

PIANOTREE VAE: STRUCTURED REPRESENTATION LEARNING FOR POLYPHONIC MUSIC

Ziyu Wang¹ Yiyi Zhang² Yixiao Zhang¹ Junyan Jiang¹
Ruihan Yang¹ Junbo Zhao (Jake)³ Gus Xia¹

¹ Music X Lab, Computer Science Department, NYU Shanghai

² Center for Data Science, New York University

³ Computer Science Department, Zhejiang University

{ziyu.wang, yz2092, yixiao.zhang, jj2731, ry649, j.zhao, gxia}@nyu.edu

ABSTRACT

The dominant approach for music representation learning involves the deep unsupervised model family *variational autoencoder* (VAE). However, most, if not all, viable attempts on this problem have largely been limited to monophonic music. Normally composed of richer modality and more complex musical structures, the polyphonic counterpart has yet to be addressed in the context of music representation learning. In this work, we propose the PianoTree VAE, a novel tree-structure extension upon VAE aiming to fit the polyphonic music learning. The experiments prove the validity of the PianoTree VAE via (i)-semantically meaningful latent code for polyphonic segments; (ii)-more satisfiable reconstruction aside of decent geometry learned in the latent space; (iii)-this model’s benefits to the variety of the downstream music generation.¹

1. INTRODUCTION

Unsupervised machine learning has led to a marriage of symbolic learning and vectorized representation learning [1–3]. In the computer music community, the MusicVAE [4] enables the interpolation in the learned latent space to render some smooth music transition. The EC²-VAE [5] manages to disentangle certain interpretable factors in music and also provides a manipulable generation pathway based on these factors. Pati *et al.* [6] further utilizes the recurrent networks to learned music representations for longer-term coherence.

Unfortunately, most of the success has been limited to monophonic music. The generalization of the learning frameworks to polyphonic music is not trivial, due to its much higher dimensionality and more complicated musical syntax. The commonly-adopted MIDI-like event sequence modeling or the piano-roll formats fed to either recurrent or convolutional networks have fell short in learn-

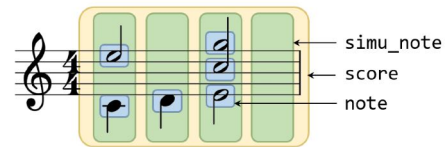


Figure 1: An illustration of the proposed polyphonic syntax.

ing good representation, which usually leads to unsatisfied generation results [7–9]. In this paper, we hope to pioneer the development of this challenging task. To begin with, we conjecture a proper set of **inductive bias** for the desired framework: (i)-a sparse encoding of music as the model input; (ii)-a neural architecture that incorporates the hierarchical structure of polyphonic music (i.e., musical syntax).

Guided by the aforementioned design principles, we propose PianoTree VAE, a hierarchical representation learning model under the VAE framework. We adopt a tree structured musical syntax that reflects the hierarchy of musical concepts, which is shown in Figure 1. In a top-down order: we define a *score* (indicated by the yellow rectangle) as a series of *simu_note* events (indicated by the green rectangles), a *simu_note* as multiple *note* events sharing the same onset (indicated by blue rectangles), and each *note* has several attributes such as *pitch* and *duration*. In this paper, we focus on a simple yet common form of polyphonic music—piano score, in which each note has only pitch and duration attributes. For future work, this syntax can be generalized to multiple instruments and expressive performance by adding extra attributes such as voice, expressive timing, dynamics, etc.

The whole neural architecture of PianoTree VAE can be seen as a tree. Each node represents the embedding of either a *score*, *simu_note*, or *note*, where a higher level representation has larger receptive fields. The edges are bidirectional where a recurrent module is applied to either encode the children into the parent or decode the parent to generate its children.

Through extensive evaluations, we show that PianoTree VAE yields semantically more meaningful latent representations and further downstream generation quality gains, on top of the current state-of-the-art solutions.

¹ Code and demos can be accessed via <https://github.com/ZZWaang/PianoTree-VAE>



2. RELATED WORK

The complex hierarchical nature of music data has been studied for nearly a century (e.g. GTTM [10], Schenkerian Analysis [11], and their follow-up works [12–15]). However, the emerging deep representation-learning models still lack the compatible solutions to deal with the complex musical structure. In this section, we first review different types of polyphonic music generation in Section 2.1. After that, we discuss some popular deep music generative models indexed by their compatible data structure from Section 2.2 to Section 2.4.

2.1 Different Types of Polyphony

In the context of deep music generation, *polyphony* can refer to three types of music: 1) multiple monophonic parts (e.g., a four-part chorus), 2) a single part of a polyphonic instrument (e.g., a piano sonata), and 3) multiple parts of polyphonic instruments (e.g., a symphony).

The first type of polyphonic music can be created by simply extending the number of voices in monophonic music generation with some inter-voice constraints. Some representative systems belonging to this category include DeepBach [16], XiaoIce [17], and Coconet [18]. Music Transformer [19] and the proposed PianoTree VAE both focus on the generation of the second type of polyphony, which is a much more difficult task. Polyphonic pieces under the second definition no longer have a fixed number of “voices” and consist of more complex textures. The third type of polyphony can be regarded as an extension of the second type, and we leave it for future work.

2.2 Piano-roll and Compatible Models

Piano-roll and its variations [7, 20–22] view polyphonic music as 3-D (one-hot) tensors, in which the first two dimensions denote time and pitch and the third dimension indicates whether the token is an onset, sustain or rest. A common way for deep learning models to encode/decode a piano-roll is to use recurrent layers along the time-axis while the pitch-axis relations are modeled in various ways [20, 21, 23]. Another method is to regard a piano-roll as an image with three channels (onset, sustain and rest) and apply convolutional layers [7, 22].

Through the proposal of PianoTree VAE, we argue that a major way to improve the current deep learning models is to utilize the built-in priors (intrinsic structure) in the musical data. In our work, we primarily use the sparsity and the hierarchical priors.

2.3 MIDI-like Event Sequence and Compatible Models

MIDI-like event sequence is first used in deep music generation in performanceRNN [24] and Multi-track MusicVAE [9], and then broadly applied in transformer-based generation [19, 25, 26]. This direction of work leverages the sparsity of polyphonic data to efficiently flatten polyphonic music into an array of events. The vocabulary size of events usually triples the vocabulary size of MIDI pitches, including “note-on” and “note-off” events for 128 MIDI pitches, “time shifts”, and so on.

However, the format of MIDI-like events lacks the proper flexibility. A few operations are made difficult due to its very nature. For instance, during addition or deletion of notes, often numerous “time shift” tokens must be merged or split with the “note-on” or “note-off” tokens being changed all-together. This has caused the model being trained inefficient for the potential generation tasks. In addition, this format has a risk of generating illegal sequences, say a “note on” message without a paired “note off” message.

Similarly, we see the note-based approaches [27, 28], in which polyphonic music is represented as a sequence of note tuples, as an alternative to the MIDI-like methods. The representation has resolved the illegal generation problem but still not revealed much of the intrinsic music structure. We argue that our work improves on the note-based approaches by utilizing deeper musical structures implied by the data. (See Section 3.1 for details.)

2.4 GNN as a Novel Structure

Recently, we see a trend in using graph neural networks (GNN) [29] to represent polyphonic score [30], in which each vertex represents a note and the edges represent different musical relations. Although the GNN-based model offers sparse representation learning capacity, it is limited by the specification of the graph structure design and it is nontrivial to generalize it for score generations.

3. METHOD

3.1 Data Structure

We first define a data structure to represent a polyphonic music segment, which contains two components: 1) *surface structure*, a data format to represent the music observation, and 2) *deep structure*, a tree structure (containing `score`, `simu_note` and `note` nodes) showing the syntactic construct of the music segment.

Each music segment lasts T time steps with $\frac{1}{4}$ beat as the shortest unit. We further use K_t , where $1 \leq t \leq T$ to denote the number of notes having the same onset t . The current model uses $T = 32$, i.e., each music segment is 8-beat long.

3.1.1 Surface Structure

The surface structure is a nested array of *pitch-duration* tuples, denoted by $\{(p_{t,k}, d_{t,k}) | 1 \leq t \leq T, 1 \leq k \leq K_t\}$. Here, $(p_{t,k}, d_{t,k})$ is the k^{th} lowest note starting at time step t . The pitch attribute $p_{t,k}$ is a 128-D one-hot vector corresponding to 128 MIDI pitches. The duration attribute $d_{t,k}$ encodes the duration ranging from 1 to T using a $\log_2 T$ -bit binary vector. For example, when $T = 32$ ($\log_2 T = 5$), ‘00000’ represents a 16th note, ‘00001’ is an 8th note, ‘00010’ is a dotted 8th note, and so on so forth. The base-2 design is inspired by the similar binary relation among different note values in western musical notation.

The bottom part of Figure 2 illustrates the surface structure of the music example in Figure 1. We see that the data structure is a sparse encoding of music, and it eliminates illegal tokens since every possible nested array has a correspondent music.

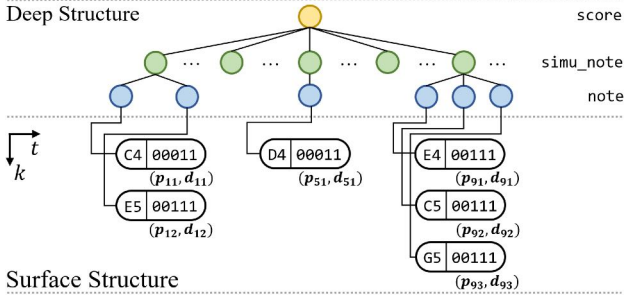


Figure 2: An illustration of PianoTree data structure to encode the music example in Figure 1.

3.1.2 Deep Structure

We further build a syntax tree to reveal the hierarchical relation of the observation. First, for $1 \leq t \leq T, 1 \leq k \leq K_t$, we define $\text{note}_{t,k}$ as the summary (i.e., embedding) of $(p_{t,k}, d_{t,k})$, which are the bottom layers of the tree. Then, for $1 \leq t \leq T$, we define simu_note_t as the summary of $\text{note}_{t,1 \leq k \leq K_t}$, which are the middle layers of the tree. Finally, we define the score as the summary of $\text{simu_note}_{1 \leq t \leq T}$, which is the root of the tree. The upper part of Figure 2 illustrates the deep structure built upon its surface structure.

The syntax tree, so-called the deep structure has both musical and linguistic consideration. In terms of music, note , simu_note and score roughly reflect the musical concept of a note, chord and grouping. In terms of linguistics, the tree is analogous to a constituency tree, with surface structure being the terminal nodes and deep structure being the non-terminals. Recent studies in natural language processing have revealed that incorporating natural language syntax results in better semantics modeling [31, 32].

3.2 Model Structure

We use the surface structure of polyphonic music as the model input. The VAE architecture is built upon the deep structure.

We denote the music segment in the proposed surface structure as x and the latent code as z , which conforms to a standard Gaussian prior denoted by $p(z)$. The encoder models the approximated posterior $q_\phi(z|x)$ in a bottom-up order of the deep structure. First, note embeddings are computed through a linear transform of pitch-duration tuples. Second, the note embeddings (sorted by pitch) are then embedded into simu_note using a bi-directional GRU [33] by concatenating the last hidden states on both ends. With the same method, the simu_note embeddings (sorted by onsets) are summarized into score by another bi-directional GRU. We assume an isotropic Gaussian posterior, whose mean and log standard deviation are computed by a linear mapping of score . Algorithm 1 shows the details.

The decoder models $p_\theta(x|z)$ in a top-down order of the deep structure, almost mirroring the encoding process. We use a uni-directional time-axis GRU to decode simu_note , another uni-directional (pitch-axis) GRU to decode note , a fully connected layer to decode pitch at-

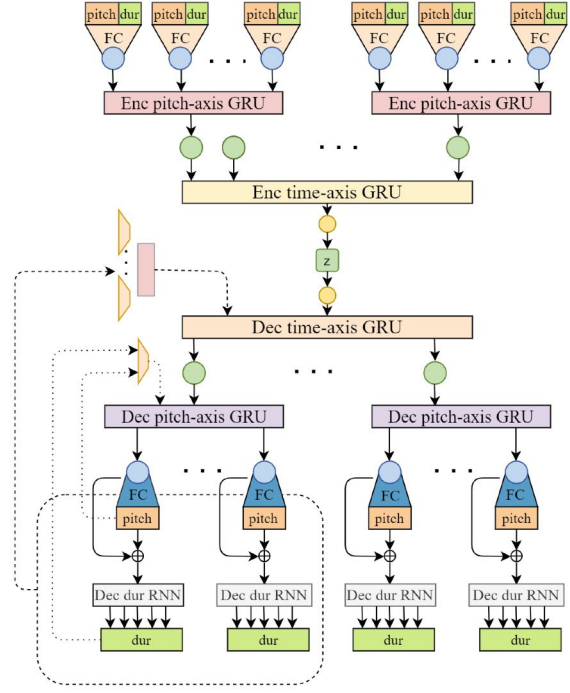


Figure 3: An overview of the model architecture. The recurrent layers are represented by rectangles and the fully-connected (FC) layers are represented by trapezoids. The note , simu_note and score events are represented by circles.

tributes, and finally another GRU to decode duration attribute starting from the most significant bit. Algorithm 2 shows the details.

We use the ELBO (evidence lower bound) [34] as our training objective. Formally,

$$\mathcal{L}(\phi, \theta; x) = -\mathbb{E}_{z \sim q_\phi} \log p_\theta(x|z) + \beta \text{KL}(q_\phi || p(z)), \quad (1)$$

where β is a balancing parameter used in β -VAE [35].

We denote the embedding size of note , simu_note and score as e_n , e_{sn} and e_{sc} ; the dimension of latent space as d_z ; and the hidden dimensions or pitch-axis, time-axis and dur GRUs as h_p , h_t and h_d respectively. In this work, we report our result on the following model size: $e_n = 128$, $e_{sn} = h_{p,dec} = 2 \times h_{p,enc} = 512$, $e_{sc} = h_{t,dec} = 2 \times h_{t,enc} = 1024$, $h_{d,dec} = 64$ and $d_z = 512$.

Algorithm 1: The PianoTree Encoder. n, sn, sc are short for note , simu_note , score .

```
/* gru(.) : passes a sequence to
bi-directional GRU and outputs the
concatenation of hidden states from
both ends. */
```

input: PianoTree

```

 $x = \{(p_{t,k}, d_{t,k}), 1 \leq t \leq T, 1 \leq k \leq K_t\}$ 
foreach  $t, k$  do  $n_{t,k} \leftarrow \text{emb}_{\text{enc}}(p_{t,k}, d_{t,k});$ 
foreach  $t$  do  $sn_t \leftarrow \text{gru}_{\text{enc}}^{\text{pitch}}(n_{t,1:K_t});$ 
 $sc \leftarrow \text{gru}_{\text{enc}}^{\text{time}}(sn_{1:T});$ 
 $\mu \leftarrow \text{fc}_\mu(sc); \sigma \leftarrow \exp(\text{fc}_\sigma(sc));$ 
return  $q(z|x) = N(\mu, \sigma^2);$ 
```

Algorithm 2: The PianoTree Decoder. We still use the abbreviation n , sn , and sc , defined in Algorithm 1

```

/* gru( $\cdot$ ), same as Algorithm 1. */
/* gruCell( $\cdot, \cdot$ ): updates the hidden state
   using the current input and the
   previous hidden state. The output is
   replicated. */
input: latent representation  $z$ 
 $sc \leftarrow z$ ;
 $\tilde{sn}_0, \tilde{n}_{:,0}, d_{:,:,0} = \langle \text{SOS} \rangle$ ;
for  $t = 1, 2, \dots, T$  do
    [ $sn_t, sc$ ]  $\leftarrow$  gruCelltimedec( $\tilde{sn}_{t-1}, sc$ );
    for  $k = 1, 2, \dots$  do
        [ $n_{t,k}, sn_t$ ]  $\leftarrow$  gruCellpitchdec( $\tilde{n}_{t,k-1}, sn_t$ );
         $p_{t,k} \leftarrow \text{softmax}(\text{fc}(n_{t,k}))$ ;
        for  $r = 1, 2, \dots, 5$  do
             $h = [n_{t,k}, p_{t,k}]$ ;
            [ $y_{t,k,r}, h$ ] = gruCelldurdec( $d_{t,k,r-1}, h$ );
             $d_{t,k,r} \leftarrow \text{softmax}(y_{t,k,r})$ ;
        end
         $d_{t,k} = [d_{t,k,1:5}]$ ;
        if  $p_{t,k} \neq \langle \text{EOS} \rangle$  then  $K_t \leftarrow k$ ; break;
         $\tilde{n}_{t,k} \leftarrow \text{emb}_{\text{enc}}(p_{t,k}, d_{t,k})$ ;
    end
     $\tilde{sn}_t \leftarrow \text{gru}_{\text{enc}}^{\text{pitch}}(n_{t,1:K_t})$ ;
end
return  $\{(p_{t,k}, d_{t,k}), 1 \leq t \leq T, 1 \leq k \leq K_t\}$ ;

```

4. EXPERIMENTS

In this section, we compare PianoTree VAE with several baseline models. We present the dataset in Section 4.1, baseline models in Section 4.2, and the training details in Section 4.3. We present the objective evaluation on reconstruction accuracy in Section 4.4. In Section 4.5, we inspect and visualize the latent space of `note` and `simu_note`. After that, we present the subjective evaluation on latent space traversal in Section 4.6. Finally, we apply the learned representation to downstream music generation task in Section 4.7.

4.1 Dataset

We collect around 5K classical and popular piano pieces from Musicalion² and the POP909 dataset [36]. We only keep the pieces with $\frac{2}{4}$ and $\frac{4}{4}$ meters and cut them into 8-beat music segments (i.e., each data sample in our experiment contains 32 time steps under sixteenth note resolution). In all, we have 19.8K samples. We randomly split the dataset (at song-level) into training set (90%) and test set (10%). All training samples are further augmented by transposing to all 12 keys.

4.2 Baseline Model Architectures

We train four types of baseline models in total using piano-roll (Section 2.2) and MIDI-like events (Section 2.3) data structures. As a piano-roll can be regarded as either a sequence or a 2-dimensional image, we couple it with

three neural encoder-decoder architectures: a recurrent VAE (**pr-rnn**), a convolutional VAE (**pr-cnn**), and a fully-connected VAE (**pr-fc**). For the MIDI-like events, we couple it with a recurrent VAE model (**midi-seq**). All models share the same latent space dimension ($d_z = 512$). Specifically,

- The piano-roll recurrent VAE (**pr-rnn**) model is similar to a 2-bar MusicVAE proposed in [4]. The hidden dimensions of the GRU encoder and decoder are both 1024.
- The piano-roll convolutional VAE (**pr-cnn**) architecture adopts a convolutional–deconvolutional architecture. The encoder contains 8 convolutional layers with kernel size 3×3 . Strided convolution is performed at the 3rd, 5th, 7th and 8th layer with stride size (2×1) , (2×3) , (2×2) and (2×2) respectively. The decoder adopts the deconvolution operations in a reversed order.
- The piano-roll fully-connected VAE (**pr-fc**) architecture uses a time-distributed 256-dimensional embedding layer, followed by 3 fully-connected layers with the hidden dimensions [1024, 768] for the encoder. The decoder adopts the counter-operations in a reversed order.
- The MIDI-like event recurrent VAE (**midi-seq**) adopts the recurrent model structure similar to **pr-rnn**. Here, the event vocabulary contains 128 “note-on”, 128 “note-off” and 32 “time shift” tokens. The embedding size of a single MIDI event is 128. The hidden dimensions of the encoder GRU and decoder GRU are 512 and 1024 respectively.

4.3 Training

For all models, we set batch size = 128 and use Adam optimizer [37] with a learning rate starting from $1e-3$ with exponential decay to $1e-5$. For PianoTree VAE, we use teacher forcing [38] for decoder time-axis and pitch-axis GRU and for other recurrent-based baselines, we use teacher forcing in the decoders. The teacher forcing rates start from 0.8 and decay to 0.0. PianoTree VAE converges within 6 epochs, and the baseline models converge in approximately 40 to 60 epochs.

Models	PianoTree	midi-seq	pr-rnn	pr-cnn	pr-fc
Onset Precision	0.9558	0.8929	0.9533	0.9386	0.9211
Onset Recall	0.9532	0.6883	0.9270	0.8818	0.8827
Onset F1	0.9545	0.7774	0.9399	0.9093	0.9015
Duration Precision	0.9908	0.3826	0.9777	0.9757	0.9688
Duration Recall	0.9830	0.9899	0.9891	0.9796	0.9743
Duration F1	0.9869	0.5519	0.9834	0.9777	0.9715

Table 1: Objective evaluation results on reconstruction criteria. PianoTree is our proposed method. Other columns correspond to the baseline models described in Section 4.2.

4.4 Objective Evaluation of Reconstruction

The objective evaluation is performed by comparing different models in terms of their reconstruction accuracy of pitch onsets and note duration [39, 40], which are commonly used measurements in music information retrieval tasks. For note duration accuracy, we only consider the

²Musicalion: <https://www.musicalion.com>.

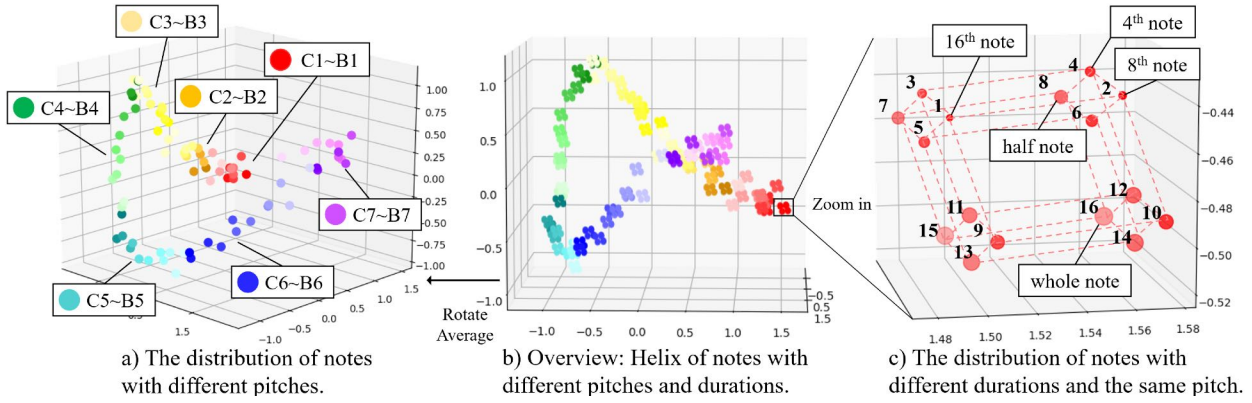


Figure 4: A visualization of `note` embeddings after dimensionality reduction using PCA.

notes whose onset and pitch reconstruction is correct. Table 1 summarizes the results where we see that the PianoTree VAE (the 1st column) is better than others in terms of F1 score for both criteria.

4.5 Latent Space Visualization

Figure 4 shows the *latent note space* by plotting different `note` embeddings after dimensionality reduction by PCA (with the three largest principal components being reserved). Each colored dot is a `note` embedding and a total of 1344 samples are displayed; note pitch ranges from C-1 to C-8 and note duration from a sixteenth note to a whole note.

We see that the `note` embeddings have the desired geometric properties. Figure 4 (a) & (b) show that at a macro level, notes with different pitches are well sorted and form a “helix” in the 3-D space. Figure 4 (c) further shows that at a micro level, 16 different note durations (with the same pitch) form a “fractal parallelogram” due to the binary encoding of duration attributes. One of the advantages of the encoding method is the translation invariance property. For example, the duration difference between the upper left cluster and the lower left cluster is 8 semiquavers, so is the difference between the upper right and lower right cluster. The same property also applies to the four smaller-scale parallelograms.

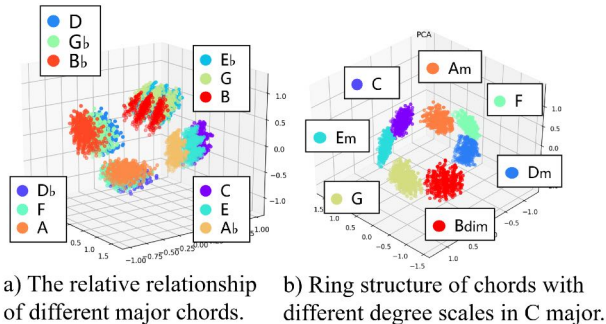


Figure 5: A visualization of `simu_note` embeddings after dimensionality reduction using PCA.

Figure 5 is a visualization of the latent chord space by plotting different `simu_note` embeddings under PCA di-

dimensionality reduction. Each colored cluster corresponds to a chord label realized in 343 different ways (we consider all possible pitch combinations within three octaves, with a minimum of 3 notes and a maximum of 9 notes). The duration for all chords is one beat.

The geometric relationships among different chords are consistent and human interpretable. In specific, Figure 5 (a) shows the distribution of 12 different major chords, which are clustered in four different groups. By unfolding the circle in a counterclockwise direction, we can observe the existence of *the circle of the fifth*. Figure 5 (b) is the visualization of seven C major triad chords: forming a ring in the order of 1-3-5-7-2-4-6 degree in the counterclockwise direction.

4.6 Subjective Evaluation of Latent Space Interpolation

Latent space traversal [4, 5, 41] is a popular technique to demonstrate model generalization and the smoothness of the learned latent manifold. When interpolating from one music piece to another in the latent space, new pieces can be generated by mapping the representations back to the signals. If a VAE is well trained, the generated piece will sound natural and form a smooth transition.

To this end, we invite people to subjectively rate the models through a double-blind online survey. During the survey, the subjects first listen to a pair of music, and then listen to 5 versions of interpolation, each generated by a model listed in Table 1. Each version is a randomly selected pair of music segments, and the interpolation is achieved using SLERP [42]. Since the experiment requires careful listening and a long survey could decrease the quality of answers, each subject is asked to rate only 3 pairs of music, i.e., $3 \times 5 = 15$ interpolations in a random order. After listening to the 5 interpolations of each pair, subjects are asked to select two best versions: one in terms of the *overall musicality*, and the other in terms of the *smoothness of transition*.

A total of $n = 33$ subjects (12 females and 21 males) with different music backgrounds have completed the survey. The aggregated result (as in Figure 6) shows that the interpolations generated by our model are better than the ones generated by baselines, in terms of both overall musi-

cality and smoothness of transition. Here, different colors represent different models (with the blue bars being our model and other colors being the baselines), and the height of the bars represent the percentage of votes (on the best candidate).

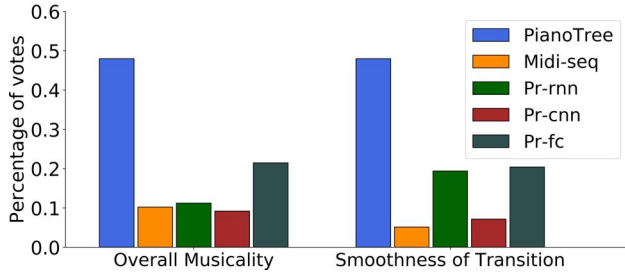


Figure 6: Subjective evaluation results of latent space interpolation.

4.7 Downstream Music Generation

In this section, we further explore whether the polyphonic representation helps with *long-term music generation* when coupled with standard downstream sequence prediction models. (Similar tasks have been applied to *monophonic music* in [43] and [6].)

The generation task is designed in the following way: given 4 measures of piano composition, we predict the next 4 measures using a Transformer decoder (as in [44]). We compare three different music representations: MIDI-like event sequence (Section 2.1), pretrained (decoder) `simu_note` embeddings, and latent vector z for every 2-measure music segment (without overlap). Here z is the mean of the approximated posterior from the encoder. For all three representations, we use the same Transformer decoder architecture (outputs of dimension = 128, number of layers = 6 and number of heads = 8) with the same training procedure. Only the loss functions are correspondingly adjusted based on different representations: cross entropy loss is applied to midi-event tokens and MSE loss is applied to both `simu_note` and latent vector z . We use the same datasets mentioned in Section 4.1 and cut the original piano pieces into 8-measure subsequent clips for generation purposes. We still keep 90% for training and 10% for testing.

We then invited people to subjectively rate different music generations through a double-blind online survey (similar to the one in Section 4.6). Subjects are asked to listen to and rate 6 music clips, each of which contains 3 versions of 8-measure generation using different note representations. Subjects are told that the first 4 measures are given and the rest are generated by the machine. For each music clip, subjects rate it based on *creativity*, *naturalness* and *musicality*.

A total of $n = 48$ subjects (20 females and 28 males) with different music backgrounds have participated in the survey. Figure 7 summarizes the survey results, where the heights of bars represent means of the ratings and the error bars represent the confidence intervals computed via within-subject ANOVA [45]. The result shows that `simu_note` and latent vector z perform significantly better

than the midi-event tokens in terms of all three criteria ($p < 0.005$).

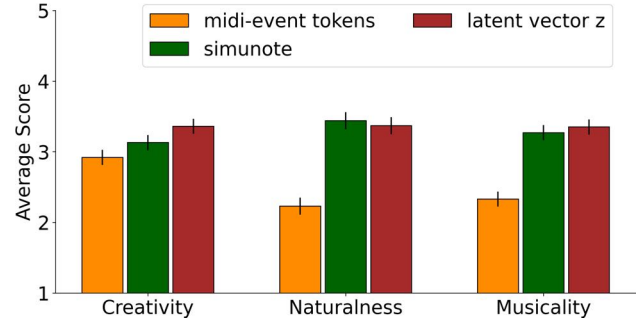


Figure 7: Subjective evaluation results of downstream music generation.

Besides the aforementioned generation task, we also iteratively feed the generated 4-measure music clips into the model to get longer music compositions. Figure 8 shows a comparison of 16-measure generation results using all three representations. The first 4 bars are selected from the test set, and the subsequent 12 bars are generated by the models. Generally speaking, using `simu_note` and latent vector z as data representations yields more coherent music compositions. Furthermore, we noticed that long generations using the `simu_note` representation tend to repeat previous steps in terms of both chords and rhythms, while those generations using the latent vector z usually contain more variations.

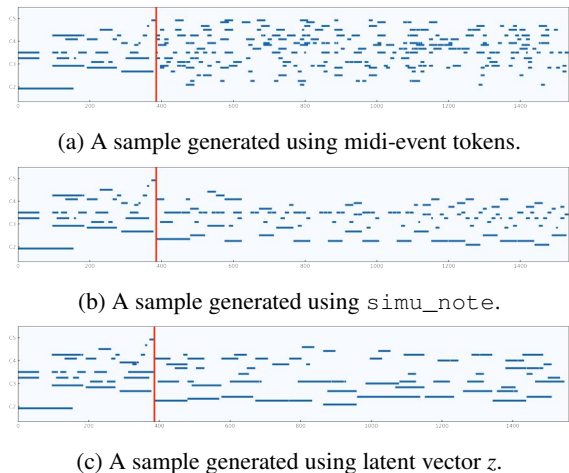


Figure 8: Long music generations given first 4 measures.

5. CONCLUSION AND FUTURE WORK

In conclusion, we proposed PianoTree VAE, a novel representation-learning model tailored for polyphonic music. The key design of the model is to incorporate both the music data structure and model architecture with the sparsity and hierarchical priors. Experiments show that with such inductive biases, PianoTree VAE achieves better reconstruction, interpolation, downstream generation, and strong model interpretability. In the future, we plan to extend PianoTree VAE for more general musical structures, such as motif development and multi-part polyphony.

6. REFERENCES

- [1] T. Zhao, R. Zhao, and M. Eskenazi, "Learning discourse-level diversity for neural dialog models using conditional variational autoencoders," *arXiv preprint arXiv:1703.10960*, 2017.
- [2] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," *arXiv preprint arXiv:1511.06349*, 2015.
- [3] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," in *Advances in neural information processing systems*, 2015, pp. 2980–2988.
- [4] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," *arXiv preprint arXiv:1803.05428*, 2018.
- [5] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, "Deep music analogy via latent representation disentanglement," *arXiv preprint arXiv:1906.03626*, 2019.
- [6] A. Pati, A. Lerch, and G. Hadjeres, "Learning to traverse latent spaces for musical score inpainting," *arXiv preprint arXiv:1907.01164*, 2019.
- [7] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [8] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," *arXiv preprint arXiv:1703.10847*, 2017.
- [9] I. Simon, A. Roberts, C. Raffel, J. Engel, C. Hawthorne, and D. Eck, "Learning a Latent Space of Multitrack Measures," *arXiv e-prints*, p. arXiv:1806.00195, Jun 2018.
- [10] F. Lerdahl and R. S. Jackendoff, *A generative theory of tonal music*. MIT press, 1996.
- [11] J. Rothgeb, *Introduction to the theory of Heinrich Schenker: the nature of the musical work of art*. New York: Longman, 1982.
- [12] M. Hamanaka, K. Hirata, and S. Tojo, "Implementing "a generative theory of tonal music"," *Journal of New Music Research*, vol. 35, no. 4, pp. 249–277, 2006.
- [13] —, " σ gttm iii: Learning-based time-span tree generator based on pcf_g," in *International Symposium on Computer Music Multidisciplinary Research*. Springer, 2015, pp. 387–404.
- [14] S. W. Smoliar, "A computer aid for schenkerian analysis," in *Proceedings of the 1979 annual conference*, 1979, pp. 110–115.
- [15] A. Marsden, "Schenkerian analysis by computer: A proof of concept," *Journal of New Music Research*, vol. 39, no. 3, pp. 269–289, 2010.
- [16] G. Hadjeres, F. Pachet, and F. Nielsen, "Deepbach: a steerable model for bach chorales generation," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1362–1371.
- [17] H. Zhu, Q. Liu, N. J. Yuan, C. Qin, J. Li, K. Zhang, G. Zhou, F. Wei, Y. Xu, and E. Chen, "Xiaoice band: A melody and arrangement generation framework for pop music," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 2837–2846.
- [18] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, "Counterpoint by convolution," in *International Society for Music Information Retrieval (ISMIR)*, 2017.
- [19] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, C. Hawthorne, A. M. Dai, M. D. Hoffman, and D. Eck, "Music transformer: Generating music with long-term structure," *arXiv preprint arXiv:1809.04281*, 2018.
- [20] G. Brunner, Y. Wang, R. Wattenhofer, and J. Wiesendanger, "Jambot: Music theory aware chord based generation of polyphonic music with lstms," in *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2017, pp. 519–526.
- [21] H. H. Mao, T. Shin, and G. Cottrell, "Deepj: Style-specific music generation," in *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*. IEEE, 2018, pp. 377–382.
- [22] E. S. Koh, S. Dubnov, and D. Wright, "Rethinking recurrent latent variable model for music composition," in *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2018, pp. 1–6.
- [23] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," *arXiv preprint arXiv:1206.6392*, 2012.
- [24] I. Simon and S. Oore, "Performance rnn: Generating music with expressive timing and dynamics," <https://magenta.tensorflow.org/performance-rnn>, 2017.
- [25] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, "Lakhnes: Improving multi-instrumental music generation with cross-domain pre-training," *arXiv preprint arXiv:1907.04868*, 2019.

- [26] Y.-S. Huang and Y.-H. Yang, “Pop music transformer: Generating music with rhythm and harmony,” *arXiv preprint arXiv:2002.00212*, 2020.
- [27] O. Mogren, “C-rnn-gan: Continuous recurrent neural networks with adversarial training,” *arXiv preprint arXiv:1611.09904*, 2016.
- [28] C. Hawthorne, A. Huang, D. Ippolito, and D. Eck, “Transformer-nade for piano performances.”
- [29] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.
- [30] D. Jeong, T. Kwon, Y. Kim, and J. Nam, “Graph neural network for music score data and modeling expressive piano performance,” in *International Conference on Machine Learning*, 2019, pp. 3060–3070.
- [31] C. Dyer, A. Kuncoro, M. Ballesteros, and N. A. Smith, “Recurrent neural network grammars,” *arXiv preprint arXiv:1602.07776*, 2016.
- [32] K. S. Tai, R. Socher, and C. D. Manning, “Improved semantic representations from tree-structured long short-term memory networks,” *arXiv preprint arXiv:1503.00075*, 2015.
- [33] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.
- [34] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.
- [35] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, “beta-vae: Learning basic visual concepts with a constrained variational framework,” 2016.
- [36] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, X. Gu, and G. Xia, “Pop909: A pop-song dataset for music arrangement generation,” in *Proceedings of 21st International Conference on Music Information Retrieval (ISMIR), virtual conference*, 2020.
- [37] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [38] N. B. Toomarian and J. Barhen, “Learning a trajectory using adjoint functions and teacher forcing,” *Neural networks*, vol. 5, no. 3, pp. 473–484, 1992.
- [39] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, “Onsets and frames: Dual-objective piano transcription,” *arXiv preprint arXiv:1710.11153*, 2017.
- [40] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, “mir_eval: A transparent implementation of common mir metrics,” in *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*. Citeseer, 2014.
- [41] R. Yang, T. Chen, Y. Zhang, and G. Xia, “Inspecting and interacting with meaningful music representations using vae,” *arXiv preprint arXiv:1904.08842*, 2019.
- [42] A. Watt and M. Watt, “Advanced animatidn and ben-dering technidues,” 1992.
- [43] K. Chen, G. Xia, and S. Dubnov, “Continuous melody generation via disentangled short-term representations and structural conditions,” *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, pp. 128–135, 2020.
- [44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *ArXiv*, vol. abs/1706.03762, 2017.
- [45] H. Scheffe, *The analysis of variance*. John Wiley & Sons, 1999, vol. 72.