Evaluation of Generative Architectures for Music Generation

Qien Song*
Department of Computer Science
University of Toronto
Toronto, ON M5S 1A1
johnsqe.song@mail.utoronto.ca

Fengkai Ye*
Department of Computer Science
University of Toronto
Toronto, ON M5S 1A1
fengkai.ye@mail.utoronto.ca

Huifeng Wu*
Department of Computer Science
University of Toronto
Toronto, ON M5S 1A1
kentwhf.wu@mail.utoronto.ca

Abstract

Music composition with complex models has received more attention in recent years, fuelled by the rising interests in algorithmic art creations. This project aims to generate polyphonic melodies with a linguistic approach, comparing performance between variants of hidden Markov models (HMM) and an Encoder-Decoder network that uses long-short term memory (LSTM) cells, which are commonly used in language generation tasks. The main objective of this project is to construct pleasant melodies that sound indistinguishable from human-composed ones. The project evaluates the models' performance by both quantitative and qualitative measures and discusses possible areas for explorations.

1 Introduction

Melody and speech share many common traits. They both convey sentiments using a combination of stresses, accents and pauses. Music is a simplified version of a speech, consisting of fewer vocabularies with less than 200 tones (including octave differences) in total. One way to view music is to decompose music into notes which represent pitch, duration, and velocity of a sound in musical notation. Similar to language, music also has dependencies over time. In our project, we focus on generating music by training probabilistic models to fit the underlying distributions of sequences of notes in a corpus of piano music.

2 Related Work

2.1 Music Generation

Music Generation have been algorithmically studied by many machine learning models. Kotecha et al. produced a Bi-axial long-short term memory (LSTM) [1] network trained with a kernel reminiscent of convolutional ones to compose polyphonic music [2]. In comparison to this deterministic approach, hidden Markov models (HMM) is a probabilistic framework that enables to simulate and represent distributions over longitudinal data. Yanchenko1 et al. examined the use of HMM and its variants in

*All authors made equal to the project. The naming order is random.

Preprint. Under review.

piano pieces from the Romantic era and successfully generated new human-sourced like pieces[3]. Besides, HMM could be efficiently used for action recognition[4]. Motivated by above works, we use their results as a great guideline to implement several variant networks based on LSTM and HMM to compose melody. At the end, we compare and evaluate the two types of generative models.

3 Model Architecture

3.1 Hidden Markov Model

The model observes some stochastic processes W_i from time 1 to t, which are correspondingly generated from some hidden unobservable states S_i that follows a Markov process. An HMM can be specified by the following notations,

$$\begin{aligned} \text{output space } W &= \{W_1, ..., W_K\} \\ \text{state space } S &= \{S_1, ..., S_N\} \\ \text{initial state probabilities } \Pi &= \{\pi_1, ..., \pi_N\} \\ \text{state transition probabilities } A &= \{a_{ij}\} \text{ where } i, j \in S \\ \text{emission probabilities } B &= b_i(x) \text{ where } i \in S, w \in W \end{aligned}$$

For music generation, we aim to first learn the model parameters $\theta = \{A, B, \Pi\}$ that maximize $P(x_{1:n}; \theta)$, given a sequence of observations $x_{1:n}$. The problem can be solved by the Baum-Welch algorithm, which is a special case of the Expectation-Maximization algorithm. We will then sample a sequence of notes with the learned model parameters.

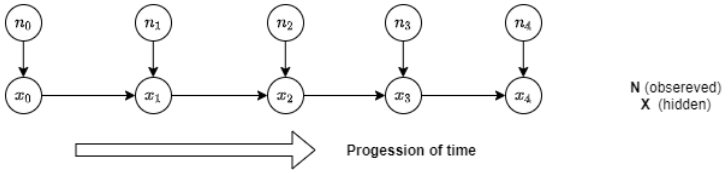

Figure 1: Details of the architecture of an HMM network. X denotes the hidden states, and N denotes the observed music notes.

3.2 Long Short-Term Memory

An LSTM is a class of recurrent feed-forward neural network that uses feedback loops to process sequential data. Memory from previous hidden units are passed onto next ones given a sequence of input, which allows information to persist over time. Also, LSTM allows input of infinite length to be fed in, and the output calculation is dependent on historic information, so-called "memory". LSTM networks is a typical RNN architecture introduced by Hochreiter and Schmidhuber [1] in 1997. The cell states of LSTM in Figure 2 solve the issue of vanishing gradient in vanilla RNN by forgetting less important message and storing more relevant one to long-term memory [5].

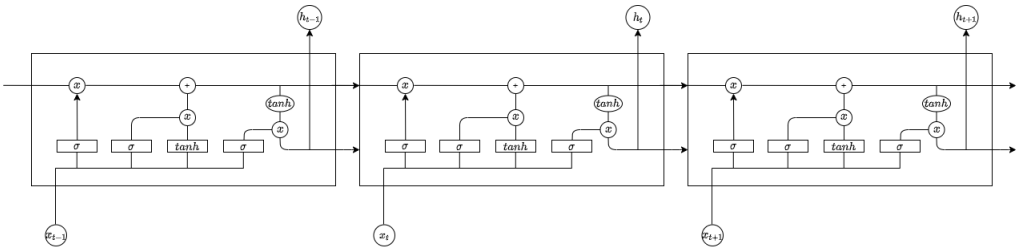

Figure 2: Details of the architecture of LSTM network. Message is transmitted in total t LSTM cells, where σ is sigmoid functions for activation, and $+$ is an addition operation depend on previous hidden units and new input.

4 Methodology

4.1 HMM Variant I: Layered Hidden Markov Model

In our project, we experiment with a Layered Hidden Markov model (LHMM) [6] with three layers. The three layered LHMM consists of 3 levels of HMMs, where the HMM at the current layer provides observations to the next layer of HMM. As shown in Figure 3, the top layer generates observation states that become hidden states in the middle layer. A new set of observation states are then generated in the middle layer and fed into the bottom layer as hidden states, which conclude final observation states.

The training process for LHMM is described as follows,

1. Given input sequence of observations $x_{1:n}$, train a base HMM and learn the model parameters by the Baum-Welch algorithm.
2. Use the Viterbi algorithm to choose a hidden state sequence that best explains the observations given the learned parameters
3. Use the output from step 2 to train a new HMM and repeat the previous steps until finding the hidden state sequence for the last layer.

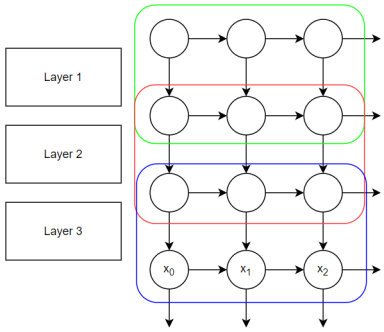

Figure 3: Architecture of LHMM. A 3 layers HMM is used in our experiment.

4.2 HMM Variant II: Hidden Markov Model Interpolation

For our experiment, we investigate the effect of linear interpolation between two HMMs. A generic HMM is trained with large dataset and a specific HMM with the same number of hidden states is trained with a little data from a single source. A new set of HMM parameters are then formed by interpolation between parameters of two HMM with

$$A = \lambda A_{generic} + (1 - \lambda) A_{specific} \tag{1}$$

$$B = \lambda B_{generic} + (1 - \lambda) B_{specific} \tag{2}$$

$$\Pi = \lambda \Pi_{generic} + (1 - \lambda) \Pi_{specific} \tag{3}$$

where initial state probabilities for each HMM is denoted as Π, state transition probabilities matrix for each HMM is denoted as A and emission probabilities matrix for each HMM is denote as B.

4.3 Variant III: LSTM Encoder-Decoder Network

We manipulate an Encoder-Decoder structure with LSTM cells to train the model to compose music. In this model, we use a melodic sequence of length t to predict the next sequence of length t. To compare with the actual Beethoven composition of the next sequence, we use cross entropy loss as our objective function.

4.3.1 Encoder and Decoder

As indicated in Figure 4, the model consists of an encoder and a decoder, both of which are created with LSTM cells.

In the encoder, $\{x_i\}_{i=1}^{t}$ represents the input melody sequence of length t. The input are embedded and passed into the encoder. Corresponding to each input, tone are encoder hidden states $\{h_i\}_{i=1}^{t}$ that encode long-term information of the melody, such as key signature, tempo and harmony. This information is crucial for the generation of future melody. Each individual cell hidden state decides what information is passed forward to the next hidden state, and ultimately the last hidden state of the encoder is passed forward to the decoder.

In the decoder, $\{\tilde{x}_{1+t}\}_{i=1}^{t}$ represents the input melody sequence to the decoder. They are derived from the target sequence $\{x_{i+t}\}_{i=1}^{t}$, which are the t tones succeeding input to the encoder. Similarly, the decoder hidden states $\{\tilde{h}_{i+t}\}_{i=1}^{t}$ correspond to each input. Each decoder cell generates a probability array $\{p_{0k}\}_{k=1}^{V}$, where V is the number of possible different combinations of notes/chords. The element with the highest probability is chosen as y_i.

Training is performed by calculating cross entropy loss on $\{y_i\}_{i=1}^{t}$ with the target sequence $\{x_{i+t}\}_{i=1}^{t}$.

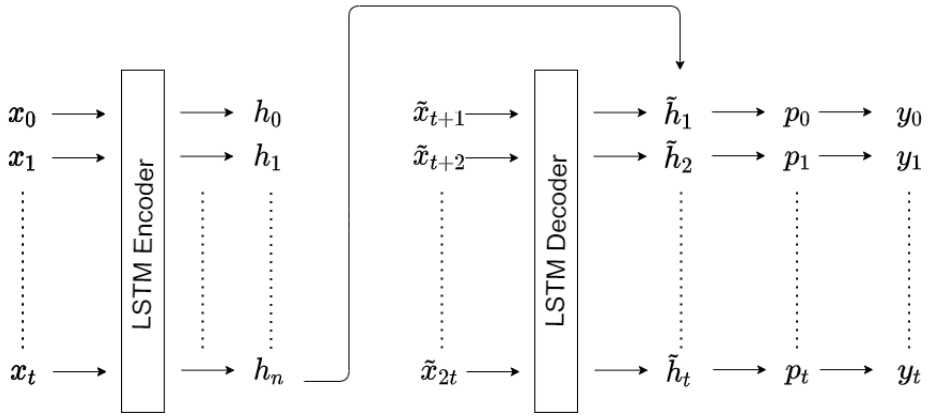

Figure 4: Architecture of LSTM Encoder-Decoder. x_i, y_i denotes input and output sequences respectively. h_i, $\tilde{h}_i$ are hidden states of LSTM cells.

4.3.2 Teacher Forcing and Attention Mechanism

A traditional LSTM decoder passes the output y_i as the input x_{t+i+1} to the decoder. However, in training, passing the target tone directly can reduce exposure bias. If the output y_i is incorrect, then all succeeding cells are affected. The autoregressive nature of the decoder can create a strong bias in the final output. Therefore, a technique called teacher forcing is used in training to pass the target melody directly into the decoder. In evaluation mode, teacher forcing is turned off so that the generated sequence can be infinitely extended.

Another technique used is attention mechanism. A melodic sequence is highly dependent on its previous notes. However, it may weigh the previous tones differently as some may be indicative of the key signature, and main melody, while others may be simply transitive or repetitive embellishments. The attention mechanism allows the decoder cell to weigh the encoder hidden states differently so that some features are more "attended" to. This process first involves determining weights $\{w_i\}_{i=1}^{t}$ for each encoder hidden state. The weights are generated by using cosine similarity between the previous hidden state $\tilde{h}_{i+t-1}$ and each encoder state, as indicated in Figure 5. With the weights, we calculate a context vector $c_i = \sum_{j=1}^{t} w_j h_j$ that represent the past information needed to predict the next tone. While the nature of the context vector can be theorized, the precise content as to exactly what the weight stands for, is more blackbox-magic. Finally, the context vector is concatenated with

x_{i+t} to obtain the decoder input $\tilde{x}_{i+t}$.

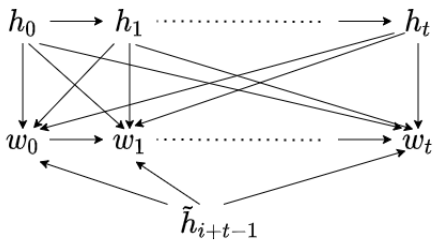

Figure 5: Attention mechanism used in encoder-decoder architecture.

5 Experiment

5.1 Dataset

We mainly use Beethoven piano music stored in MIDI format. MIDI file is a specific file format for storing music compositions, thus making extracting pitch, duration and velocity features conveniently. Pitches can be viewed as frequency of sound and are encoded into integer categorical variables.

Piano music is downloaded from [7] in MIDI format. MIDI files are then broken down into note pitches, durations and velocities. There are 86451 notes/chords in 29 compositions, with 3275 unique notes/chords combinations.

The note pitches can be viewed as frequency of sound and are encoded into integer categorical variables in each model. Note velocity determines the loudness of each note and it is also coded from 0 to 127. Time click specifies when the note is played and gaps between two observations is constant across each MIDI file. Training is only performed on the pitches, while the durations and velocities are stored as templates for later use.

For the variants of HMM model, our training set consists of a random piece from these compositions, and for interpolated-HMM, the generic HMM uses JSB Chorale dataset. [8] All HMM models except interpolated-HMM do not require substantial amount of data for training. In comparison, the LSTM model uses all the Beethoven compositions.

5.2 Metrics

To measure the quality of our music generated, we adopted the idea from [3] and evaluated the originality of our music through calculating entropy, mutual information and Word error rate (WER) of our generated with the original music template. In addition, to measure music diversity, we examine the categories of intervals produced.

Entropy measures the average amount of information we get from observation the notes sequence. Hence, entropy can be used to measure how predictable our generated music is. We expect a generated piece to be more original and less predictive when we observe a higher entropy.

Mutual information measures the average amount of information shared between the original music and the generated music. A higher mutual information indicates that the generated music shares more similarity with the original and hence we consider it to be less original.

WER measures the ratio of minimum total substitution, insertion and deletion required to change the generated sequence back to the original sequence, divided by the length of the original sequence. It is a common measure in speech transcription models. A higher WER between the generated music and the original music indicates the generated sequence is more different from the original piece and hence more original.

We also attempt to summarize the composition of intervals by their structure and texture. The two structural types are harmonic, which are chords, and melodic, which are monophonic sequence of

notes. The two texture types are consonance that are specific note sequences or chords associated with joy, and dissonance, which are those that associate with melancholy. We can analyse the naturalness of the music by benchmarking results against original music.

6 Results

6.1 Quantitative Analysis

The main measure of successful music generation requires an equilibrium between the originality of the piece and the composer's idiosyncrasies exhibited. To put it plainly, the music generated should be original but identifiable as "Beethoven" style.

Examining with different metrics in Table 1, we see that all 4 models demonstrate high entropy. The original pieces have the second highest entropy due to melodic diversity in Beethoven's style. The interpolated-HMM and LSTM model have entropy above 3 as well, while Interpolated-HMM has the highest entropy. This matches our expectation since interpolated-HMM includes an HMM that learned from a generic dataset and LSTM learns from all Beethoven's pieces. Therefore, we expect them to capture more variety in the generated output. A high entropy also indicates that the sequences generated have high variations in its intervals.

However, a completely random sequence has high entropy, too. To better diagnose the model performance, it is necessary to examine the generated pieces' mutual information with the original ones. This serves as an indicator of the degree to which the generated music reflects Beethoven's characteristics. The LSTM model has the highest mutual information, which shows that when a listener hears the LSTM generated samples, he/she is more likely to identify it with Beethoven. This is partially attributed to the complex internal structure of the model and the substantial amount of training data seen. The interpolated and layered HMM also contains sufficient mutual information as training on specific Beethoven pieces, namely Fur Elise, helps to encapsulate Beethoven style.

Furthermore, from WER we see that values fall between 0.7 - 0.8 for most trained models, indicating that the pieces have high originality, but share some features with the original works.

	Metric		
Model	Entropy	Mutual Information	WER
Beethoven	3.42	3.42	0.0
Random	3.56	0.25	0.89
LSTM	3.21	1.37	0.70
HMM Base	2.60	0.35	0.78
HMM Interpolate	3.31	0.61	0.87
HMM Layered	2.59	0.68	0.71

Table 1. Metrics used for model evaluation. 10 generated samples are drawn from each model and all metrics are averaged.

We can further breakdown the style of the pieces as in Table 2. Among the harmonic intervals, there is an approximate 60-40 split between consonance and dissonance for all models. Among the melodic intervals, the approximate split is 65-35. However, it is difficult to distinguish the music's character from these measures as the random sequence also follows this distribution. Therefore, to better examine the generated pieces' naturalness, human listeners are required.

	Interval Decomposition			
Model	% Consonant Harmonic	% Dissonant Harmonic	% Consonant Melodic	% Dissonant Melodic
Beethoven	59.98	40.02	68.55	31.45
Random	60.63	39.36	63.98	36.02
LSTM	58.88	41.12	67.67	32.33
HMM Base	63.41	36.59	60.59	39.41
HMM Interpolate	55.41	44.59	58.32	41.67
HMM Layered	54.22	45.78	58.95	41.05

Table 2. The interval composition table. The percentage value calculates melodic and harmonic intervals separately. Therefore, consonant and dissonant percentages sum up to one for each structural interval category.

6.2 Qualitative Analysis

We generated 6 samples on SoundCloud (https://soundcloud.com/john-qien-song/sets/csc412-final-project) that can be evaluated by human ears. There are 3 samples with LSTM, where the first 64 notes are original composition from Beethoven and the rest 256 notes are model improvisation. There are also 3 samples with each variant of HMM, where the entire composition is generated by the model.

Different listeners may have diverging opinions on the generated music's composure, texture and colorfulness given listeners' music knowledge and background. What can be universally agreed is that HMMs incline to use more harmonic intervals while LSTM uses more melodic intervals. The use of melodic intervals require more long-term memory as it may require 10 - 20 single notes to form a meaningful melody. HMMs are designed to predict the immediate succeeding pitch, where chords are more expressive and used more frequently. We can use a linguistic analogy to describe the difference between the two models. If the LSTM model were to say "I could not decide whether to go forward with the plan.", the equivalent generation in an HMM would be "I am hesitant to commence the scheme." In some sense, the LSTM model is more lengthy and vernacular, while the HMM is more concise and formal.

From perspective of listeners, output generated from base HMM, layered HMM and LSTM sound more like a human composed music piece than output generated from interpolated-HMM, due to the fact that interpolated-HMM mixes different music styles and could potentially produce output that sound "messy" while others trained from the Beethoven style music could exhibit a more consistent style. In addition, the piece generated by layered HMM shares some highly noticeable similarity with the trained song "Fur Elise" in some parts. This could be an indication of Layered HMM suffers from overfitting due to its structure.

7 Discussion

7.1 Limitations

A major limitation of the HMM model is the difficulty to train a generalized model with vast amount of data. As HMM uses transitional probabilities, when a large amount of data is present, the transitional probability approaches uniform distribution as the data contains compositions of different key signatures and melodies. In other words, the optimal performance is achieved when only one music piece is used as input. In contrast, the LSTM model's major limitation is that it requires a large amount of data, and substantial computing resources. Therefore, the two models can merge into an ensemble model, given the amount of input data available.

In this project, durations and velocities are not incorporated in either model. Durations and velocities resemble continuous variables, and training them together with embedding is not a viable option in the LSTM Encoder-Decoder network. It is also harder to model continuous and discrete variables together in HMM variants.

7.2 Future Work

For the LSTM model, it is possible to model notes, durations and velocities separately in 3 encoder-decoder networks and then combine them together to form a sequence of melody. This, however, would rely on the independence assumptions between these internal features of music.

Both models lack the ability to conclude a musical composition. Both models can be extended indefinitely, but in real life every melodic sequence should come to an end. Adding a decaying factor to music generation process so that it could stop intelligently would make the models more "human".

8 Conclusion

This project compares the results of the variants of HMMs with Encoder-Decoder network. We used several quantitative metrics to classify the music quality and also provided generated samples for human listeners to judge. We have found that both HMMs and the Encoder-Decoder network can generate pleasing melodic sequences, albeit with very different styles due to the internal structures of the respective models. We hope that the baseline framework used in this project can be enhanced to better encompass the abundant features of music compositions in future research.

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[2] Nikhil Kotecha and Paul Young. Generating music using an lstm network, 2018.

[3] Anna K. Yanchenko and Sayan Mukherjee. Classical music composition using state space models, 2018.

[4] Xinze Guan, Raviv Raich, and Weng-Keen Wong. Efficient multi-instance learning for activity recognition from time series data using an auto-regressive hidden markov model. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2330–2339, New York, New York, USA, 20–22 Jun 2016. PMLR.

[5] Frank Rudzicz. University of toronto csc401 course content.

[6] Nuria Oliver, Ashutosh Garg, and Eric Horvitz. Layered representations for learning and inferring office activity from multiple sensory channels. *Computer Vision and Image Understanding*, 96(2):163–180, 2004.

[7] Bernd Kruger. Classical piano midi page, 2018. Available at `http://www.piano-midi.de/beeth.htm`.

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