Graph Neural Network for Music Score Data and Modeling Expressive Piano Performance

Dasaem Jeong¹ Taegyun Kwon¹ Yoojin Kim¹ Juhan Nam¹

Abstract
Music score is often handled as one-dimensional sequential data. Unlike words in a text document, notes in music score can be played simultaneously by the polyphonic nature and each of them has its own duration. In this paper, we represent the unique form of musical score using graph neural network and apply it for rendering expressive piano performance from the music score. Specifically, we design the model using note-level gated graph neural network and measure-level hierarchical attention network with bidirectional long short-term memory with an iterative feedback method. In addition, to model different styles of performance for a given input score, we employ a variational auto-encoder. The result of the listening test shows that our proposed model generated more human-like performances compared to a baseline model and a hierarchical attention network model that handles music score as a word-like sequence.

1. Introduction

“To be or not to be, that is the question.” Imagine a professional actor speaking this sentence. Although the text itself he or she speaks is not changed, every detail such as tempo, accent, nuance, pause, and rhythm will be different from the speech of the same text by non-trained people. One needs a thorough understanding of the text and context, and skills to evoke listeners’ emotion. The detailed difference makes some actor “great” or “special”.

A similar difference is found in the performance of music as well. It is the role of instrument performers to decide the exact tempo and its subtle change, and dynamics and articulation of every note, which are notated in the score only in general and qualitative directions such as Allegro, or poco crescendo. Therefore, music performance can be regarded as an instance of a performer’s interpretation of the given music score. The artistic quality of interpretation is often measured and ranked quantitatively in music competitions, and some pianists gain international and historical fame for their interpretation. Revealing the secret of great pianists was one of the long-cherished goals in music performance analysis (Widmer et al., 2003).

Modeling expressive performance with computational methods is a task that challenges the imitation of the complex and artistic activity of expert human musicians. Starting from the rule-based approach in early period (Sundberg et al., 1983), many researchers have tackled this task with various methods, such as Gaussian Process (Teramura et al., 2008), switching Kalman Filter (Gu & Raphael, 2012), Bayesian networks (Flossmann et al., 2013), and conditional random fields (Kim et al., 2013). Some of recent research employed recurrent neural network (RNN) (Lauly, 2010; Chacón & Grachten, 2016; Malik & Ek, 2017; Jeong et al., 2018; Maezawa, 2018). However, research using deep neural networks still lacks in this task despite of the potential capability.

Recently, generative models using deep neural network achieved remarkable results in visual and audio domains such as image synthesis (Van den Oord et al., 2016; Wang et al., 2018b), video generation (Tulyakov et al., 2018), and speech synthesis (Van Den Oord et al., 2016; Wang et al., 2017). There have been also notable achievements in music generation tasks such as automatic music composition (Huang et al., 2019), and automatic music transcription and sound synthesis (Hawthorne et al., 2019).

One of the main issues in applying neural networks to music data, particularly, music score data is defining the input structure. Music score data is often handled as 1D sequential data by ordering note events with its time position and pitch (Simon & Oore, 2017; Oore et al., 2018; Jeong et al., 2018). But this is dissimilar to a sequence of word or audio samples because of distinct characteristics of musical notes. For example, two notes can be a sequence in one voice but also they can be in polyphony. In some cases, a rest can be inserted between the two, or the preceding note can be

¹Graduate School of Culture Technology, Korea Advanced Institute of Science and Technology (KAIST), Dajeon, South Korea. Correspondence to: Dasaem Jeong <jdasam@kaist.ac.kr>, Juhan Nam <juhannam@kaist.ac.kr>.

Figure 1. The main idea of the proposed music score encoding model. It learns note-level representations with a graph neural network and their measure-level dependency with a recurrent neural network. Although the graph is represented for only a single measure, it connects the entire notes in the piece regardless of the measure boundary.

sustained while the following note begins. Representing the two notes as an 1D sequence discards these multifaceted relations between musical notes.

Another input representation is plotting notes as a piano-roll-like 2D matrix by sampling the note activity in time (Malik & Ek, 2017; Maezawa, 2018). The piano-role matrix can be regarded as an image and so it is easy to handle it with a convolutional neural network (CNN). However, the sampling-based representation requires much higher dimensionality compared to the event-based representation of notes. Furthermore, if the rhythm of music becomes complex, the temporal dimensionality increases even more according to the required time resolution. For example, our music score dataset includes septuplets (seven notes for a quarter note) and sixteenth notes. This requires at least 28 grids per quarter note to discretize the time axis. This high dimensionality in time may hinder the model from learning long-term musical structure (Korzeniowski & Widmer, 2017).

To address this issue, we propose a model based on graph neural network (GNN). This represents music score as a graph that captures multifaceted relations of note events in music score. Each note is regarded as a node in the graph and the neighboring notes are connected with different types of edges depending on their musical relations in the score. As illustrated in Figure 1, we incorporate the graph neural network into a measure-level RNN model to learn the long-term structure. Furthermore, we suggest an iterative loop for updating the inputs of GNN and RNN using the results from each other.

As similar to other generative tasks, generating various types of results for a given condition is an important goal for modeling expressive performance. We employ variational auto-encoder (VAE) (Kingma & Welling, 2014) to train the model with the data that consist of the same condition $C$ but with different output $Y$, e.g., different styles of performances with the same music score, without teacher-force or scheduled sampling (Bengio et al., 2015).

Note that the scope of our system focuses on generating a performance in MIDI format for a given music score in musicXML format. Automatic composition or sound synthesis of performance is out of the scope. Also, this paper will focus on the utilization of graph neural network rather than the VAE module because of the limited space in this paper.

1.1. Contribution

This paper presents the first attempt to apply a graph neural network to learn note representations from music scores in western notation. To fully exploit the temporal characteristics of music data, we propose a novel method to combine graph neural network with hierarchical attention RNN in an iterative feedback method. Although there have been many data-driven approaches for generating expressive piano performance, this paper is first to employ a deep neural network that covers the full expression of piano performance without any extra manual annotation, and also includes quantitative and qualitative evaluation.

2. Background

2.1. Graph Neural Network

While CNN and RNN achieved a significant progress in image processing and sequence modeling, respectively, there are various types of data that cannot be properly handled with these networks and graph is one of the examples. Early research for handling graph data with graph neural network (GNN) was introduced by (Gori et al., 2005) and (Scarselli et al., 2009). (Li et al., 2016) introduced gated graph neural network (GGNN), which combines a gated recurrent unit of modern RNN practice with GNN. While the previous models were restricted by contraction mapping, the GGNN model first overcame this limitation. Recent research using GNN has achieved state-of-the-art results in various tasks such as analyzing citation network (Kipf & Welling, 2016), molecular structure (Jin et al., 2018), program code (Allamanis et al., 2018), and learning structured policy (Wang et al., 2018a).

2.2. Variational Auto-encoder

Variational auto-encoder (VAE) is a type of auto-encoder that compresses the input data $x$ into a low-dimensional vector $z$. VAE differs from a standard auto-encoder in that it learns posterior $p(z|x)$ for encoding and likelihood $p(x|z)$ for decoding instead of a deterministic function. Also, VAE constrains the distribution of $z$ to follow prior $p(z)$, which is
We define six edge types in musical score:

1. **next** edge: connects a note to the next note in the same voice.
2. **rest** edge: connects two notes after the rest, having a self-connection to every note. Therefore, a total number of edges are a subset of next edges which connect notes in the same voice only. Among **voice** edges, we add **slur** edges between notes under the same slur. All edges are directed except **onset** edges. We regard forward and backward directions as a different type of edges. We also add a self-connection to every note. Therefore, a total number of edge types are twelve. Each edge type shares different weight parameters.

Among various types of graph neural network, we employ GGNN (Li et al., 2016) because of its advantage in learning node-level representations in a graph. During the information propagation, GGNN uses the propagation rule of gated recurrent unit (Cho et al., 2014) which can be represented as below:

\[
\mathbf{a}_v^{(t)} = \mathbf{A}_v^{T} \begin{bmatrix} h_1^{(t-1)} \hfill \ldots \hfill h_{|V|}^{(t-1)} \end{bmatrix} \\
\mathbf{z}_v^{(t)} = \sigma (\mathbf{W} \mathbf{a}_v^{(t)} + \mathbf{U} \mathbf{h}_v^{(t-1)}) \\
\mathbf{r}_v^{(t)} = \sigma (\mathbf{W} \mathbf{a}_v^{(t)} + \mathbf{U} \mathbf{h}_v^{(t-1)}) \\
\tilde{\mathbf{h}}_v^{(t)} = \tanh (\mathbf{W} \mathbf{a}_v^{(t)} + \mathbf{U} (\mathbf{r}_v^{(t)} \otimes \mathbf{h}_v^{(t-1)})) \\
\mathbf{h}_v^{(t)} = (1 - \mathbf{z}_v^{(t)}) \otimes \tilde{\mathbf{h}}_v^{(t-1)} + \mathbf{z}_v^{(t)} \otimes \mathbf{h}_v^{(t)}
\]

where \(\mathbf{A}\) represents an adjacency matrix of graph edges, \(h_v^{(t)}\) denotes the hidden state of node \(v\) at the \(t\)-th iteration, \(\sigma\) and \(\otimes\) represent the sigmoid function and hadamard product, respectively.

The model can be modified to limit the propagation to specific dimension of the hidden state, so that only a portion of hidden states are updated and the others remain constant. In this partial-GGNN, where the input dimension is \(D\) and the number of updated dimension is \(d\), the dimension of weight matrices \(\mathbf{W}^z, \mathbf{W}^r, \mathbf{U}^z, \mathbf{U}^r,\) and \(\mathbf{W}\) becomes \(D \times d\) instead of \(D \times D\), and the dimension of \(\mathbf{U}\) becomes \(d \times d\).

### 3. Model

Our model combines note-level gated graph neural network (GGNN) (Li et al., 2016) and measure-level hierarchical attention network (HAN) with LSTM (Yang et al., 2016) in an iterative method. In this section, we explain how these methods are implemented in our system.

#### 3.1. Gated Graph Neural Network

We employ directional multi-edge-type GGNN (Li et al., 2016) to learn note-level hidden representations from an input music score. The input of the graph neural network can be represented as a graph \(\mathcal{G} = (\mathcal{V}, \mathcal{E})\), where \(\mathcal{V}\) and \(\mathcal{E}\) denotes nodes and edges, respectively. If the input is a music score, a node \(v\) corresponds to a single note in the score and an edge \(e\) corresponds to a connection between two musically neighboring notes.

We define six edge types in musical score: **next**, **rest**, **onset**, **sustain**, **voice**, and **slur** as shown in Figure 2. A **next** edge connects a note to its following note, i.e., the following note that begins exactly when the note ends. A **rest** edge links a note with the rest following to other notes that begin when the rest ends. If there are consecutive rests, they are combined as a single rest. An **onset** edge is to connect notes that begin together, i.e., on the same onset. Notes that appear on the same onset connected as a single rest. An **slur** edge is to connect notes beginning exactly when the note ends. A **sustain** edge connects a note to its following note, i.e., the following note that begins exactly when the note ends. A **voice** edge connects notes under the same slur. All edges are directed except **onset** edges. We regard forward and backward directions as a different type of edges. We also add a self-connection to every note. Therefore, a total number of edge types are twelve. Each edge type shares different weight parameters.

The loss function of VAE is as follow:

\[
\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{rec}} + \beta D_{KL}[(q_{\theta}(z|x)||p(z)]
\]

where the first term \(\mathcal{L}_{\text{rec}}\) denotes the reconstruction error and the second term represents Kullback-Leibler divergence (KLD) between the posterior and the prior. \(\beta\) is a weight of KLD in total loss which is a hyperparameter during the training of the model (Bowman et al., 2016).

#### 3.2. Hierarchical Attention RNN

Exploiting hierarchies of information is important for deep neural network (Lecun et al., 2015). While CNN naturally learns the hierarchies in spatial data with pooling, RNN needs a modified architecture to explicitly learn such hierarchical representations. Various types of solutions have been proposed particularly in the field of natural language processing where word, sentence and paragraphs are hierarchical units in document data. They include hierarchical RNN (HRNN) (Ling et al., 2015), hierarchical multiscale RNN (HM-RNN) (Chung et al., 2017), and hierarchical attention RNN (HAN) (Yang et al., 2016). Since music also has a hierarchical structure such as beat, measure, phrase, and section, some of music generation models exploited it to improve their performance. For example, (Roberts et al., 2018) proposed a hierarchical decoder model for music.

Figure 2. An example of edge types in a music score. The note in the red box is connected with neighboring notes, having **next**, **onset**, **sustain**, **voice**, **slur** edges. Another note in the red dashed-box is connected with two notes after the rest, having **rest** edges.

We employ directional multi-edge-type GGNN (Li et al., 2016) among various types of graph neural network, we employ weight parameters.
Among various hierarchical RNN models, we employ HAN because it is directly applicable to graph neural network. In HRNN and HM-RNN, the hidden state in lower-level $h_t^0$ is fed into higher-level $h_t^1$ at the hierarchical boundary $t$. But if the lower-level network is GNN instead of RNN, it is difficult to define $h_t^0$ because there can be several nodes at the hierarchical boundary $t$, e.g., at the end of a measure in a music score. Also, $h_t$ naturally includes information of all previous state $h_0, h_1, ..., h_{t-1}$ so that a single hidden state can represent the whole previous sequence. But in the case of GNN, the range of hidden state propagation is limited to the number of iterations. Therefore, the number of minimum iterations required for a single hidden state to contain the states of all other nodes in a boundary (e.g., all notes in a measure) is not constant. On the other hand, HAN uses attention to summarize the lower-level representations, hence it is directly applicable to any type of networks such as a simple dense network, RNN, or GNN.

In our system, we compose a measure vector from notes representations using a context attention proposed in (Yang et al., 2016). Instead of strictly following the context attention, we modify it to employ a concept of multi-head attention as proposed in (Vaswani et al., 2017). Rather than applying a constant weight to the whole dimension, weight $\alpha$ and hidden state $h_v$ are divided into $I$ number of heads with the same number of dimension, so that each head of hidden state satisfies $h_v^i \in \mathbb{R}^d$, where $d = D/I$ denotes dimension of a single head and $D$ denotes a dimension of original hidden states. For each hierarchical boundary $M$ which can be a beat or a measure in a music score, the lower-level hidden state $h_v$ for the node $v$ in $M$ is summarized by context attention to compose a higher-level node $m$. The following equations illustrates the computation:

$$ u_v = \tanh(W_u h_v + b_u) $$

$$ u_v^i = u_{v,i:(i+1)d} $$

$$ h_v^i = h_{v,i:(i+1)d} $$

$$ \alpha_v^i = \frac{\exp(u_v^i,u_u^i)}{\sum_{i} \exp(u_v^i,u_u^i)} $$

$$ m^i = \sum_v \alpha_v^i \cdot h_v^i $$

$$ m = \text{Concat}(m^0, ..., m^I) $$

where $u_v$ denotes a context vector. The context vector represents a query for importance, and it is a trainable parameter. The sequence of attention sum $m$ is used for the input of the higher-level RNN to make a sequence of higher-level hidden state $H_m \in \mathbb{R}^{L_m \times D_m}$, where $L_m$ denotes number of measures. We can span $H_m$ to have the same sequence length with the lower-level hidden units by concatenating them: $H_{m,s} = [h_{m,0} \times N_s \ldots h_{m,L_m} \times N_s]$, where $N_s$ denotes the number of notes in the $s$-th measure.

### 3.3. Iterative Sequential Graph Network

A simple way to combine the outputs from different hierarchical units is concatenating them as a single vector. However, this approach has a limitation that the lower-level layers cannot be conditioned on long-term contexts encoded in higher layers because the higher-level outputs do not influence lower-level layers back. In HRNN and HM-RNN, hidden states in lower-level are fed into higher-level at the hierarchical boundary, and vice versa. In HAN, however, hidden state propagation is only done in the bottom-up direction. This limitation does not matter much when the target result of the model is a single output given a sequential input, such as in document classification which HAN was originally applied to (Yang et al., 2016). In our application where the musical representation is learned for each note in a sequence-to-sequence manner, it is crucial to consider a more extended context when calculating note-level hidden states, so that the model can learn the role of a note not only in a local context but also in a longer context such as phrase or section that it belongs to. To overcome this limitation, we propose a novel combination of GGNN and HAN that we refer to it as iterative sequential graph network (ISGN), in which GGNN and HAN feed their results to each other in an iterative way as described in Figure 3.

Instead of giving only note-level representation as an input to GGNN, we add higher-level hidden state by concatenating with the note-level data, so that each node contains both...
4. Expressive Performance Rendering System

Figure 4 shows the entire system for expressive performance rendering. We describe the input and output features, and the modules of the system.

4.1. Input and Output Features

Our model exploits pre-defined score and performance features for input and output. The feature extraction scheme is detailed in (Jeong et al., 2019). The input features include various type of musical information such as pitch and duration of note, tempo and dynamic markings. The input features are all embedded in note-level. The output features consist of tempo, MIDI velocity (loudness), onset deviation (micro-timing of each note), articulation (duration ratio), and seven features to handle piano pedaling.

4.2. Modules

Our system consists of three modules: a score encoder \(E_s\), a performance encoder \(E_p\) and a performance decoder \(D_p\). For a given score input \(X\), the score module \(E_s\) infers a score condition \(C\). The score module employs ISGN with two GGNN layers and a single layer of measure-level RNN with a long short term memory (LSTM). The initial input \(X\) passes through the three-layer dense network. The first layer of GGNN updates note-level features only, and measure-level features remain fixed. The second layer of GGNN updates the whole hidden state. The outputs of both layers are concatenated to compose \(C\) as a skip connection.

For the encoded score condition \(C\) and its corresponding performance features \(Y\), the performance encoder module encodes the performance style, \(Y\) for given \(C\), as a latent vector \(z\). The input of the style encoder is the score condition \(C\) concatenated with its corresponding performance features \(Y\). The dimension of concatenated data is reduced with a dense layer and used for input of the performance encoder. The performance encoder consists of a GGNN and two-layers measure-level LSTM. The higher-level of performance encoder employs HAN instead of ISGN since it focuses on the summarization of total sequence rather than characteristics of individual notes. During the inference, the encoder can be bypassed by sampling \(z\) from the normal distribution or using a pre-encoded \(z\).

The last part of the system is the performance decoder \(D : C, z, \tilde{Y}^0 \rightarrow \tilde{Y}\), which generates the performance features \(Y\). We employ the ISGN structure in the decoder module as well. It infers performance parameters for each note in the input score by the iterative method. We employed the concept of hierarchical decoding of latent vector in VAE introduced in (Roberts et al., 2018). Instead of directly using the encoded \(z\), we decode \(z\) into measure-level by concatenating \(z^{\times L_m}\) with measure-level representation \(H_m\) in the...
encoded score $C$ and feeding it to another LSTM module. The LSTM module returns measure-level performance style vector $z_m \in \mathbb{R}^{L_m \times D_c}$, where $L_m$ and $D_c$ represent number of measures and dimension of encoded vector $z$.

The input data of the ISGN performance decoder is a concatenation of score condition $C$, measure-level performance style vector $z_m$, and initial performance parameters $Y^0$. The hierarchical RNN of ISGN is a beat-level LSTM so that the module can directly infer beat-level tempo from the output of the RNN. The initial output $Y^0$ is generated by a simple dense network. The input of GGNN is concatenation of $C$, $z_m$, $Y^0$ with additional zero padding. The GGNN updates dimensions for $Y$ and zero-padded dimension so that $C$ and $z_m$ remain constant.

### 5. Related Works

Most notable recent research in music generation using deep neural network is Music Transformer (Huang et al., 2019). The aim of the research is generating music by combining composition and performing as a single stage. Based on the original Transformer model that consists of self-attention (Vaswani et al., 2017), they proposed relative positional embedding for pitch and time and succeeded in generating a musical piece with long-term structure by outperforming their former LSTM-based model (Oore et al., 2018).

As aforementioned, however, the task is different from the scope in this paper. While our research focuses on interpreting and performing a given score, Music Transformer is more like making improvisations.

Recently, VAE has been utilized in several music data generation models, such as music generation (Roberts et al., 2018), musical style transfer (Brunner et al., 2018). (Maezawa, 2018) employed conditional VAE for expressive music performance as we did in this work. However, the latent vector in VAE was generated in note-level whereas our model encodes an entire performance with VAE so that a single latent vector can represent the performance style throughout the piece. There has been research on automatic generation of expressive performance using data-driven approach including neural networks, which is well summarized in (Cancino-Chacón et al., 2018) but they implemented limited performance elements. For example, they inferred only dynamics (Malik & Ek, 2017), neglected tempo change (Lauly, 2010; Giraldo & Ramirez, 2016), assumed that the melody is always in high pitch (Flossmann et al., 2013; Kim et al., 2013), or used normalized tempo (Grachten & Cancino Chacón, 2017). Our model aims to implement full-blown performance elements.

Employing a graph-like connection for music data was also utilized in generating expressive performance for jazz music (Moulieras & Pachet, 2016). But the system was limited to handling monophonic melody only and the graph connection was used to capture different features of notes in monophonic melody. This is fundamentally different from our proposed graph model than handles polyphonic music.

### 6. Experiment

#### 6.1. Data

Since there was no available public data set for our task, we created our own data set by collecting pairs of score in MusicXML and corresponding performances in MIDI. We collected the score files from MuseScore\(^1\) and Musicalion\(^2\), and the performance MIDI files from Yamaha e-competition data\(^3\). The performance data consists of human piano performances of classical music from the Baroque to contemporary music, which were recorded by a computer-controlled piano (Yamaha Disklavier) during the competitions. This dataset has been widely used in music generation tasks (Huang et al., 2019; Hawthorne et al., 2019). To make note-level score and performance data pairs, XML-to-MIDI matching is required. We utilized an automatic alignment algorithm (Nakamura et al., 2017) to synchronize the score XML to performance MIDI in note-level.

In our data set, there are 227 pieces by 16 composers with 1061 performances and 3,606,930 notes, when we count notes from different performances of a single piece separately. We split the data set into train, validation, and test set so that each set has the size of approximately 8:1:1 in terms of the number of pieces, performances, and notes. We also considered the composer distribution in the split. Since the result of the performance rendering is mostly dependent on the characteristics of the piece, i.e., score condition, we split the data set in piece-wise so that each piece only belongs to one of the subsets.

\(^1\)https://musescore.com/sheetmusic
\(^2\)https://www.musicalion.com/
\(^3\)http://www.yamahaden.com/midi-files
6.2. Comparative models

We set up several comparative models to evaluate the performance of our proposed model. For the comparison, we implemented the VAE version of our previous work based on HAN (Jeong et al., 2018) and its modified versions. The HAN model consists of score encoder using HAN and a performance decoder. The score encoder module employs note-wise and voice-wise LSTM for note-level representations and HAN for beat-level and measure-level representations. All the LSTM layers are bidirectional in the score encoder. One of the modifications as a Baseline (BL) omits HAN and voice-wise LSTM, thereby having only note-wise LSTM for score encoding. Another modification is replacing note-wise and voice-wise LSTM with GGNN. This is termed as G-HAN.

Table 1 summarizes the architectures of the compared models. BL, HAN and G-HAN models have the same architecture in the performance encoder and decoder. For the performance encoder, we use note-level LSTM encoder for a concatenated input \((C, Y)\) and measure-level attention LSTM. The decoder is composed of a single-layer auto-regressive unidirectional LSTM. To employ the LSTM structure, the input notes were represented as an 1D sequence using the time position and pitch of individual notes.

### 6.3. Training

We trained all models using the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.0003, weight decay of 1e-5, and dropout of 0.1. For the training batch, the notes were sliced at the end of measure that include the 400-th note. We tried both classification and regression for the output, but the output quality was better in regression. Therefore, we used mean square error for the loss, and the output features were all normalized with \(\mu = 0, \sigma = 1\). The loss was calculated note-wise except the tempo loss, which was calculated in beat-wise. Since human performances can include missing notes compared to the music score, some of the input notes \(X\) do not have the corresponding output feature \(Y\). These non-matched notes were excluded in loss calculation. The number of parameters of each model was about 4M.

Training an RNN-based VAE model is a challenging problem because the latent vector \(z\) can be by-passed in the decoder RNN (Bowman et al., 2016). This is especially more crucial to our model because the decoder takes not only latent vector \(z\) but also score information as a condition. As (Widmer et al., 2003) pointed out, there are commonalities between human performances of the same music. Therefore, the model can reproduce reasonable performance \(Y\) without the latent vector \(z\), i.e., neglecting the style of the specific performance. Therefore, we employed a KLD weight annealing, which starts from zero up to 0.02 or 0.003, 0.0003 depending on the model.

<table>
<thead>
<tr>
<th>Model</th>
<th>Tempo</th>
<th>Vel</th>
<th>Dev</th>
<th>Pedal</th>
<th>KLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>BL</td>
<td>0.2721</td>
<td>0.6011</td>
<td>0.7678</td>
<td>0.8056</td>
<td>2.2581</td>
</tr>
<tr>
<td>HAN</td>
<td>0.2380</td>
<td>0.6290</td>
<td>0.7938</td>
<td>0.7681</td>
<td>13.666</td>
</tr>
<tr>
<td>G-HAN</td>
<td>0.2785</td>
<td>0.6212</td>
<td>0.7705</td>
<td>0.8092</td>
<td>7.1113</td>
</tr>
<tr>
<td>Proposed</td>
<td>0.2379</td>
<td>0.5877</td>
<td>0.7978</td>
<td>0.7544</td>
<td>3.7247</td>
</tr>
</tbody>
</table>

### 6.4. Evaluation

#### 6.4.1. Reconstruction Loss

One of the metrics for the quantitative evaluation in performance modeling is the mean square error of the output features between a “target” human performance and the generated one (Cancino-Chacón et al., 2018). Unlike other generative tasks, performances of the same piece are easily comparable by examining how they performed the notes in the same context with different performance features. Since there are many valid ways for playing the given piece, it might be arbitrary to directly compare the generated performance with a “target” performance. But in our case, the comparison is reasonable because our system encodes the “style” of given target performance and generates complete performance for the encoded style.

We measured the reconstruction loss in terms of the mean square error of each output features and KL divergence of \(p(z|x)\) using our score input and corresponding human performances in the test set, which is the same definition for calculating loss in the training and validation steps. For a given score and performance pair in the test set, the model encoded the score and performance style of the piece and reconstructed the output. The result is shown in Table 2. The errors are represented in the average of piece-wise MSE loss of 21 pieces in the test set.

The proposed model showed the best results in tempo, velocity and pedal estimation, while the baseline was better in onset deviation and KL divergence. The small loss in the reconstruction error indicates that the model is capable of predicting human-like performance features for given scores even in the unseen data. The high KL divergence of the HAN model indicates that the model worked more like an ordinary autoencoder, rather than limiting the \(p(z|x)\) into the normal distribution of \(\mu = 0\) and \(\sigma = 1\). The divergence could be reduced if we applied higher KLD weight during the training but we failed to find a proper KLD weight without model collapse.
6.4.2. Correlation

Another metric is measuring the correlation between generated performances and human performances, which several previous works used (Cancino-Chacón et al., 2017; Flossmann et al., 2009). As suggested in recent work on evaluating automatic music composition (Yang & Lerch, 2018), we investigated the intra-set correlation in human performances and the inter-set correlation between generated performances and human performances.

We first measured the correlation in beat tempo between different human performances of the same piece. Among the various pieces in the test set, we sorted out the excerpts in which the human performances showed strong commonalities, i.e., \( \min(r) > 0.75 \), where \( r \) denotes the Pearson correlation coefficient. Each piece was cut into excerpts with at least 30 beats included. Among the test set, 28 excerpts satisfied the condition of correlation coefficient. Then, we generated the performance of given pieces with each model, with the \( z \) sampled from \( \mathcal{N}(0, 1) \). We measured the correlation coefficients between the generative performance and each human performance. We counted only the maximum correlation coefficient for each excerpt.

The results are presented in Table 3. The proposed model achieved the highest number of excerpts for \( r > 0.7 \) and \( r > 0.5 \), showing that it generates human-like tempo curve for most various patterns of input score. However, the marginal differences among models indicate that we still need further research on quantitative evaluation for performance modeling.

### Table 3. The result of tempo correlation between the generated performances and human performances in the selected 50 excerpts. \( r \) denotes the maximum value of correlation coefficients between each model performance and human performances. \( r > x \) represents the number of excerpts that satisfied the condition.

<table>
<thead>
<tr>
<th>Model</th>
<th>avg ( r )</th>
<th>( r &gt; 0.9 )</th>
<th>( r &gt; 0.7 )</th>
<th>( r &gt; 0.5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>BL</td>
<td>0.6604</td>
<td>18</td>
<td>29</td>
<td>36</td>
</tr>
<tr>
<td>HAN</td>
<td>0.6943</td>
<td>19</td>
<td>34</td>
<td>40</td>
</tr>
<tr>
<td>G-HAN</td>
<td><strong>0.6994</strong></td>
<td><strong>21</strong></td>
<td><strong>32</strong></td>
<td><strong>41</strong></td>
</tr>
<tr>
<td>Proposed</td>
<td>0.6821</td>
<td>18</td>
<td><strong>35</strong></td>
<td><strong>42</strong></td>
</tr>
</tbody>
</table>

6.4.3. Listening Test

For the qualitative evaluation, we conducted a pairwise listening test to judge the quality of output performance of our model. Among the 21 pieces in the test set, we selected 6 pieces for the listening test considering the composer of the piece. We generated the performance MIDI files of each piece using three different models including BL, HAN, and ISGN (the proposed). The performance style vector \( z \) was sampled from \( \mathcal{N}(0, 1) \).

The generated performances and the human pianist performances in Yamaha E-competition data were recorded in audio by performing the MIDI files with a computer-controlled piano. In the experiment, we used only 30 seconds of each recording\(^4\). We asked the subjects to listen to a pair of performance of the same piece and choose a performance with better musical quality. We recruited a group of subjects among college students for the listening test. The total number of subjects was 40. Each participant evaluated half of all possible combinations and we collected 720 responses in total. Each model was involved in 360 evaluations.

Figure 5 shows the statistics of the listening test. The proposed ISGN model recorded the highest number in total win among the generative models, and also highest win rate when each of generated performance was compared with human performance. However, further investigation is needed because the difference between ISGN and the others was not significant in the total wins, and there is a large fluctuation depending on the piece.

7. Conclusion

We proposed an iterative sequential graph network which combines gated graph neural network and hierarchical attention RNN for modeling expressive piano performance for the given music score. The quantitative and qualitative evaluation showed that the proposed model made a notable improvement compared to the previous models. For future work, we will investigate more on evaluating the expressive performance models quantitatively.

\(^4\)Audio files are available in the supplementary file
Acknowledgement

This research is funded by Samsung Research Funding Center under the project number SRFC-IT1702-12.

References


Graph Neural Network for Music Score Data and Modeling Expressive Piano Performance


