The Power of Fragmentation: A Hierarchical Transformer Model for Structural Segmentation in Symbolic Music Generation

Guowei Wu, Shipei Liu, and Xiaoya Fan

Abstract—Symbolic music generation relies on the contextual representation capabilities of the generative model, where the most prevalent approach is the Transformer-based model. Learning contextual representations are also related to the structural elements in music, i.e., intro, verse, and chorus, which have not received much attention of scientific publications. In this paper, we propose a hierarchical Transformer model to learn multiscale contexts in music. In the encoding phase, we first design a fragment scope localization module to separate the music parts into chords and sections. Then, we use a multiscale attention mechanism to learn note-, chord-, and section-level contexts. In the decoding phase, we propose a hierarchical Transformer model that uses fine decoders to generate sections in parallel and a coarse decoder to decode the combined music. We also designed a music style normalization layer to achieve a consistent music style between the generated sections. Our model is evaluated on two open MIDI datasets. Experiments show that our model outperforms other comparative models in 50% (6 out of 12 metrics) and 83.3% (10 out of 12 metrics) of the quantitative metrics for short- and long-term music generation, respectively. Preliminary visual analysis also suggests its potential in following compositional rules, such as reuse of rhythmic patterns and critical melodies, which are associated with improved music quality.

Index Terms—Symbolic music generation, Transformer-based model, structural segmentation, multiscale attention.

I. INTRODUCTION

SYMBOLIC music generation (SMG) refers to generating continuation from initial notes. It has received great attention with the success of deep learning [1], [2]. Music can be seen as a sequence of notes in time. A musical generative model should be able to refer to the context of note representations, as required for natural language models. Hence, language models, such as the autoregression model, are prevalently used for music generation. A representative model is PerformanceRNN [3], an LSTM-based recurrent neural network designed to model polyphonic music with complex dynamics. It performs well in generating short music (∼30 seconds). However, generating long music sequences (≥4 minutes) is still a challenge because errors accumulate as the length of the sequence increases. Therefore, the performance of current music generation models degrades rapidly when the reference length exceeds the default range, i.e., the output is restricted to a maximum length.

Many efforts have been devoted to maintaining long-term relevance with reasonable computational complexity. In particular, the Transformer model [4] has helped advance state-of-the-art (SOTA) in symbolic music generation. For instance, Google researchers [5] have proposed MusicTransformer, which can deal with longer sequences (∼4096 notes) with optimized intermediate memory occupation. MuseNet [6] is a Transformer-based model that generates 4-minute music and discovers patterns of harmony, rhythm, and style in long-form music. Transformer-GANs [7] introduce a pretrained discriminator that uses adversarial loss to complement the negative log-likelihood objective, enabling improvement in synthesizing minute-long compositions. Despite recent improvements in long-term music generation, existing approaches fail to learn music structure effectively. This may be attributed to their incompetence in segmenting structural elements.

Most music is typically structural, such as intro, verse, and chorus, meaning that some attractive melodies are repeated throughout the song. Because of this, some critical factors related to the quality of music are difficult to model [8], for instance, chord progression, rhythm pattern and music style. Thalmann et al. [9] introduced an extension of the musical formalism CHARM for representing reusable musical content and compared various representations of multilevel graph structures with logical constraints for music modeling. In traditional neural networks, hierarchical architectures have been proposed to generate these structured melodies. MusicVAE [10] uses a hierarchical decoder to model structural elements, such as the repetition and variation between measures and sections of a piece of music. MuseGAN [11] consists of three generative adversarial network (GAN) models, called the jamming model, composer model, and hybrid model, for generating multitrack music with their respective temporal dynamics. Therefore, we argue that structural segmentation and section-level contextual learning are the primary challenges in generating realistic long-term music.

In this paper, we propose a hierarchical Transformer model to learn multiscale representations extracted by structural segmentation methods. Specifically, the note sequences are first
We propose an FSL module that divides the music into structural musical sections by chord recognition and region proposal methods. Then, fine decoders are pretrained with multiscale attention through these structural corpora, enabling chord-level and section-level contextual learning. Next, the first few notes and labels are input as activation information to generate sections. We propose a music style normalization (MSN) layer to control the music style of the generated sections. The following aggregation layer and fine-tuned by a coarse decoder. The generated sections are combined by an aggregation layer and fine-tuned by a coarse decoder.

fed into a custom-designed fragment scope localization (FSL) module, which separates them into structural musical sections by chord recognition and region proposal methods. Then, fine decoders are pretrained with multiscale attention through these structural corpora, enabling chord-level and section-level contextual learning. Next, the first few notes and labels are input as activation information to generate sections. We propose a music style normalization (MSN) layer to control the music style of the generated sections. The following aggregation layer and coarse decoder are used to combine the generated sections and fine-tune them at the global scale. An illustrative diagram of the proposed framework is shown in Fig. 1. Our main contributions are as follows:

- We propose an FSL module that divides the music into structural elements by custom-designed chord recognition and musical section region proposal methods. The ablation studies further analyzed the utility of the FSL module with different settings.
- We propose a multiscale attention mechanism to learn music representations at the note, chord, and section levels. The visual evaluation shows its superiority in melody reuse.
- We propose a hierarchical architecture to alleviate the error accumulation problem in long-term music generation and an MSN layer to control musical style based on mutual information optimization. Experimental results show that our model achieves SOTA performance on two open datasets, quantized by metrics evaluating style consistency.

II. RELATED WORKS

A. Transformer Baselines

Transformer baselines have been the typical solution for sequence modeling [12], [13] due to their advantages in contextual representation. However, applying the Transformer baseline to the symbolic music generation (SMG) task is computationally prohibitive since the complexity increases quadratically with the sequence length [14]. The sparse Transformer [15] introduces a sparse factorization of the attention matrix that reduces the complexity from $O(L^2)$ to $O(L\sqrt{L})$. Many Transformer models are dedicated to better modeling longer-term dependencies. For instance, Transformer-XL [16] learns dependencies beyond a fixed length by a segment-level recurrence mechanism and a positional encoding scheme. Transformers have various applications in music generation tasks. Jin et al. [17] used cross-track Transformers to learn the information among the tracks of different musical instruments and proposed a music theory-based reward network to reliably adjust the generated music. Some studies incorporate audio with other modalities, for instance, visual information, to generate music. A representative work is the Foley music system [18], which learns to generate music from videos about people playing musical instruments using the Transformer framework and music gesture [19].

B. Music Structure Segmentation

Structure segmentation is commonly used for modeling long-form music, which can be approached with hierarchical architecture. For instance, McCallum et al. [20] used a convolution neural network for unsupervised training of music segmentation, aiming to detect the boundaries of music segments in audio music. In a music annotation work [21], convolution networks and recurrent networks were hierarchically combined to learn music structure. Wu et al. [22] present a PopMNet model to generate pop music melodies based on melody structures defined by repetition and sequence pairwise relations. Dai et al. [23] proposed a hierarchical model to generate a full-length melody guided by a long-term repetitive structure and achieved near-human performance in melody generation approximately half
the time. Berardinis et al. [24] segment music hierarchically and compute quantitative metrics on the resulting hierarchies to characterize the music decomposition process into its structural components. These works suggest that structure segmentation is beneficial for SMG tasks.

C. Music Style Transfer

Musical styles are perishable and inimitable but can be reserved through quantification-based style transformation. An effective work [25] generated new accompaniments for songs by performing style transfer in the symbolic representation. Furthermore, Lim et al. [26] proposed continuous style embedding to the general formulation of a variational autoencoder to allow users to be able to condition the style of the generated music. Instance normalization methods are often used for style transfer. For instance, Huang et al. [27] proposed an adaptive instance normalization approach to achieve flexible style control. Mutual information (MI) is also commonly used for style transfer. Chawla et al. [28] proposed a model for formality style transfer, which maximized the MI between original and target styles as the training objective and achieved better performance. Inspired by these studies, we utilized an MI-based method to maintain a consistent style in long-form music.

III. THE PROPOSED MODEL

The key challenge for the SMG task is to generate long-term music with structural relevance and consistent style. We propose to address this challenge by recognizing the structural elements (chord and section) of the note sequence prior to the generation task. In this section, we begin with a description of the proposed fragmentation scope localization (FSL) module for structural element recognition. In the encoding phase, the encoder embeds note sequences and structural elements into a multiscale representation for a pretrained fine-grained decoder. During the decoding phase, we propose to generate music section by section through a hierarchical architecture from the recognized structural elements. A music style normalization (MSN) layer is also proposed for style constraints between the generated sections. Additionally, a coverage analysis of the attention patterns of the proposed model and baseline models is presented. All symbols used in this paper are summarized in Appendix A.

A. Structural Element Recognition: Fragmentation Scope Localization

The FSL module is designed for the structural element (chords and sections) recognition task on symbolic music sequences. The details of the FSL module are illustrated in Fig. 1(a).

Chord recognition: A chord in music is a group (typically three or more) of notes that sound together and serve as a basis of harmony. In tonal music rules, a chord can also be a group of notes that appear dispersed in a bar. Therefore, we extracted the chord feature, called the chord profile, by counting the notes of each bar. The notes (transferred to the same octave) were mapped to a 12-dimensional vector, usually called twelve-tone equal temperament, which is a set of pitch classes \([C_1, Db_2, D_3, Eb_4, E_5, F_6, F_{7/2}, G_8, Ab_9, A_{10}, Bb_{11}, B_{12}]\).

We counted the 47 most common forms of chord composition (major triads, minor sixth chords, minor-major seventh chords, suspension chords, etc.). Single notes that cannot form a chord are denoted as type 48. For example, the \(C\) triad chord consists of three notes: a root note \(C\), intervals of a third \(E\), and a fifth above the root note \(G\). Thus, the \(C\) triad chord can be represented by \([C, E, G]\) and in twelve-tone equal temperament by \(C_{chord} = C_1 + \text{Triad}_{0,4,7} = [C_1, E_5, G_8]\). The chord profile used in our model is shown in Appendix B.

In our model, chords are identified from MIDI data in two steps:

1. Template matching for complete chord [29];
2. Sequence estimation for unidentified bars [30, 31].

More specifically, we use the template matching method to match the same chords as in the chord profiles. These matching chords have neither missing notes nor other ornamental notes. Then, we construct an n-gram sequence to represent unidentified bars and use a nonlinear classifier to calculate the correlation, which can determine the probability of the chord class. The result of chord recognition is demonstrated in Fig. 2.

A series of chords is called a chord progression, and it is periodically reused in sections of the same label. For example, in Fig. 2, bars 19, 20, and 21 use the same chord as bars 2, 3, and 4. The construction of these specific chord progressions is one of the goals we hope to achieve with our model.

Fragment region proposal: We use the region proposal method to selectively search for music sections. A common strategy to obtain the regional proposals is to slide the candidate windows on the note representation from left to right, which we call the left-to-right (L2R) strategy. Specifically, the coordinate of the left border of the \(m\)-th candidate window \(\omega\) is calculated by \(\sum_{j=1}^{m-1} k_j\). The window size \(k\) is set to be the multiplication of the chord length (16 semiquavers), with \(k \in [8, 16, 32, 64, 128, 256, 512, 1024]\) sizes available, which coincides with the section length distribution in the training dataset. The goal of the L2R strategy is to produce multiple arrangements of candidate regions \(\text{Cand} (x)\), including different positions and lengths. Then, the cls layer selects the appropriate combination of candidate windows with the maximum classification probability.

However, a drawback of the L2R strategy is that error accumulates when the candidate window slides. We design a global
strategy, where the locations of the region proposals are randomly generated, with a padding operation to ensure coverage of the whole sequence with all region proposals. The randomness in the region proposal location can greatly alleviate the error accumulation problem. The comparison of the two alignment strategies is shown in Fig. 3.

Classification layer: After several candidate region proposal operations, the obtained multiple alignments of the candidate window are fed into the cls layer to determine the best combination with the maximum average classification probability. The class probabilities $p_\varphi \in \mathbb{R}^\phi$ predict the label $\varphi \in \phi$ of candidate sections, such as intro, verse, and chorus. Since samples of different classes are unbalanced, we use Gumbel softmax [32] to define the classification loss as follows:

$$
L_{cls} = -\log \frac{\exp ((p_\varphi + \theta_\varphi)/\tau)}{\sum_{\tilde{\varphi}} \exp ((p_{\tilde{\varphi}} + \theta_{\tilde{\varphi}})/\tau)}
$$

(1)

where $\tau$ is a nonnegative coefficient of spread in the Gumbel distribution; The smaller the coefficient, the closer the sample expectation is to the ArgMax function; The larger the coefficient, the more average the sample expectation is. $\theta$ are samples drawn from Gumbel(0, 1). $\tilde{\varphi}$ lists all labels, and $p_{\tilde{\varphi}}$ is the probability of the candidate section for every label. The value of $L_{cls}$ should be close to 0 if the candidate is consistent with the ground-truth label.

Regression layer: To measure the overlap ratio between the candidate and ground-truth sections, we check the percentage of shared and different notes in these paired sections. Let $[\text{Cand}_m, y_m]$ be a training instance pair, where $\text{Cand}_m$ and $y_m$ are the predicted and ground-truth music sections at the $n$-th position. We use the Jaccard similarity coefficient to evaluate the similarity between the predicted and ground-truth scopes. For a predicted window with class $\varphi$, the Jaccard similarity is denoted as $J\text{Jac}_m = |\text{Cand}_m \cap y_m| / |\text{Cand}_m \cup y_m|$. The regression loss function can be calculated by:

$$
L_{reg} = \begin{cases} 
-\log(J\text{Jac}_m), & \text{if } 0 < |J\text{Jac}_m| < 1 \\
0, & \text{if } |J\text{Jac}_m| = 1 
\end{cases}
$$

(2)

A low regression loss indicates a high coincidence between the prediction and the ground truth. Accordingly, the loss function of our FSL module can be represented by:

$$
L_{FSL} = \frac{1}{N_{cls}} \sum_{m=1}^{N_{cls}} L_{cls} + \frac{1}{N_{reg}} \sum_{m=1}^{N_{reg}} L_{reg}
$$

(3)

where $L_{cls}$ and $L_{reg}$ are normalized by $N_{cls}$ and $N_{reg}$, which are the sample numbers in these two layers, respectively. After passing through the FLS layer, the note sequence is segmented into structural sections.

B. Encoding: Multiscale Representation for Pretraining Fine Decoders

During the encoding phase, we pretrain the fine decoders through the proposed multiscale attention, which enables our model to learn musical contexts at different time scales. The details of the multiscale attention mechanism are illustrated in Fig. 1(b).

Multiscale attention: In Transformer-based models, the attention pattern determines the perceptual range. We designed a multiscale attention mechanism for music representation to support learning additional chord-level and section-level context features that are not available in traditional Transformers, as shown in Fig. 4.

Given a music sequence $X = [x_1; x_2; \cdots; x_L]$ with length $L$, the output embedding of note $x$, i.e., $h(x)$, is obtained by a Transformer encoder, denoted as:

$$
h_l = LN(h_{l-1} + FFN(\text{Attention}_{MS}(Q, K, V)))
$$

(4)

where $Q$, $K$ and $V$ represent the query, key and value vectors in the Transformer baseline, respectively; $l$ represent the number of encoding layers; $LN$ and $FFN$ represent the layer normalization operation and feed-forward network, respectively; $\text{Attention}_{MS}$ represents the multiscale attention. Scale pattern $S_i \in \{N_{note}, N_{chord}, N_{section}\}$ was parameterized to determine the reference context of the designed multiscale attention, which can be calculated as:

$$
\text{Attention}_{MS} = \begin{bmatrix} 
\text{head}_1(S_{note}), \ldots, \text{head}_N(S_{note}) \\
\text{head}_1(S_{chord}), \ldots, \text{head}_N(S_{chord}) \\
\text{head}_1(S_{section}), \ldots, \text{head}_N(S_{section}) 
\end{bmatrix}
$$

(5)

where $N$ is the number of heads at each scale. The attention head is the weighted sum of the scaled dot-product of the input
vectors, calculated by:

$$\text{head}(S_i) = \text{Softmax} \left( \frac{Q_S K_S^T}{\sqrt{D}} \right) V_S,$$

(6)

where $D$ represents the dimension of query, key and value vectors. $Q_S$, $K_S$, $V_S$ are transformed by the weight matrices $W_Q, W_K, W_V$:

$$\begin{bmatrix} Q_S \\ K_S \\ V_S \end{bmatrix} = \begin{bmatrix} W_Q & W_K & W_V \end{bmatrix} \odot X(S_i) + \begin{bmatrix} 0 \\ P_S \\ P_S \end{bmatrix},$$

(7)

where $X(S_i)$ defines the references at the $i$ scale, corresponding to note, chord and section. $P_S$ represent the position embeddings (PEs) [33] for the scale pattern $S_i$. We use relative PEs in the note and chord scales and absolute PEs in the section scale.

Through the multiscale representation, fine decoders with maximum length [256, 512, 768, 1024] were pretrained by the masked language model (MLM) loss function. For the sequence $X$, the fine decoder predicts $T$ masked notes among them. We minimize the following MLM loss:

$$\mathcal{L}_{\text{MLM}}(X_{\text{II}} | X_{-\text{II}}) = -\frac{1}{T} \sum_{t=1}^{T} \log p(X_{\text{II}} | X_{-\text{II}}),$$

(8)

where $X_{\text{II}}$ and $X_{-\text{II}}$ denote the masked and unmasked notes, respectively.

### C. Decoding: Hierarchical Architecture With Music Style Normalization

During the decoding phase, the proposed hierarchical architecture, a bottom-up model, aims to generate music at scales from fine to coarse. To generate more realistic music, an MSN layer was used to normalize the musical style among the generated sections. The generated sections were then aggregated through an aggregation layer to obtain the fine-tuned music sequence at a coarse scale. The decoding phase is shown in Fig. 1(c).

At the bottom of the hierarchical architecture, we use fine decoders pretrained with different maximum lengths to generate music sections. For example, the $m$-th section $\text{Section}_m$ is decoded with the activation information, which usually uses the first few notes and a label:

$$G_m(x) = \text{Decoder}_{\text{fine}}(\text{Section}_m(x_{1:r}); \varphi_m)$$

(9)

where $G_m(x)$ represents the $m$-th generated section; $\varphi_m$ indicates the target label of the generated section, i.e., intro, verse, chorus, etc. Given the start notes $x_{1:r}$, the next output note is obtained by:

$$x_{r+1} = LN(x_r + FFN(a_{r+1}(x_{1:r})))$$

(10)

where $a_{r+1}$ is the attention block at the $r + 1$ position, which calculates a weighted score from previous notes.

**Music Style Normalization:** To enhance the style consistency between the original and generated sections, we design an MSN layer in which the musical style is quantified by a variable $\mathbf{z}$ with the same dimensions as the note embedding. Through the MSN layer, the style variable $z$ is transferred to the generated sections, which can be expressed as $z \rightarrow z'$ [27]. The MSN layer normalizes the mean and standard deviation of note pitches of the generated sections and maps the style to the output. For the $m$-th generated section, this process can be calculated by:

$$\text{MSN}(G_m(x), z_m) = \gamma_z \left( \frac{G_m(x) - \mu(z_m)}{\sigma(z_m)} \right) + \beta_z,$$

(11)

where $\gamma_z$ and $\beta_z$ are the scaling and translation parameters, respectively, which are calculated independently for each class. $\mu(z_m)$ and $\sigma(z_m)$ represent the mean and standard deviation of note pitches in the $i$-th generated section, respectively.

At the top of the hierarchical architecture, these sections are aggregated into a long sequence by the concatenation operation $\vee$. The coarse decoder is used to learn the global contextual reference from the merged sequence. The coarse decoding process can be represented by:

$$G(x) = \text{Decoder}_{\text{coarse}} \left( \vee \left( \sum_{m=1}^{M} \text{MSN}(G_m(x), z_m) \right) \right)$$

(12)

where $G(x)$ stands for the output sequence. The coarse decoder $\text{Decode}_{\text{coarse}}$ uses multiscale attention with the same length as the encoder.

**Multitask Loss:** We fix the parameters of fine decoders to further train the coarse decoder. The loss function of the decoding phase consists of two terms:

1) $\mathcal{L}_{\text{MLM}}$ for the note prediction task;
2) $\mathcal{L}_{\text{style}}$ for the music style normalization task.

$$\mathcal{L}_{\text{Decoding}} = \mathcal{L}_{\text{MLM}}(G(x)) + \lambda \mathcal{L}_{\text{style}} (x, G(x))$$

(13)

where hyperparameter $\lambda$ is used to balance the magnitude of the loss terms.

To improve the style similarity between the sample and observations, we maximize their mutual information (MI) in the optimizer. Mathematically, we can examine the KL-divergence $\delta_{KL}$ between the distribution of joint probability and the marginal probability to determine the independence of the two variables. Thus, we can use the variational information maximization method [34] to estimate the mutual information $I(x; G(x))$ between the sample and observations by instantiating an approximated distribution $q(x | G(x))$:

$$I(x; G(x)) = \mathbb{E}_{p(x,G(x))} \left[ \log \frac{q(x | G(x))}{P(x)} \right]$$

$$+ \mathbb{E}_{p(G(x))} \left[ \delta_{KL}(P(x | G(x)) || q(x | G(x))) \right]$$

(14)

where $P(x)$ is the distribution of training data; $P(G(x))$ is the distribution of generated samples. We empirically deduce that $\delta_{KL} > 0$; then, $I(x; G(x))$ can be transformed to:

$$I(x; G(x)) \geq \mathbb{E}_{p(x,G(x))} \left[ \log q(x | G(x)) \right] + H(X)$$

(15)

where $H(X)$ is the differential entropy of $X$. We can derive an initial lower bound, which is tight when $q(x | G(x)) = P(x |
For pretraining the fragment scope localization, Grooving pattern (Pitch range) represents the observation of the generation. The range and location of section-level attention are adaptive to the corresponding section in each music composition rather than hard-coded in advance. Therefore, section segmentation can be thought of as a dynamic assignment of full attention at a local scale, which decreases the computational cost while maintaining the long perception range of full attention.

**IV. Experiments**

We used the same methods as MusicTransformer to process MIDI data. The structural information of musical sections is automatically extracted from the music score, which is obtained by the GuitarPro software, a music recording tool. The annotation extraction method is detailed in Appendix C.

**Datasets:** For pretraining the fragment scope localization (FSL) module, we collected a total of 300 scores with 3,188 sections, called the GuitarPro dataset. After the pretraining process, we trained and tested our hierarchical Transformer on two canonical music corpora:

1. J.S. Bach Chorales [35];
2. Maestro v3.0.0 [36].

Based on the average length of samples, we tested the performance of the models for short-term and long-term music generation on the J.S. Bach and Maestro datasets, respectively.

**Evaluation metrics:** A set of commonly used quantitative metrics scrutinizes the generated samples from different perspectives. We apply these metrics to the training data and compare them with the results generated by the comparison model. These metrics include four aspects:

1. Probability-based metrics [37]: **PPL**, i.e., Perplexity, evaluates the conditional probability of notes in the sample. It is a commonly used metric for evaluating the distribution of the generated samples and reflects the degree of matching with the training set.
2. Pitch-based metrics [7, 38]: **UPC** (Unique pitch classes) represents the number of pitch classes per bar; **TUP** (Total number of unique pitches); **PR** (Pitch range) represents the average difference between the highest and lowest pitches in semitones; **APS** (Average pitch by semitone) represents the average semitone interval between two consecutive pitches. These metrics give a sense of the pitch diversity (UPC and TUP) and pitch variation (PR and APS) of the music samples.
3. Rhythm-based metrics [7, 39]: **ISR** (ratio of nonzero entries); **ISR** indicates the rhythmic tightness of the sample on the global time scale. **PRS** (ratio of pitches with time steps greater than 4 to the total number of pitches in a sample); **PRS** reflects the percentage of notes of different lengths, such as dichotomous notes and quarter notes, in the sample. **IOI** (Interonset-interval) represents the time between two consecutive notes; An IOI value closer to that of the real sample indicates that the generated sample has a rhythmic interval that is more similar to the original sample from which it was generated. **GS** (Grooving pattern similarity); If a sample possesses a clear rhythm, the grooving patterns between adjacent bars should be similar, thereby producing high GS scores. Rhythm-based metrics focus on the consistency of note length on the time step.
TABLE II
COMPARISON OF MODEL PERFORMANCE WITH FOUR OTHER MODELS, TESTED ON J.S.

| Models                  | PPL  | UPC  | ISR  | PRS  | TUP  | PR   | APS  | TOI  | PCH  | GS   | CPI  | SI   |
|-------------------------|------|------|------|------|------|------|------|------|------|------|------|------|
| Training Set (J.S. Bach)|      |      |      |      |      |      |      |      |      |      |      |      |
| PerformanceRNN          | 1.96 | 7.5  | 0.853| 0.446| 9.3  | 28.4 | 3.901| 0.05 | 2.665| 0.999| 0.999| 0    |
| Transformer-XL          | 1.846| 5.742| 0.785| 0.383| 34.859| 40.136| 10.763| 0.074| 2.531| 0.919| 0.990| 0.008|
| MusicTransformer        | 1.833| 5.768| 0.792| 0.386| 36.069| 42.542| 10.257| 0.075| 2.535| 0.916| 0.993| 0.009|
| Transformer-GANs        | 1.789| 5.9  | 0.807| **0.423**| 34.817| 40.41 | **10.993**| **0.072**| 2.537| **0.921**| **0.987**| **0.009**|
| Hierarchical model (Ours)| **1.716**| **7.061**| **0.819**| **0.412**| **25.194**| **34.597**| **10.982**| **0.124**| **2.944**| **0.872**| **0.994**| **0.01**|
| Training Set (Maestro v3.0.0)|      |      |      |      |      |      |      |      |      |      |      |      |
| PerformanceRNN          | 1.403| 3.99 | 0.682| 0.268| 5.52 | 10.71| 3.98 | 0.999| 1.912| 0.880| 0.322| 0.014|
| Transformer-XL          | 1.188| 6.121| 0.796| 0.305| 53.53| 61.54| 11.18 | 0.067| 2.639| 0.931| **0.957**| 0.083|
| MusicTransformer        | 1.167| 6.16 | 0.833| 0.499| 55.55| 62.72| **11.69**| 0.071| 2.539| 0.919| 0.990| 0.095|
| Transformer-GANs        | 1.145| 6.136| 0.814| 0.298| 56.17| 63.93| 11.95 | 0.091| 3.51 | 0.919| 0.997| 0.129|
| Hierarchical model (Ours)| **1.063**| **6.716**| **0.815**| **0.429**| **57.26**| **67.62**| **11.55**| **0.087**| **2.985**| **0.894**| **0.987**| **0.191**|

Bach and maestro datasets. For each metric, the closer to the training set the value is, the better. The best performance is highlighted in bold.

4) Structure-based metrics [39]: PCH (Pitch-class histogram) represents an octave-independent representation of the 12-dimensional pitch content; CPI (Chord progression irregularity); SI (Structure indicators). These metrics describe the structural events. For instance, PCH measures pitch stability over shorter time scales; CPI measures harmony consistency within a sample; SI detects the presence of repeated structures within a specific time range. A large value of these metrics suggests the presence of multiplexed note patterns, chords or melodies within a detection range.

A. Symbolic Music Generation

We compared the performance of the present hierarchical model with other outstanding music generation models that had achieved SOTA performance at the moment. The minimum unit of the token in note representation is the semiquaver note. All models were retrained with the same training sets. We ran the results 5 times and reported the average values. The training steps are detailed in Appendix D.

As shown in Table II, Transformer-GANs achieved better performance in a few metrics for short-term music generation (J.S. Bach dataset). This may be attributed to the use of adversarial training, which can reduce the distributional discrepancy between the generated and real data. Our model achieved similar performance in short-term music generation, as shown by the results in J.S. Bach. However, its superiority in long-term music generation is quite obvious, as demonstrated by the results in the Maestro dataset. Specifically, our model performs best on the Maestro dataset for all metrics except APS and CPI. The best PPL indicates that the music generated by our model shows the highest similarity to real music in terms of data distribution. Our model also shows superiority in UPC and TUP. Therefore, our model generates samples with tonal diversity similar to that of real samples. For ISR, if it has a low value, some unpleasant long pauses will be heard. Our model can generate samples that resemble the training sample in this aspect. Moreover, our model is capable of generating long music with a wide range of tones and stable rhythm, as indicated by PR and PCH, respectively.

More excitingly, our model achieves the best SI, suggesting its advantage in reusing structural melody.

Error accumulation analysis: We tested the accumulation of errors by showing the value of PPL with respect to section numbers and output sequence length. As shown in Fig. 6, the PPL increases rapidly when the section number exceeds 6 for MusicTransformer and Transformer-GANs, while staying stable for our hierarchical model. Similarly, the PPL for MusicTransformer and Transformer-GANs increases significantly when the output length exceeds 1500, while the drop is much less for our hierarchical model. This indicates that our model can better avoid the error accumulation problem compared to other Transformer-based models.

B. Ablation Studies

The improvements in the different configurations of the FSL and MSN layers were analyzed by ablation studies. We performed all the ablation studies on the J.S. Bach and Maestro
TABLE III
ABALATION STUDIES OF FSL AND MSN LAYERS ON J.S.

| Baseline               | Configuration         | PPL  | UPC  | ISR  | PRS  | TUP  | PR   | APS  | IOI  | PCH  | GS   | CPI  | SI   |
|------------------------|-----------------------|------|------|------|------|------|------|------|------|------|------|------|------|
| Training Set (J.S. Bach) | -                     | -    | -    | -    | -    | -    | -    | -    | -    | -    | -    | -    | -    |
| Local Attention        | Global, MSN           | 1.724| 7.123| 0.817| 0.419| 24.859| 34.699| 11.237| 0.124| 2.934| 0.922| 0.993| 0.013|
| Sparse attention       | Global, MSN           | 1.782| 6.979| 0.815| 0.417| 24.812| 34.832| 11.533| 0.125| 2.92 | 0.930| 0.985| 0.011|
| Multiscale Attention   | [LR, None]            | 1.917| 5.565| 0.802| 0.391| 22.178| 41.955| 6.983 | 0.084| 2.566| 0.879| 0.976| 0.011|
| Multiscale Attention   | [LR, MSN]             | 1.729| 6.939| 0.815| 0.415| 24.68  | 34.704| 11.32 | 0.127| 2.915| 0.874| 0.997| 0.011|
| Multiscale Attention   | [Global, None]        | 1.733| 6.449| 0.789| 0.402| 20.021| 38.186| 6.883 | 0.074| 2.501| 0.889| 0.998| 0.003|
| Multiscale Attention   | [Global, MSN]         | 1.716| 7.061| 0.819| 0.412| 25.194| 34.597| 10.982| 0.124| 2.944| 0.872| 0.994| 0.010|
| Training Set (Maestro) | -                     | -    | -    | -    | -    | -    | -    | -    | -    | -    | -    | -    | -    |
| Local Attention        | Global, MSN           | 1.184| 6.361| 0.786| 0.411| 52.924| 61.422| 10.651| 0.090| 2.817| 0.942| 0.743| 0.089|
| Sparse Attention       | Global, MSN           | 1.086| 6.626| 0.834| 0.436| 73.965| 76.011| 10.767| 0.082| 2.863| 0.892| 0.998| 0.171|
| Multiscale Attention   | [LR, None]            | 1.108| 6.516| 0.825| 0.44  | 76.0   | 77.7   | 11.4  | 0.087| 2.949| 0.882| 0.997| 0.093|
| Multiscale Attention   | [LR, MSN]             | 1.042| 6.795| 0.832| 0.435| 74.17  | 76.93  | 11.09 | 0.083| 2.987| 0.887| 0.992| 0.156|
| Multiscale Attention   | [Global, None]        | 1.181| 6.751| 0.824| 0.439| 66.91  | 73.0   | 11.19 | 0.084| 2.988| 0.892| 0.994| 0.139|
| Multiscale Attention   | [Global, MSN]         | 1.063| 6.716| 0.815| 0.429| 97.36  | 67.62  | 11.35 | 0.087| 2.985| 0.894| 0.987| 0.191|

Bach and maestro datasets. The values of the metrics are expected to be close to that for the training set. The best performance is highlighted in bold.

The results are demonstrated in Table III. In ablation studies, the multiscale attention baseline with the [Global, MSN] configuration achieved more than 5 best metrics on both datasets. Even in less-than-optimal metrics, this configuration achieves better performance than at least half of the baselines in comparison. For the alignment strategy in the FSL module, most baselines achieved better results with the global strategy. The use of the MSN layer resulted in better pitch diversity in the generated music (TUP, PR, and APS are great improvements), validating its utility in musical style control. These results demonstrate the effectiveness of the FSL module and MSN layer, which provides structural segmentation and style transfer abilities for the symbolic music generation task.

C. Visual Analysis

As a supplement, we analyzed the musical structure of the generated samples. We visually compare the generated music and the original music in the form of MIDI diagrams. Fig. 7 shows examples of generated music fragments that can reflect the ability of our model in terms of melody reuse. Our model learns to reuse self-melodies not only in adjacent chords (Fig. 7(a)) but also in different sections (Fig. 7(b)). It seems that the model learns to reuse the same rhythmic pattern to compose, i.e., use the same note rules and durations within a period. Such reuse makes the generated music more structured and appropriated paused. Our model also well imitates the real music by bringing some important melodies into the generated samples (Fig. 7(c)). These phenomena illustrate that some factors related to musical quality, such as rhythmic patterns and musical styles, are addressed to some extent in our model.

Furthermore, samples generated by our model and the other three comparison models are also presented under the same experimental conditions, as shown in Fig. 8. In the sample generated by Transformer-XL, almost all notes are the minimum unit, making the composition sound piecemeal and monotonous. The sample generated by MusicTransformer shows a rhythmically inconsistent note density (see the purple box in Fig. 8) and a monotonous root note throughout almost the whole sample. All notes in the samples generated by Transformer-GANS are large in timescale (longer duration), in higher octaves (higher pitches), and rigid in musical style. In our model, both of our randomly selected samples yield reasonable pitch trends and stable rhythms. These samples follow many compositional rules.
such as the use of various bass tones and smooth melodies, that make them realistic.

V. DISCUSSION AND CONCLUSION

We propose a hierarchical Transformer model to generate music with structural sections, whose advantage is learning contextual representation at multiple scales. We quantitatively evaluated the samples generated by our model from several aspects using a set of metrics, including a probability-based metric (PPL), pitch-based metrics (UPC, TUP, PR and APS), rhythm-based metrics (ISR, PRS, IOI and GS), and structure-based metrics (PCH, CPI and SI). The results showed that our model is capable of generating long-term music that better resembles real music than contemporary models: Specifically, (1) our model achieves the best PPL, indicating a more consistent note distribution of the generated samples with the training set. (2) Our model performs best in terms of the pitch-based metrics (UPC, TUP and PR), suggesting its ability to maintain pitch diversity, although the APS is slightly lower. (3) Our model also shows an advantage in maintaining a steady rhythm in long-term music, as shown in rhythm-based metrics. (4) The music generated by our model better follows compositional rules in terms of pitch regularity (PCH) and structural reuse (SI). Ablation studies also show the effectiveness of the proposed fragment scope localization and music style normalization modules.

Another advantage of our model is that our hierarchical architecture dramatically reduces decoding time compared to MusicTransformer and Transformer-GANs in the SMG task. Given the sequence length of $L$, the computational complexity of the Transformer baseline using sparse attention is $O(L \log L)$. As a comparison, our hierarchical architecture reduces the computational complexity to approximately $O\left(\frac{L}{m} \log \left(\frac{L}{m}\right)\right)$, which can be attributed to noncomplete concatenation and parallel decoding. Table IV shows the computational complexity of contemporary canonical models.

The visual evaluation made us aware of some shortcomings that we did not address. In some cases, the segmentation operation of the FSL module may lose its effectiveness for dealing with a composition with rhythmic variations, such as in Beethoven’s Piano Sonata No. 30. This problem can be improved by self-adapt resizing the window length according to the note density of candidates. Additionally, the chord arrangement in the generated samples is not as regular as that of the human composition, which leads to difficulty in obtaining an approximate chord progression for the same starting notes. This can be improved by adding a chord generation module in subsequent work.

In the future, we plan to design a multitask model to improve the performance of chord progression, rhythm pattern, and musical style. We also want to explore other strategies, for instance, curriculum learning, diffusion models, and polyphony modeling, in music generation tasks.

APPENDIX

A. Chord Profiles

We investigate the problem of constructing chord-level representations using weighted N-grams of the chord profile. In our approach, 47 common chord profiles [40] and an additional notation ‘non-chord’ are illustrated in Table V:

B. Symbol Explanation

The symbols we use in this paper are listed in Table VI:

C. Data Preprocessing

To pretrain the cls and reg layers of the fragment scope localization (FSL) module, the location and length of music sections need to be used as labels in supervised learning, which can be extracted independently from the GuitarPro (GTP) software. We proposed a transfer method for automatically marking the section regions from GTP data. The approach is as follows:

1) The GTP data are first converted to the corresponding MIDI data and exported to ASCII code;
2) The ASCII codes are discretized onto a note grid and then serialized by iterating through all the symbols within a time step;
### Table V

**Description of Usual Chord Profiles**

| Chord type | Chord profile |
|------------|---------------|
| 0          | [0, 3, 4, 5, 7, 10] |
| 2          | [0, 2, 4, 6, 7, 10] |
| 3          | [0, 2, 4, 5, 7, 9, 10] |
| 4          | [0, 2, 4, 5, 7, 9, 10] |
| 5          | [0, 2, 4, 5, 7, 9, 10] |
| 6          | [0, 2, 4, 5, 7, 9, 10] |
| 7          | [0, 2, 4, 5, 7, 9, 10] |

### Table VI

**Explanation of Symbols**

| Symbol | Meaning |
|--------|---------|
| \(X, Y\) | Single note in the form of MIDI symbol and its distribution. |
| \(m, M\) | The index and number of predicted sections. |
| \(y\) | The position of the ground-truth section. |
| \(section(x)\) | Sequence of recognized section segments. |
| \(\omega\) | The candidate window. |
| \(k\) | The candidate window size. |
| \(Cand\) | The candidate region extracted by fragment region proposal. |
| \(p(c)\) | The classification probability of \(m\)-th candidate fragment. |
| \(cls, reg\) | Classification and regression layers in our FSL module. |
| \(\theta\) | The samples drawn from \(Gumbel(0.1)\) distribution. |
| \(\phi\) | Predicted musical section class and the set of all classes. |
| \(\psi\) | Any class in all section classes. |
| \(\sigma\) | A non-monotonic spread coefficient in the Gumbel distribution. |
| \(\sigma_{jac}\) | The Jacobian similarity coefficient. |
| \(N_{cls}, N_{reg}\) | The number of samples in the cls and reg layers. |
| \(L_{cls}, L_{reg}\) | The loss function for cls and reg layers in FSL module. |
| \(h(z)\) | The output embeddings of Transformer baseline with input \(z\). |
| \(L_{seq}\) | The sequence length for the Transformer baseline. |
| \(S\) | The set of multiple scales with \(S_{bott}, S_{char}\) and \(S_{section}\). |
| \(Q, K, V\) | Weight matrices for query, key, and value vectors. |
| \(D\) | The dimension of \(Q, K, V\) vector in the Transformer baseline. |
| \(P\) | The positional embeddings configuration. |
| \(N\) | The number of attention heads in each scale. |
| \(G(z), G'(z)\) | Theoretical and observed values of generated sequence. |
| \(\tau\) | The length of start notes, which are used to activate decoders. |
| \(\gamma\) | The style variable recognized by melodic change. |
| \(\beta\) | Scaling parameter calculated by style \(w_{style}\) and category \(\psi\). |
| \(\mu\) | Translation parameter calculated by style \(w_{style}\) and category \(\psi\). |
| \(\sigma\) | The mean of pitches in \(m\)-th generated section. |
| \(\nu\) | The standard deviation of pitches in \(m\)-th generated section. |
| \(\Pi_1, \Pi_2\) | The aggregation operation for generated sections. |
| \(f_\text{mask}\) | The masked and unmasked notes in MLM method. |
| \(T\) | The number of masked notes. |
| \(I_{f}\) | The mutual information between two distributions. |
| \(KL\) | The KL-divergence between two terms. |
| \(\text{Info}\) | Information entropy of samples. |
| \(L_{MLM}, L_{style}\) | MLM loss and our proposed musical style loss. |

---

3) The section labels can be marked by adding a location coordinate since there is a direct correspondence between the sequence location and the grid location. An example of a music section in GTP data with the corresponding ASCII codes and MIDI symbols is illustrated in Fig. 9.

In GTP data, the symbols "∥" and ":" represent the beginning and the end of the section, respectively, and are equivalent to the symbol "||" in ASCII data. These section labels can be synchronously marked in the MIDI data. We compiled 3188 sections from various genres of music, including blues, flamenco, rock, classical, and other music. The diversity of music styles in the music samples enables the FSL layer to realize better and learn to fragment music with various styles. In the FSL layer, the number \(K\) of candidate windows was set to 8, and the window size was selected from \([8, 16, 32, 64, 128, 256, 512, 1024]\) to accommodate music sections with various musical styles.

### D. Training Details

Applying language models to downstream tasks by pretraining and fine-tuning is a common strategy. The Transformer-based model has the problem of error accumulation when dealing with musical sequences with thousands of notes, it is also slow. We first pretrain fine decoders to generate sections at a fine scale and then fine-tune the combined sequence at a coarse scale. The coarse-encoder and fine decoders were trained with different default hyperparameters. While maintaining the overall design of our hierarchical model, we set the maximum length of the coarse decoder to 4096, as in other Transformer-based models in comparison. The maximum length of fine decoders is chosen from \([256, 512, 768, 1024]\), using a carry-up strategy to fit the section length (the smallest one that exceeds the section length). The details of the fine decoder pretraining are shown in Fig. 10.
Then, we fix the weights of the fine decoders and train the music style normalization (MSN) layer and the coarse decoder. The MSN layer is optimized together with the coarse decoder. The music style loss $L_{style}$ is used to optimize the coarse decoder, which is calculated as the mutual information of the input and the renormalized music sections. The training process of the coarse decoder is shown in Fig. 11.

In our experiments, both fine decoders and coarse decoders have 6 hidden layers. We implemented the model in the Tensorflow framework, and the hyperparameters for training were as follows:

1. 1e-03 initial learning rate minimized with 1e-04 weight decay;
2. 100 epochs and 8 batch sizes;
3. 0.2 dropout;
4. Dynamic position embedding;
5. Multiple GPU training and early stopping strategy.

Platform: All experiments were trained/tested on two Nvidia GeForce RTX 2080-Ti 12 GB GPUs.

ACKNOWLEDGMENT

The material includes the GuitarPro dataset, our model-generated samples, and the corresponding visual analysis diagrams. Contact tapliu@mail.dlut.edu.cn for further questions about this work.

REFERENCES

[1] J. Briot, G. Hadjeres, and F. Pachet, Deep Learning Techniques for Music Generation. Berlin, Germany: Springer, 2020.

[2] J.-P. Briot, “From artificial neural networks to deep learning for music generation: History, concepts and trends,” Neural Comput. Appl., vol. 33, no. 1, pp. 39–65, 2021.

[3] I. Simon and S. Oore, “Performance RNN: Generating music with expressive timing and dynamics,” 2017, [Online]. Available: https://magenta.tensorflow.org/performance-rnn

[4] A. Vaswani et al., “Attention is all you need,” in Proc. 31st Int. Conf. Neural Inf. Process. Syst., 2017, pp. 6000–6010.

[5] C. A. Huang et al., “Music transformer: Generating music with long-term structure,” in Proc. 7th Int. Conf. Learn. Representations, 2019, pp. 1–15.

[6] C. M. Payne, “MuSeNet,” 2019, [Online]. Available: https://openai.com/blog/musenet

[7] A. Muhamed et al., “Symbolic music generation with transformer-GANs,” in Proc. AAAI Conf. Artif. Intell., 2021, pp. 408–417.

[8] H. Zhu, Q. Liu, N. J. Yuan, K. Zhang, G. Zhou, and E. Chen, “Pop music generation: From melody to multi-style arrangement,” ACM Trans. Knowl. Discov. from Datas, vol. 14, no. 5, pp. 1–31, 2020.

[9] J. Bai, Y. Du, Y. Wang, S. Kong, J. Gregoire, and C. P. Gomes, “Xtal2DoS: Predicting and explaining ligand-protein interactions,” in Proc. 32nd AAAI Conf. Artif. Intell., 2018, pp. 34–41.

[10] A. Gulati et al., “Conformer: Convolution-augmented transformer for speech recognition,” in Proc. Interspeech, 2020, pp. 5036–5040.

[11] H. Dong, W. Hsiao, L. Yang, and Y. Yang, “MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment,” in Proc. 32nd AAAI Conf. Artif. Intell., 2020, pp. 1–13.

[12] S. Li et al., “Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting,” in Proc. Int. Adv. Conf. Neural Inf. Process. Syst., 2019, pp. 5243–5253.

[13] R. Child et al., “Generating long sequences with sparse transformers,” 2019, arXiv:1904.10509.

[14] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. Le, and R. Salakhutdinov, “Transformer-XL: Attentive language models beyond a fixed-length context,” in Proc. 57th Annu. Meeting Assoc. Comput. Linguistics, 2019, pp. 2978–2988.

[15] C. Jin et al., “A transformer generative adversarial network for multi-track music generation,” CAAI Trans. Intell. Technol., vol. 7, no. 3, pp. 369–380, 2022.

[16] C. Gan, D. Huang, P. Chen, J. B. Tenenbaum, and A. Torralba, “Foley music: Learning to generate music from videos,” in Proc. 16th Eur. Conf. Comput. Vis., 2020, pp. 758–775.

[17] C. Gan, D. Huang, H. Zhao, J. B. Tenenbaum, and A. Torralba, “Music gesture for visual sound separation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 10478–10487.

[18] M. C. McCallum, “Unsupervised learning of deep features for music segmentation,” in Proc. IEEE Int. Conf. Acoust., Speech Signal Process., 2019, pp. 346–350.

[19] Q. Wang, F. Su, and Y. Wang, “A hierarchical attentive deep neural network model for semantic music annotation integrating multiple music representations,” in Proc. Int. Conf. Multimedia Retrieval, 2019, pp. 150–158.

[20] J. Wu, X. Liu, X. Hu, and J. Zhu, “PopMNet: Generating structured pop music melodies using neural networks,” Artif. Intell., vol. 286, 2020, Art. no. 103303.

[21] S. Dai, Z. Jin, C. Gomes, and R. B. Dannenberg, “Controllable deep melody generation via hierarchical music structure representation,” in Proc. 22nd Int. Soc. Music Inf. Retrieval Conf., 2021, pp. 143–150.

[22] J. De Bernardinis, A. Cangelosi, and E. Cautin, “Measuring the structural complexity of music: From structural segmentations to the automatic evaluation of models for music generation,” IEEE/ACM Trans. on Audio, Speech, and Lang. Process., vol. 30, pp. 1963–1976, 2022.

[23] O. Cifka, “Deep learning methods for music style transfer,” Ph.D. dissertation, Institut Polytechnique de Paris, Paris, France, 2021.

[24] Y.-Q. Lim, C. S. Chan, and F. Y. Loo, “ClawNet: Generate music with different musical styles,” IEEE MultiMedia, vol. 28, pp. 83–93, Jan./Mar. 2021.
[27] X. Huang and S. Belongie, “Arbitrary style transfer in real-time with adaptive instance normalization,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 1501–1510.

[28] K. Chawla and D. Yang, “Semi-supervised formality style transfer using language model discriminator and mutual information maximization,” in Proc. Findings Assoc. Comput. Linguistics, 2020, pp. 2340–2354.

[29] K. Lee, “Automatic chord recognition from audio using enhanced pitch class profile,” in Proc. Int. Comput. Music Conf., 2006, pp. 306–313.

[30] J. P. Bello and J. Pickens, “A robust mid-level representation for harmonic content in music signals,” in Proc. Int. Soc. Music Inf. Retrieval, 2005, pp. 304–311.

[31] M. Ogihara and T. Li, “N-gram chord profiles for composer style representation,” in Proc. 9th Int. Conf. Music Inf. Retrieval, 2008, pp. 671–676.

[32] E. Jang, S. Gu, and B. Poole, “Categorical reparameterization with gumble-softmax,” in Proc. Int. Conf. Learn. Representations, 2017, pp. 1–12.

[33] B. Wang et al., “On position embeddings in BERT,” in Proc. Int. Conf. Learn. Representations, 2020, pp. 1–21.

[34] B. Poole, S. Ozair, A. Van Den Oord, A. Alemi, and G. Tucker, “On variational bounds of mutual information,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 5171–5180.

[35] G. Hadjeres, F. Pachet, and F. Nielsen, “Deepbach: A steerable model for bach chorales generation,” in Proc. Int. Conf. Mach. Learn., 2017, pp. 1362–1371.

[36] C. Hawthorne et al., “Enabling factorized piano music modeling and generation with the MAESTRO dataset,” in Proc. 7th Int. Conf. Learn. Representations, 2019, pp. 1362–1317.

[37] F. Jelinek, R. L. Mercer, L. R. Bahl, and J. K. Baker, “Perplexity—a measure of the difficulty of speech recognition tasks,” J. Acoust. Soc. Amer., vol. 62, no. S1, pp. S63–S63, 1977.

[38] L.-C. Yang and A. Lerch, “On the evaluation of generative models in music,” Neural Comput. Appl., vol. 62, pp. 4773–4784, 2020.

[39] S. Wu and Y. Yang, “The jazz transformer on the front line: Exploring the shortcomings of ai-composed music through quantitative measures,” in Proc. 21st Int. Soc. Music Inf. Retrieval Conf., 2020, pp. 142–149.

[40] M. R. Ellis, A Chord in Time: The Evolution of the Augmented Sixth From Monteverdi to Mahler. Aldershot, U.K.: Ashgate Publishing, Ltd., 2010.

Guowei Wu received the Ph.D. degree from Harbin Engineering University, Harbin, China, in 2003. He is currently a Professor with the School of Software, Dalian University of Technology, Dalian, China. He has authored or coauthored more than 100 journal and conference papers. His research interests include embedded real-time systems, cyber-physical systems, smart edge computing, and artificial intelligence.

Shipei Liu received the M.S. degree in communication and information systems from the Wuhan Research Institute of Posts and Telecommunications, Wuhan, China, in 2017. He is currently working toward the Ph.D. degree in software engineering with the Dalian University of Technology, Dalian, China. His research interests include natural language processing, signal processing, and Symbolic Music Generation.

Xiaoya Fan received the Ph.D. degree from Université Libre de Bruxelles, Brussels, Belgium, in 2018. She is currently an Assistant Professor with the School of Software, Dalian University of Technology, Dalian, China. Her research interests include signal processing, EEG modeling, artificial intelligence in medicine and cognition.