to-image translation with conditional adversarial networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Johnson, D. D. 2017. Generating polyphonic music using tied parallel networks. In *International Conference on Evolutionary and Biologically Inspired Music and Art*, 128–143. Springer.

Kinga, D., and , J. B. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Koutnik, J.; Greff, K.; Gomez, F.; and Schmidhuber, J. 2014. A clockwork RNN. In *International Conference on Machine Learning (ICML)*, 1863–1871.

Li, J.; Luong, T.; and Jurafsky, D. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, 1106–1115.

Mehri, S.; Kumar, K.; Gulrajani, I.; Kumar, R.; Jain, S.; Sotelo, J.; Courville, A.; and Bengio, Y. 2016. Samplernn: An unconditional end-to-end neural audio generation model. *arXiv preprint arXiv:1612.07837*.

Mogren, O. 2016. C-rnn-gan: A continuous recurrent neural network with adversarial training. In *Constructive Machine Learning Workshop (CML) at NIPS 2016*.

Mozer, M. C. 1994. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science* 6(2-3):247–280.

Roberts, A.; Engel, J.; Raffel, C.; Hawthorne, C.; and Eck, D. 2018. A hierarchical latent vector model for learning long-term structure in music. In Dy, J., and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 4364–4373. Stockholmsmssan, Stockholm Sweden: PMLR.

van den Oord, A.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A.; and Kavukcuoglu, K. Wavenet: A generative model for raw audio. In *9th ISCA Speech Synthesis Workshop*, 125–125.

Waite, E.; Eck, D.; Roberts, A.; and Abolafia, D. 2016. Project magenta: Generating long-term structure in songs and storie.

Yamashita, Y., and Tani, J. 2008. Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment. *PLoS Computational Biology* 4(11):e1000220.

Yang, L.-C.; Chou, S.-Y.; and Yang, Y.-H. 2017. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, Suzhou, China*.

Zhu, J.-Y.; Park, T.; Isola, P.; and Efros, A. A. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2223–2232.

Zhu, H.; Liu, Q.; Yuan, N. J.; Qin, C.; Li, J.; Zhang, K.; Zhou, G.; Wei, F.; Xu, Y.; and Chen, E. 2018. Xiaoice band: A melody and arrangement generation framework for pop music. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, 2837–2846. New York, NY, USA: ACM.