

# SINTRa: LEARNING AN INSPIRATION MODEL FROM A SINGLE MULTI-TRACK MUSIC SEGMENT

Qingwei Song<sup>1</sup>    Qiwei Sun<sup>2</sup>    Dongsheng Guo<sup>1</sup>    Haiyong Zheng<sup>1\*</sup>

<sup>1</sup> College of Electronic Engineering, Ocean University of China, China

<sup>2</sup> School of Microelectronics, Xi'an Jiaotong University, China

\* Corresponding author: zhenghaiyong@ouc.edu.cn

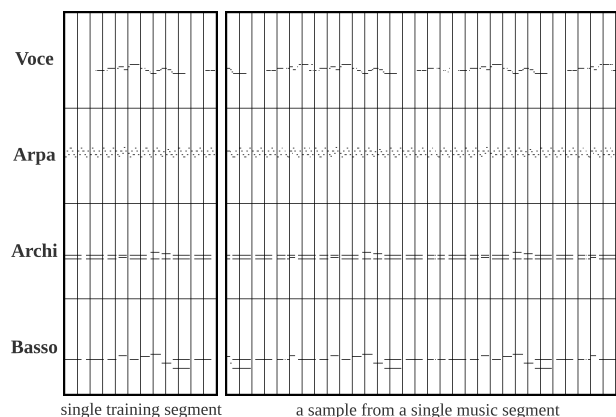
## ABSTRACT

In this paper, we propose SinTra, an auto-regressive sequential generative model that can learn from a single multi-track music segment, to generate coherent, aesthetic, and variable polyphonic music of multi-instruments with an arbitrary length of bar. For this task, to ensure the relevance of generated samples and training music, we present a novel pitch-group representation. SinTra, consisting of a pyramid of Transformer-XL with a multi-scale training strategy, can learn both the musical structure and the relative positional relationship between notes of the single training music segment. Additionally, for maintaining the inter-track correlation, we use the convolution operation to process multi-track music, and when decoding, the tracks are independent to each other to prevent interference. We evaluate SinTra with both subjective study and objective metrics. The comparison results show that our framework can learn information from a single music segment more sufficiently than Music Transformer. Also the comparison between SinTra and its variant, i.e., the single-stage SinTra with the first stage only, shows that the pyramid structure can effectively suppress overly-fragmented notes.

## 1. INTRODUCTION

The current development trend of music generation is to generate harmonious multi-track music with longer-term dependency. However, in the composition of real life, the inspiration is the beginning of a song, and the composer usually creates a music based on a single music segment that comes to mind. Thus, it's more important to find ideas that are relevant to the inspiration.

As for composers, the composition process can be divided into two stages. The first stage is to generate a large number of ideas that provide inspiration for subsequent creation. The second stage is about idea convergence. Composers need to find ideas that can express their feelings and emotions to the audience from a large number of ideas, then expand, repeat and arrange them, and



**Figure 1.** One sample of four-track piano-roll (right) with 32-bar length (each block represents a bar), generated by our SinTra model trained from a single music segment (left). The y-axis and x-axis represent note pitch (range from 0 to 127) and time step  $T$ , respectively.

finally end up with a song. At the first stage, generated ideas constitute small segments of music, from which to compose more similar segments can provide more inspiration for composers. Therefore, as to music generation, it's significant to learn an inspiration model, that is, a generative model to generate music, inspiring the composers for creating a song according to a single music segment.

Recently, music generation has witnessed great progress due to the development of deep learning technologies. The mainstream methods are modeling the note sequences by drawing lessons from language models in natural language processing (NLP), which require a large number of MIDIs as training set to learn the distribution of notes. Generally, the training period is long, and the randomness of generated music is relatively high. However, if only a single music segment is available for training, it's hard for previous models to acquire enough information for leaning reasonable musical structure and relation position relationship between notes, resulting in chaotic and aesthetically unpleasant music (refer to Section 6).

At present, some one-shot generation works [1, 2] followed the multi-scale training mechanism and achieved compelling results, which allow the network to learn the information from single training data in different scales more sufficiently. Besides, recent works in music gener-

ation [3–5] adopted the Transformer-XL [6], an improved variant of the Transformer [7], to introduce recurrence to the architecture, as the backbone sequence model.

In this paper, we leverage the multi-scale training scheme and the Transformer-XL architecture, to tackle the more challenging and meaningful one-shot music generation, that is, generating music from a single multi-track music segment. To ensure the relevance of generated music and training segment, we present a novel pitch-group representation to contain all pitch group types of the single training music segment. Besides, in order to deal with more complex multi-track music, we design three modules in each stage (scale) of our multi-scale training.

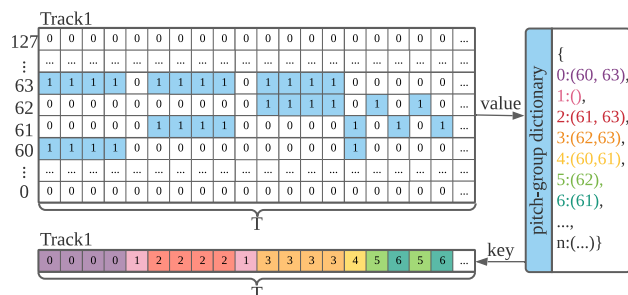
To summarize, SinTra can actually be regarded as an inspiration model for music composition. When the composer is lacking in inspiration, or in creation, it would be very repetitive to make detailed adjustments at the structure level, and SinTra can help. Certainly, the subsequent fine-tuning still needs to be done by humans. We make four contributions: (1) A novel pitch-group representation is presented to model polyphonic music of single instrument into a sequence; (2) A novel inspiration model, namely SinTra, is devised to generate meaningful music from only a single music segment; (3) Three modules based on Transformer-XL are designed for each stage of multi-scale training to process multi-track music. (4) The source code<sup>1</sup> and music data are made publicly available.

## 2. RELATED WORK

### 2.1 Music Generation

Music generation, as a niche research task of music information retrieval (MIR), has a long history and has attracted great attention in both industrial and art communities recently.

As a traditional method, Markov models are often used in the field of MIR, such as the work from Simon et al. [8] and Tsushima et al. [9]. Chuan et al. [10] have also used support vector machine (SVM) to select chord tones from given melodies. Then, recurrent neural network (RNN) [11] with long short-term memory (LSTM) [12] and gated recurrent unit (GRU) [13], variational auto-encoder (VAE) [14], and generative adversarial network (GAN) [15], are common deep learning frameworks used for modeling music sequence. MuseGAN [16] generated music as an image (converting MIDI into piano-roll) with GANs, and used an inter-track latent vector to make the generated multi-track music coherent. To overcome the binarization issue, the upgraded version of MuseGAN, Binary MuseGAN [17] proposed an additional refiner network, which enables generator to directly generate binary-valued piano-rolls at test time. Moreover, two types of binary neurons (BNs) considered features fewer overly-fragmented notes as compared to MuseGAN. MIDI-Sandwich2 [18], which also used piano-roll, applied a hierarchical multi-modal fusion generative VAE network



**Figure 2.** Illustration of our pitch-group representation. The pitch-group dictionary is built and regarded as the database of music segment containing  $n + 1$  key-value pairs, where the key (e.g., 0) means the index of pitch group type and the value (e.g., (60, 63)) means the pitch group information. Piano-roll ( $T \times 128$ ) can be mapped to token sequence ( $T \times 1$ ) through pitch-group dictionary ( $T$  means time step).

based on RNN to collaboratively generate multi-track symbolic music. XiaoIce Band [19], a melody and arrangement generation framework for pop music, introduced cooperate GRUs between each generation track to generate melody and multi-track music arrangement. DeepJ [20], based on Bi-LSTM [21], was trained using piano-roll for style-specific music generating (baroque, classical, and romantic). Different from previous music generation models, our work devotes to learning a model to generate music from only a single segment.

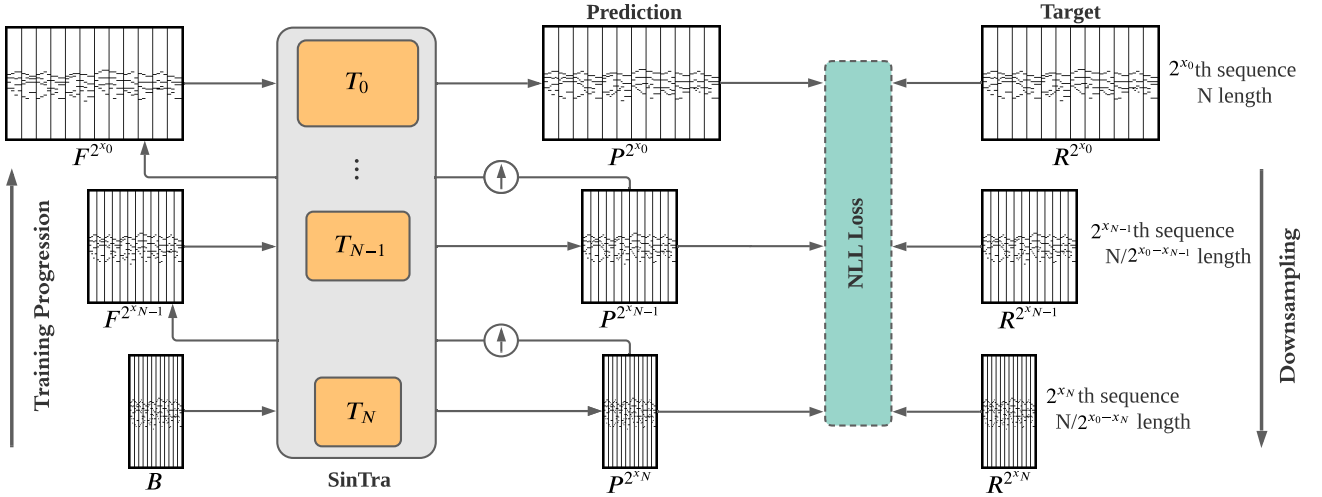
### 2.2 Transformer and Multi-scale Training

Compared to LSTM or GRU, Transformer, a sequence model based on multi-head self-attention mechanism, is more parallelizable for both training and inferring, and more interpretable [7]. Transformer has achieved compelling results in tasks that require maintaining long-range dependencies, such as neural machine translation [7], pre-training language models [22], text-to-speech synthesis [23], and speech recognition [24].

For music generation, Music Transformer [25] was the first work that applies the Transformer to symbolic music generation, Huang et al. used relative positional encoding [26] within the original Transformer architecture to capture relative timing information. MuseNet [27] used sparse kernels [28] to remember the long-term structure in the composition. More recent works [3–5] adopted Transformer-XL [6] that uses recurrent memory to enable the model to attend beyond a fixed context. In this work, we also leverage the powerful long-term dependency modeling of Transformer-XL for one-shot music generation.

Notably, SinGAN [1] has achieved compelling results on the task of unconditional generation from a single natural image, via a pyramid of fully convolutional light-weight GANs in a coarse-to-fine fashion. Then TOAD-GAN [2] was proposed for coherent style level generation following the one-shot training approach of SinGAN. Our work also involves the multi-scale training scheme into our SinTra model for better training from a single music segment.

<sup>1</sup> <https://github.com/qingweisong/SinTra>



**Figure 3.** SinTra’s multi-scale pipeline. Our model consists of a pyramid of Transformer-XLs, where both training and inference are done in a coarse-to-fine fashion. At each scale,  $T_n$  learns the information of target  $R^{2^{x_n}}$  by NLL loss,  $2^{x_n}$ th sequence means the sequence sampled by the  $2^{x_n}$ th note (visualize the sequence into piano-roll,  $2^{x_0} > \dots > 2^{x_N} \geq 4$ ). The input  $F^{2^{x_n}}$  to  $T_n$  from the previous output  $P^{2^{x_{n+1}}}$ , upscaled to the current temporal resolution (except for the coarsest stage which comes from  $B$ , a downsampled version of the real music). The generation process at stage  $n$  involves all Transformer-XLs  $\{T_N, \dots, T_n\}$  up to this level.

### 3. DATA REPRESENTATION

We use the method of language modeling to train the generation models for symbolic music. Therefore, by serializing polyphonic music into a single sequence, we express music as a series of discrete symbols determined by the data in music. For music generation learning from a single music segment, we present a novel pitch-group representation, which uses the index to represent various pitch group types by flexibly building a pitch-group dictionary of key-value pairs. The principle of pitch-group representation and the construction of pitch-group dictionary are shown in Figure 2. For a segment of music, the dictionary of pitch-group representation will not be very complex, so this representation method is feasible.

We treat all types of pitch group in each time step as elements, such as *harmony*, *single tone* and *interval*, which can use 1-dim sequence to represent polyphonic music. And the pitch-group representation can be regarded as an upgraded representation of pitch-based representation (support only monophonic music). However, this method is a double-edged sword, which will limit the output space and affect the diversity of generated music.

Piano-roll and event-based are the two most commonly data representations. Piano-roll representation used in DeepJ [20], MuseGAN [16], Binary MuseGAN [17], MIDI-Sandwich2 [18], and Music Transformer [25], is a 5-dim matrix representation of music where the vertical and horizontal axes respectively represent note pitch and time step. However, the piano-roll matrix is sparse since there are many zeros, only a few notes are attacked during each time step. Gale et al. [29] proved that sparse matrix has great computational potential. Treating piano-roll directly as dense matrix processing will waste computing resources.

Event-based representation used in Music Transformer [25] and LakhNES [3], means that the MIDI note events are converted into a sequence of tokens by a vocabulary containing 388 events. A one-minute song may need about 900 tokens in event-based representation. When the temporal resolution is 16th note and the tempo is 120 bpm, the piano-roll is a matrix whose shape is (480, 128) and the pitch-group is a sequence whose length is 480. It means that the length of the event-based is twice that of the other two methods. Besides, although events are generated in probability order, when there is not enough training data, it’s easy to generate unreasonable note (e.g., *Note\_on* event of the same note is generated before the *Note\_off* event or super long note). In addition, *TIME\_SHIFT* can possibly cause the confusion of time value information as proposed by Wu et al. [5].

Since traditional representation considers the universality of music representation, the coding space utilization is low when representing the information of a specific segment of music. For example, a segment of music only uses 20 pitches but still needs to use 128 pitch coding spaces. Or only 80 events appeared, but the encoding space of 388 events still needs to be used.

## 4. MUSIC GENERATION LEARNED FROM A SINGLE MUSIC SEGMENT

### 4.1 Pyramid of Transformer-XL Model

For learning reasonable musical structure and relative position relationship between pitch-group indexes of a single music segment, we adopt the multi-scale training mechanism to design a pyramid of Transformer-XLs  $\{T_0, \dots, T_N\}$ . Figure 3 shows the pipeline of SinTra for the generation of music samples. SinTra is trained

with a sequence pyramid of each stage’s real music  $R$ :  $\{R^{2^{x_0}}, \dots, R^{2^{x_N}}\}$  by the Negative Log Likelihood (NLL) loss, where  $R^{2^{x_n}}$  is a downsampled version of  $R^{2^{x_0}}$ . Each Transformer-XL  $T_n$  is responsible of producing music samples  $P^{2^n}$  with the corresponding scale of  $R^{2^{x_n}}$ . The generation of a music sample starts at the coarsest scale and sequentially passes through all models up to the finest scale. The Transformer-XLs have the different processing length and thus capture more details as we go up the generation process.

In order to get the sequence pyramid of  $R$  for each stage, we use different note-values to sample the original sequence. This kind of down-sampling method preserves the coarse-grained music structure information of the training music. We artificially define the note-value of each stage as  $2^{x_n}$ th, and the scale of the corresponding stage is  $\frac{N}{2^{x_0-x_n}}$ . For our training, the 16th note sequence of  $N$  length is sampled down to  $\frac{N}{2}$  and  $\frac{N}{4}$  by the 8th note and the 4th note respectively.

## 4.2 Processing of Multi-track Sequences

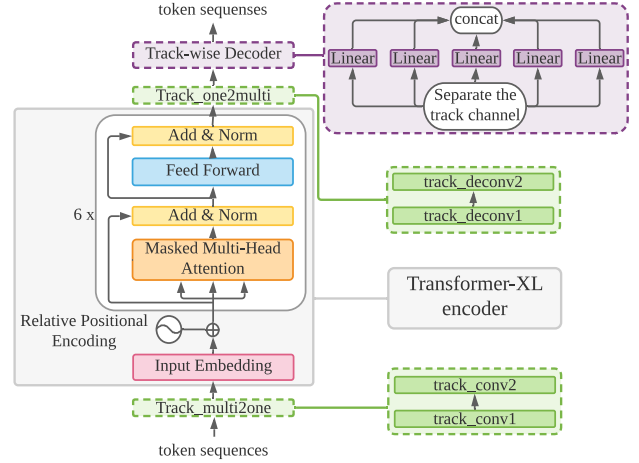
All the models of each scale have a similar architecture, as depicted in Figure 4. For maintaining the inter-track correlation, we use convolution operation to process multi-track music, and when decoding, the tracks are independent of each other to prevent interference. The `Track_multi2one` module is used to map multi-track sequence into a single sequence, and the `Track_one2multi` is the inverse process of `Track_multi2one`. The `Track-wise Decoder` module is used to decode each track independently.

## 4.3 Training and Inference

The first transformer generates bar by bar sequentially and the others refine each bar in a coarse-to-fine manner. In the 1<sup>st</sup> scale of training, the sequence sampled by the 4th note of the  $t^{th}$  bar  $R_t^4$  is fed into the model, and the  $(t+1)^{th}$  bar  $R_{t+1}^4$  is taken as training target with NLL loss. In the 2<sup>nd</sup> scale of training, to get the 8th sequence  $F_{t+1}^8$ , the output  $P_{t+1}^4$  to the previous scale needs to be upsampled, and the model will be trained with  $R_{t+1}^8$  as output. In the same way, the 3<sup>rd</sup> scale needs  $F_{t+1}^{16}$  as input and to be trained with output  $R_{t+1}^{16}$ . And the final output is  $P_{t+1}^{16}$  from the 3<sup>rd</sup> scale. The overall training process is shown in Eqn (1).

$$\begin{aligned}
 1st : P_{t+1}^4 &= Model_{1st}(R_t^4) \\
 loss_{1st} &= NLL(P_{t+1}^4, R_{t+1}^4) \\
 2nd : F_{t+1}^8 &= Upsample(P_{t+1}^4) \\
 P_{t+1}^8 &= Model_{2nd}(F_{t+1}^8) \\
 loss_{2nd} &= NLL(P_{t+1}^8, R_{t+1}^8) \\
 3rd : F_{t+1}^{16} &= Upsample(P_{t+1}^8) \\
 P_{t+1}^{16} &= Model_{3rd}(F_{t+1}^{16}) \\
 loss_{3rd} &= NLL(P_{t+1}^{16}, R_{t+1}^{16})
 \end{aligned} \tag{1}$$

In the inference of the NLP model, if the highest probability result is used as the prediction result every time, the



**Figure 4.** Network structure of each scale  $T_n$  (left) and three modules related to multi-track sequence processing (right). After the process of `Track_multi2one`, the shape of input sequence  $(1, track, T)$  becomes to  $(1, T)$ . The shape of the Transformer-XL encoder output  $(1, T, Feature)$  becomes to  $(1, Track, T, Feature)$  after `Track_one2multi`. Via the `Track-wise Decoder`, the final output token sequence shape is  $(1, Track, T)$ .

generated content will repeat easily, this method is called `Top1`. Hence, we adopt the `Top $p$`  method proposed by Holtzman et al. [30], that is, the model will sample the predicted results from several of the most likely results. In our inference process, the 1<sup>st</sup> scale is used to predict the next bar, so we can set a larger  $p = 0.9$  to increase the diversity of the generated samples. However, the latter two scales are used to enhance the details, if the variety is robust, the generated content will become overly-fragmented, so we set a smaller  $p = 0.3$ .

## 5. EXPERIMENT SETUPS

### 5.1 Data

We test our method both qualitatively and quantitatively on a variety of music files in MIDI format, containing well-known works of multiple styles of music. The MIDIs that we used are taken from the JSB Chorale dataset [31], classical music used in C-RNN-GAN [32] from the website<sup>2</sup>. We only use the JSB Chorale dataset for objective evaluation, because Music Transformer only supports single-track music. And we use all the MIDIs for subjective study.

### 5.2 Model Configurations & Training Setup

We performed our experiments under an NVIDIA GeForce GTX TITAN X graphics card, with PyTorch 1.4.0 running under CUDA 11.2. We implemented our multi-scale training framework (Figure 3) based on the Transformer-XL encoder. The encoder has 6 layers and the number of heads is 8 with a dimension of 32. The dimension of the embedding layer is 256 and the hidden layer dimension is 1024.

<sup>2</sup> <https://www.classicalarchives.com/>

The dropout ratio is set at 0.09. The length of training input tokens (processing length) and the memory length of 3 stages are 4, 8, 16, respectively.

Considering downsampling, to make SinTra learn the single training segment sufficiently, it is necessary to choose the note-value of each scale reasonably. The shorter notes appear in the training music segment, the smaller note-value used for sampling needs to be set in the finest scale, and the number of stages required for training is larger. Besides, the note-values of the adjacent scale are preferably a 2-fold relationship. Formally, the note-value of the finest scale is set to the shortest note that appears, the note-value of the 1<sup>st</sup> stage is set to the 4th note value (standard time unit in music).

We choose Music Transformer<sup>3</sup> and our variant, the model with only the first stage named single-stage SinTra, whose temporal resolution is set to the 16th note value, for comparison. We use Adam optimizer with  $\beta_1 = 0.5$ ,  $\beta_2 = 0.999$ ,  $\epsilon = e^{-8}$  and follow the same learning rate schedule in Transformer-XL [6]. We set the number of input bars as 12 and the number of generated bars as 32.

### 5.3 Subjective Study

We set up a blind listening test for human evaluation in which test-takers listen to 7 segments of music, one from the real music, three from SinTra and three from Music Transformer. In the test, test-takers will be asked the same set of questions after listening to each of the two test groups, namely, to rate them on a five-point scale about the following aspects:

- **Quality (Q):** Does the generated music sound pleasing overall?
- **Relevance (R):** Whether the generated music give you the same feeling as the real music?
- **Diversity (D):** Does the generated music have new arrangement that impresses you?

Finally, we collect responses from 50 subjects, of which 20 are classified as professional composers for their musical background. The 20 professions were asked to rate each music segment they heard from the music composition theory aspect, while 30 non-composers were asked to rate their subjective feelings.

### 5.4 Objective Evaluation

Objective evaluation in music generation is still an open question, though various metrics have been proposed, the feeling of music varies from person to person, it's hard to measure the quality of generated music. To quantitatively compare the performance differences between the three models, we use the following metrics to measure the similarity and diversity between generated music samples and the realistic single music segment.

<sup>3</sup> Since the official code is highly coupled with Magenta, data processing and training scripts are not shown explicitly. We thus used a third-party implementation (<https://github.com/jason9693/MusicTransformer-pytorch>) instead.

	Quality	Relevance	Diversity
SinTra	3.20/5.00	3.66/5.00	2.86/5.00
Music Trans.	2.34/5.00	2.38/5.00	2.54/5.00

**Table 1.** Results of subjective study in a five-point scale.

#### 5.4.1 KL Divergence

KL divergence measures the distance between two distributions. In this paper, pitch group indexes are used as the essential element for calculating the music distribution. The way we measure similarity is by calculating KL divergence between distributions of pitch group:

$$D_{kl}(P||Q) = \frac{1}{N_{sample}} \sum_{j=0}^{N_{sample}} \sum_{i=0}^{N_{type}} P(i) \log_2 \left( \frac{P(i)}{Q(i)} \right), \tag{2}$$

where  $N_{sample}$  means the number of generated samples,  $N_{type}$  means the number of pitch group types,  $P(i)$  means the  $i^{th}$  pitch group type of the generated samples,  $Q(i)$  means the  $i^{th}$  pitch group type of the original music. The smaller the KL divergence, the closer the two distributions.

#### 5.4.2 Pitch Group Overlap

Referring to the idea of IoU (Intersection over Union), we design a metric named pitch group overlap:

$$Overlap = \sum_{j=0}^N \frac{len(set(P) \cap set(Q))}{N * len(set(P) \cup set(Q))}, \tag{3}$$

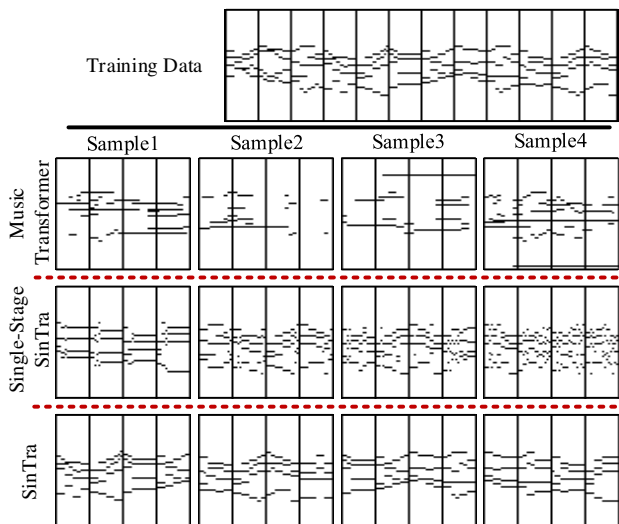
where  $set(P)$  means a set contains all pitch group types appearing in the sample,  $set(Q)$  means a set contains all types appearing in the real music segment. Overlap means the length of intersection between  $set(P)$  and  $set(Q)$  divided by the length of the union. Compared to KL divergence, the overlap can measure the difference between  $P$  and  $Q$  distributions at a coarser granularity. The larger the overlap, the more similar the pitch group types of two songs.

It should be noted that both KL divergence and overlap can only roughly measure the similarity between the two pieces of music. In the music generation task, if the similarity is too high, the diversity will be low. Conversely, if the similarity is too low, the correlation with real music will not be ideal. Therefore, to balance diversity and relevance, it is necessary to find a suitable similarity interval.

## 6. RESULTS AND ANALYSIS

### 6.1 Results of Subjective Study

The results shown in Table 1 indicate that our SinTra receives commendable scores, especially in relevance (R). After communicating with the subjects, they reflected that there were several fluent segments in samples, but the occasional messy notes led to the decline of the overall auditory perception. The output space dictionary constructed in the pitch-group representation ensures the correlation



**Figure 5.** Samples generated by the three models. Each model lists 4 samples. All models are trained under the same training data.

	Music Trans.	Single-stage	SinTra
KL-Div	40.53	27.84	<b>21.67</b>
Overlap	5%	55%	<b>79%</b>

**Table 2.** Results of objective metrics by Music Transformer, single-stage SinTra and SinTra on JSB Chorale dataset. The metrics of SinTra are marked in bold.

between generated music and real music. Besides, the coarse-to-fine fashion introduced by the pyramid structure enables the model to fully learn the features of training data. Although we introduce randomness at each stage in terms of diversity (D), the results show that the generated music is still close under a single training music.

## 6.2 Comparison with Previous Work

We compare music generation quality of SinTra with Music Transformer by: 1) we conduct experiments in the same one-shot learning condition, and since Music Transformer only supports single-track music, we use JSB Chorale dataset to evaluate the performance; 2) both models are asked to generate 10 segments in about 1 minute; 3) we set the velocity of all notes in musical pieces generated by models to a reasonable value (100).

The comparison results are shown in Table 2 and the local detail of the generate music is demonstrated in Figure 5. The convergence values of NLL loss shown in Table 3 also validate that Music Transformer does not fit the training music well. The music generated by SinTra is more realistic and fits more closely with real music.

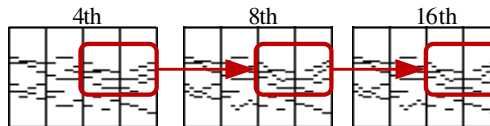
## 6.3 Method Analysis

### 6.3.1 Analysis on Pyramid Structure

To verify the effectiveness of the pyramid structure, we build the single-stage SinTra. As depicted in Figure 5,

	Music Trans.	Single-stage	SinTra
NLL	$\sim 1.0$	$10^{-3} \sim 10^{-4}$	$10^{-3} \sim 10^{-4}$

**Table 3.** Results of NLL loss by Music Transformer, single-stage SinTra and SinTra on JSB Chorale dataset when model converges. Both variants of SinTra can converge in  $10^{-3} \sim 10^{-4}$  while Music Transformer can't.



**Figure 6.** The effect of detail enhancement at each stage of SinTra. In the stage of the 4th note-value, simple notes will be generated but lack details, while in the 8th and 16th stages, the model refines some details.

we can see that the embryonic form of melody appears in single-stage SinTra, but is overly-fragmented and noisy. The KL divergence (27.84) and overlap (55%) shown in Table 2 are not ideal because of overly-fragmented notes. The comparison results show that SinTra can effectively suppress messy notes.

### 6.3.2 Analysis on Multi-stage Output

Figure 6 shows the output effect of SinTra at each stage. In the 1<sup>st</sup> stage (the 4th note-value), the model generates simple notes, ignores local details, and sketches roughly the outline of songs. Then the 2<sup>nd</sup> and 3<sup>rd</sup> stages enrich the contour in turn, bringing local detail changes. This model structure can effectively solve the problem of music generated by single-stage SinTra, while ensuring the generation quality, which can introduce randomness in each stage.

## 7. CONCLUSION AND FUTURE WORK

In this work, we propose SinTra, an inspiration generation framework to complete the task of one-shot learning in music generation. SinTra, consisting of a pyramid of Transformer-XL with a multi-scale training strategy, can learn both the musical structure and the relative positional relationship between notes of the single training music segment. Moreover, we present a novel pitch-group representation to ensure the relevance of generated samples and training music. The results of subjective study and objective evaluation show the effectiveness of SinTra for learning from single training data, generating music samples with a strong correlation with the training music. However, there is still room for improvement in the quality and diversity of generated music.

In the future, we will study controllable music generation that can integrate emotion- and style-controlled generations into SinTra. We will also consider large-scale generative pre-training to improve generation quality. We hope SinTra can be leveraged to enhance musicians' productivity and inspire them to compose higher quality music.

## 8. ACKNOWLEDGEMENTS

This paper was supported by the National Natural Science Foundation of China under Grant Number 61771440.

## 9. REFERENCES

- [1] T. R. Shaham, T. Dekel, and T. Michaeli, “SinGAN: Learning a generative model from a single natural image,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, Seoul, Korea, 2019, pp. 4570–4580.
- [2] M. Awiszus, F. Schubert, and B. Rosenhahn, “TOADGAN: Coherent style level generation from a single example,” in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Online, 2020, pp. 10–16.
- [3] 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,” in *Proceedings of the 20th Conference of the International Society for Music Information Retrieval*, Delft, Netherlands, 2019.
- [4] Y.-S. Huang and Y.-H. Yang, “Pop Music Transformer: Beat-based modeling and generation of expressive pop piano compositions,” in *Proceedings of the 28th ACM International Conference on Multimedia*, Seattle, United States, 2020, pp. 1180–1188.
- [5] X. Wu, C. Wang, and Q. Lei, “Transformer-XL based music generation with multiple sequences of time-valued notes,” *arXiv preprint arXiv:2007.07244*, 2020.
- [6] 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 *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, 2019, pp. 2978–2988.
- [7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*, Long Beach, United States, 2017, pp. 5998–6008.
- [8] I. Simon, D. Morris, and S. Basu, “MySong: Automatic accompaniment generation for vocal melodies,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Florence, Italy, 2008, pp. 725–734.
- [9] H. Tsushima, E. Nakamura, K. Itoyama, and K. Yoshii, “Function-and rhythm-aware melody harmonization based on tree-structured parsing and split-merge sampling of chord sequences,” in *Proceedings of the 18th Conference of the International Society for Music Information Retrieval*, Suzhou, China, 2017, pp. 502–508.
- [10] C.-H. Chuan and E. Chew, “A hybrid system for automatic generation of style-specific accompaniment,” in *Proceedings of the 4th International Joint Workshop on Computational Creativity*, London, United Kingdom, 2007, pp. 57–64.
- [11] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, “Recurrent neural network based language model,” in *Proceedings of the 11th Annual Conference of the International Speech Communication Association*, Chiba, Japan, 2010.
- [12] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [13] 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,” in *Proceedings of the 19th Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, 2014, pp. 1724–1734.
- [14] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *Proceedings of the 2nd International Conference on Learning Representations*, Banff, Canada, 2014.
- [15] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” in *Proceedings of the 28th Annual Conference on Neural Information Processing Systems*, Montreal, Canada, 2014.
- [16] 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, New Orleans, United States, 2018.
- [17] H.-W. Dong and Y.-H. Yang, “Convolutional generative adversarial networks with binary neurons for polyphonic music generation,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, Paris, France, 2018.
- [18] X. Liang, J. Wu, and J. Cao, “MIDI-Sandwich2: RNN-based hierarchical multi-modal fusion generation VAE networks for multi-track symbolic music generation,” *arXiv preprint arXiv:1909.03522*, 2019.
- [19] 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*, London, United Kingdom, 2018, pp. 2837–2846.

- [20] H. H. Mao, T. Shin, and G. Cottrell, “DeepJ: Style-specific music generation,” in *Proceedings of the 12th IEEE International Conference on Semantic Computing*, California, United States, 2018, pp. 377–382.
- [21] D. D. Johnson, “Generating polyphonic music using tied parallel networks,” in *Proceedings of the 6th International conference on Evolutionary and Biologically Inspired Music and Art*, Amsterdam, Netherlands, 2017, pp. 128–143.
- [22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [23] N. Li, S. Liu, Y. Liu, S. Zhao, M. Liu, and M. Zhou, “Neural speech synthesis with Transformer network,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, Hawaii, United States, 2019, pp. 6706–6713.
- [24] A. Mohamed, D. Okhonko, and L. Zettlemoyer, “Transformers with convolutional context for ASR,” *arXiv preprint arXiv:1904.11660*, 2019.
- [25] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, “Music Transformer: Generating music with long-term structure,” in *Proceedings of the 19th Conference of International Society for Music Information Retrieval*, Paris, France, 2018.
- [26] P. Shaw, J. Uszkoreit, and A. Vaswani, “Self-attention with relative position representations,” in *Proceedings of the Human Language Technology Conference of the NAACL*, New Orleans, United States, 2018.
- [27] C. Payne, “MuseNet,” *OpenAI Blog*, 2019. [Online]. Available: <https://openai.com/blog/musenet>.
- [28] R. Child, S. Gray, A. Radford, and I. Sutskever, “Generating long sequences with sparse Transformers,” *arXiv preprint arXiv:1904.10509*, 2019.
- [29] T. Gale, M. Zaharia, C. Young, and E. Elsen, “Sparse GPU kernels for deep learning,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Atlanta, United States, 2020, pp. 1–14.
- [30] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, “The curious case of neural text degeneration,” in *Proceedings of the International Conference on Learning Representations*, New Orleans, United States, 2019.
- [31] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, “Modeling temporal dependencies in high-dimensional sequences: application to polyphonic music generation and transcription,” in *Proceedings of the 29th International Conference on International Conference on Machine Learning*, Wisconsin, United States, 2012, pp. 1881–1888.
- [32] O. Mogren, “C-RNN-GAN: Continuous recurrent neural networks with adversarial training,” *arXiv preprint arXiv:1611.09904*, 2016.