Museformer: Transformer with Fine- and Coarse-Grained Attention for Music Generation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Music sequences are typically very long, e.g., with over $10{,}000$ tokens, making them hard to model by Transformer due to its quadratic complexity of self-attention. Although there are many Transformer variants aiming at modeling long sequences in natural language processing, directly using them in music generation is suboptimal, because music has both short-term structures and long-term structures, which is very different from text. In this paper, we propose Museformer, a Transformer with a novel fine- and coarse-grained attention for symbolic music generation. Specifically, with fine-grained attention, a token of a specific bar directly attends to all the tokens of the bars that are most relevant in regard to music structures (e.g., the previous 1st, 2nd, 4th and 8th bars, selected based on musical knowledge); with coarse-grained attention, a token only attends to the summarization of the other bars rather than each token of them so as to reduce computational cost. The advantages of our model are two-fold. First, it can capture both music structure-related correlations via the fine-grained attention, and other contextual information via the coarse-grained attention. Second, it is efficient and can well model over $4\times$ longer music sequences compared to its full-attention counterpart. Both objective and subjective experimental results demonstrate its ability to generate long music sequences with high quality and better structures. Our generated music samples can be found at `https://museformer.github.io/`.

1 Introduction

Symbolic music generation aims at generating music scores automatically and has drawn more and more attention in recent years [1–3]. Since music can be represented in organized sequences of discrete tokens just like text, Transformer-based models, which have demonstrated to work well on text generation, are increasingly applied in music generation [3–9]. While the self-attention mechanism empowers Transformer to directly capture the correlations over the whole input sequence, the quadratic complexity limits its scalability to music sequences especially for the multi-instrument polyphonic music, as they are typically very long, e.g., over $10{,}000$ tokens.

Although many Transformer variants in natural language processing (NLP) have been proposed to handle long sequences [10–12], directly applying them in music generation is suboptimal, because in addition to being very long, music has its unique bar-aware structures that are very different from text. Simply summarized, the structures include both the *short-term structures* and the *long-term structures* [13]. The former one means that adjacent bars usually follow the same rules, such as the same rhythmic patterns or chord progression; the latter one is mainly embodied in the long-distance repetitions, where a bar is a repetition of a previous bar at a long distance. Figure 1 shows an example of the song *Twinkle, Twinkle, Little Star*. As can be seen, every two consecutive bars repeat the same rhythmic pattern, which is a short-term structure, and at the 9th bar, the song starts to repeat the melodic pattern at the beginning, which is a long-term structure. In this paper, we call these bars that

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

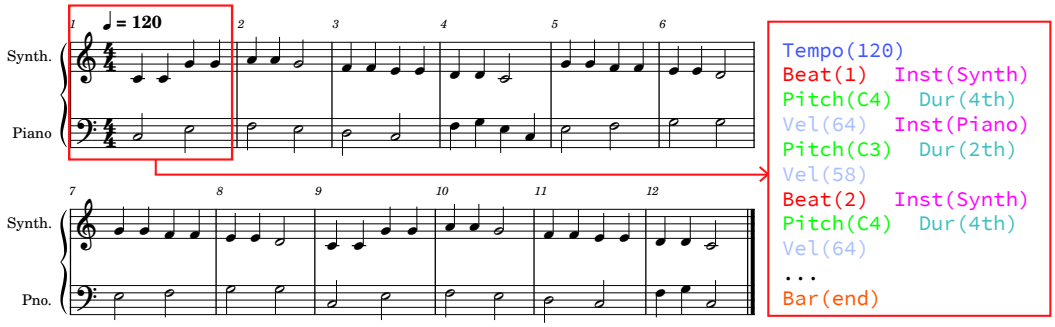


Figure 1: The music score of *Twinkle, Twinkle, Little Star* and its corresponding representation.

are the most relevant to the current bar in structure as the *structure-related bars*. Obviously, directly referencing these bars is essential for music generation, only by doing which the model can possibly generate structured music pieces. Note that the structure-related bars can be either close or far from each other, which obviously violates the common assumption in NLP that the most recent content is the most relevant. Thus, Transformers in NLP may not perform well on modeling music sequences, especially for capturing the long-term structures.

To better model the long music sequence in regard to the music structures, we propose Museformer, a long-sequence Transformer with a novel fine- and coarse-grained attention (FC-Attention in short) for symbolic music generation. It respects the fact that bar is an important unit for music, and all the attention operations are conducted with respect to the bar ranges. For a specific bar, we select out the previous *structure-related bars*, and leverage two different attention schemes namely fine-grained attention and coarse-grained attention on the structure-related bars and the other bars, respectively. Specifically, with fine-grained attention, a token of a specific bar directly attends to all the tokens of the structure-related bars, for better learning the structure-related correlations. With coarse-grained attention, the token attends to only the summarization of the other bars rather than each token of them, to decrease the computation and space complexity while also remain the necessary and concentrated information of these bars.

Note that in Museformer, the structure-related bars mentioned above are not necessarily the most recent adjacent bars but can also include some distant ones, to cater to the long-term structures of music. These bars are selected based on musical knowledge, and include both the short-term structure-related bars (e.g., the previous 1st and 2nd bars) and the long-term structure-related bars (basically they are 4 bars or a multiple of 4 bars away from the current, e.g., the previous 4th, 8th bars). Experimental results show that the proposed selection of the structure-related bars can help decrease the perplexity and generate music that exhibits better structures.

The main contributions of this work are summarized as follows:

- We propose Museformer, a Transformer model with a fine- and coarse-grained attention for music generation. It captures the correlations of structure-related bars via fine-grained attention for better learning the music structures, as well as the necessary and concentrated information of the other bars through their summarization via coarse-grained attention.

- We carefully select the structure-related bars based on musical knowledge, which cover both the short-term structures and the long-term structures.

- The computation complexity and space complexity are reduced from quadratic to linear with respect to the sequence length. This enables Museformer to scale up to long music sequences.

- Experimental results show that Museformer can generate full-song level music with high quality and better structures. The generated music samples can be found at `https://museformer.github.io/`. Source code is currently uploaded in the supplement and will be released on GitHub.

2 Related Work

2.1 Symbolic Music Generation

Symbolic music generation uses machines to compose music scores automatically. This fascinating idea has been attracting more and more people to work on it, and the solutions continuously evolve from rules [1] or probability models [14, 15, 2] to deep learning methods [4–9].

In recent years, the Transformer-based models have achieved great success in many text generation tasks, thus have also been increasingly applied in the similar music generation tasks. Huang et al. [3] apply Transformer in generating music for the first time, and show that it can achieve the state-of-the-art performance. While Transformer shows promising results in music generation as it does in other sequential data processing tasks, it suffers from the quadratic complexity, which limits the applications on the typically long music sequences. To resolve this problem, works on symbolic music generation come up with different solutions. One popular solution is to modify the representation method that transfers the symbolic music file (e.g., the MIDI format) to the sequential tokens, such as REMI in [16] and compound word in [17]. While the new representation methods indeed decrease the sequence length of music to certain extent, Transformer still cannot handle the overly long music sequences. Another solution is to use a long-sequence Transformer variant as the backbone. For example, Huang and Yang [16] leverage Transformer-XL [10], and [17] leverages Linear Transformer with a kernel-based linearized attention. We will review more long-sequence Transformer in §2.2. Although these models can process long sequences, they are proposed in regard to the NLP problems. In view of the fact that music has its unique structures different from text, directly applying them in music generation is suboptimal, as we have mentioned in §1 and will give a more extensive discussion in the following section. The existing Transformer-based models for music generation directly borrow the long-sequence Transformers from NLP to generate long music sequence, without considering music's characteristics. To the best of our knowledge, we are the first to propose a Transformer variant that are targeted at symbolic music generation.

2.2 Long-Sequence Transformers

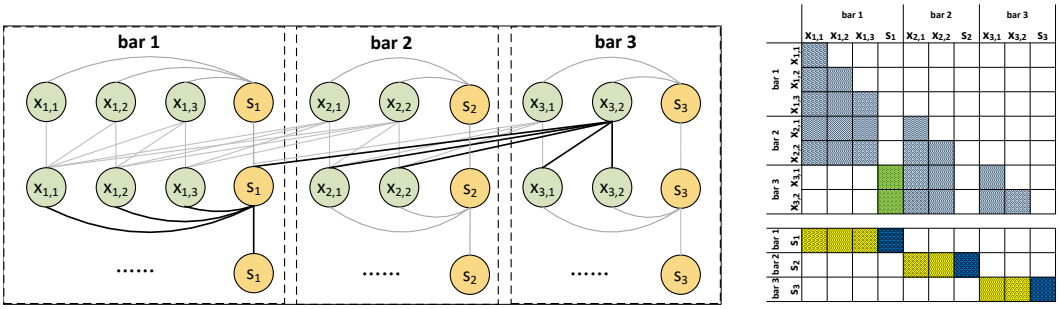
Various kinds of Transformer variants in NLP have been proposed to tackle the document-level tasks. One famous kind is the recurrent Transformers, which chunk the sequence into blocks, and encode each block with the gradient-stopped memory of several previous blocks [10, 18, 19]. Another one called sparse attention is to sparsify the attention matrix with predefined patterns such as sliding windows, or with techniques such as hashing functions [11, 20, 21]. Some other models use linearized attention, which approximates the pairwise softmax-based attention with kernel-based computation in linear time complexity [12, 22].

The existing works on music generation directly adopt some of those long-sequence Transformers to process long music sequences, but we argue that it is optimal due to the different structures of music compared to text. A common and reasonable assumption in NLP is that the nearest content is the most relevant, but it is obviously not true for the music structures. As we have shown in §1 that besides the short-term structures, there are many long-term structures in music. Therefore, models focusing on the most recent content, such as the recurrent or the sliding window-based Transformers, can be able to capture the short-term structures but may not well capture the long-term structures. As for the linearized attention whose attention ranges can cover the whole input, it only approximates the attention weights on the whole sequence and does not model the structure-related bars in a fine-grained and targeted way, which harms the performance, as we will show in the experiments.

3 Museformer

In order to handle the typically long music sequences, as well as capture both the short- and long-term music structures, we propose Museformer for symbolic music generation. It basically follows the original Transformer structure [23]. The core component of Museformer is the fine- and coarse-grained attention (FC-Attention), which replaces the original self-attention module to tackle the challenges in long music sequence modeling.

Museformer takes the music tokens as input. For each bar in the input, we insert a *summary token* after it to aggregate its local information. The embedding layer embeds the overall token sequence



(a) Visualization of the FC-Attention. The dark lines represent the information flow towards $x_{3,2}$

(b) The corresponding attention layouts.

Figure 2: The fine- and coarse-grained attention. It shows a toy example of 3 bars, where for each bar, only the previous 1st bar is regarded as the structure-related bar. The bottom parts of the two subfigures correspond to the summarization step, while the upper parts correspond to the combination step based on the selected structure-related bars.

into a vector space, and concatenates the bar embedding and the beat embedding as the position information for music, followed by a linear projection. The following Museformer layers model the contextual representation, and the hidden state output by the last layer is fed into the output layer to predict the next token.

We first introduce the necessary preliminaries and notations in §3.1, then describe the details of FC-Attention in §3.2, and finally discuss the merits of our model in §3.3

3.1 Preliminaries and Notations

Attention The attention mechanism [23] is the basis of FC-Attention. It receives two sequential inputs, namely target $\boldsymbol{X} \in \mathbb{R}^{n_{\text{tgt}} \times d}$ and source $\boldsymbol{X}' \in \mathbb{R}^{n_{\text{src}} \times d}$, and computes the contextual representation for each item $\boldsymbol{x}_i \in \mathbb{R}^{1 \times d}$ of target $\boldsymbol{X}$ as

$$\text{Attn}(\boldsymbol{x}_i, \boldsymbol{X}') = \text{softmax}\Big(\frac{\boldsymbol{x}_i \boldsymbol{W}_Q (\boldsymbol{X}' \boldsymbol{W}_K)^T}{\sqrt{d}}\Big) \boldsymbol{X}' \boldsymbol{W}_V. \tag{1}$$

Here, we formulate attention with respect to a single item $\boldsymbol{x}_i$, because in our model, we assign different $\boldsymbol{X}'$ for different $\boldsymbol{x}_i \in X$. We use the multi-head version in our implementation, but omit its equations for simplicity.

Token Notations Museformer receives the music token sequence as input, denoted by $X = (X_1, \ldots, X_b)$, where b is the number of music bars, and each bar $X_i = (x_{i,1}, \ldots, x_{i,|X_i|})$ contains $|X_i|$ tokens. We then insert a summary token after each bar to summarize its local information. The summary token for the i-th bar is denoted by s_i, and the summary token sequence is denoted as $S = (s_1, \cdots, s_b)$.

3.2 Fine- and Coarse-Grained Attention

In this section, we describe the proposed FC-Attention in detail. The basic idea of FC-Attention is that, instead of attending to all the tokens of the sequence which causes the quadratic complexity, a token of a specific bar only directly attends to the structure-related bars that are essential for generating structured music (fine-grained attention). For the other bars, the token only attends to their summary tokens to obtain concentrated information (coarse-grained attention). To achieve this, we need to summarize the local information of each bar through a *summarization* step, and decide which bars to be attended via fine-grained attention through *structure-related bar selection*, and finally aggregate the fine-grained and coarse-grained information through a *combination* process.

Summarization In the summarization step, we aggregate the information of each bar into the corresponding summary token. For the i-th bar, given the representation of summary token $\boldsymbol{s}_i \in \mathbb{R}^{1 \times d}$, and that of the music tokens $\boldsymbol{X}_i = [\boldsymbol{x}_{i,1}, \ldots, \boldsymbol{x}_{i,|X_i|}] \in \mathbb{R}^{|X_i| \times d}$, the summarization of this bar is

$$\tilde{\boldsymbol{s}}_i = \text{Attn}\big(\boldsymbol{s}_i, [\boldsymbol{X}_i, \boldsymbol{s}_i]\big), \tag{2}$$

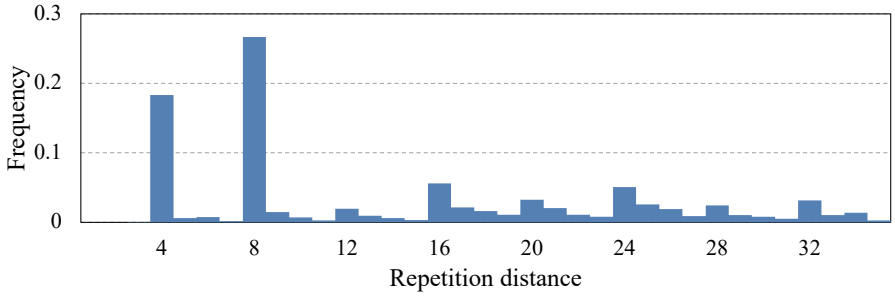


Figure 3: Statistical results on the repetition intervals.

where $\text{Attn}(\cdot)$ is defined in Equation 1, $[\cdot]$ is the concatenation operation. Each summary token attends to music tokens that belong to its corresponding bar as well as the summary token itself.

Structure-Related Bar Selection One important issue is that what are the structure-related bars. We conduct a statistical analysis on the POP909 dataset[24], where the segments of a song are annotated, through which we can easily calculate the distribution of the repetition intervals, i.e., the distance between two repetitive bars. As shown in Figure 3, music tends to repeat every 4 bars or its multipliers in most cases, and this also complies with our musical expertise and experience. Based on this musical knowledge, we select 8 bars as the structured-related bars, which include the previous 1st, 2nd, 4th, 8th, 16th, 24th and 32nd bars, where the 1st and the 2nd bar are for capturing the short-term structures while the others are for capturing the long-term structures.

Combination In the combination step, we aggregate the information of the tokens belonging to the structure-related bars or within the current bar, as well as the summarization of the other bars, so as to update the contextual representation of music tokens. The updated representation for $x_{i,j}$ is

$$\tilde{\boldsymbol{x}}_{i,j} = \text{Attn}\Big(\boldsymbol{x}_{i,j}, [\boldsymbol{X}_{N(i)}, \boldsymbol{X}_{i,k\leq j}, \tilde{\boldsymbol{S}}_{\bar{N}(i)}]\Big). \tag{3}$$

In this equation, $N(i)$ is the set of the indices of the previous structure-related bars with respect to the i-th bar, and $\bar{N}(i)$ is the set of the indices of other previous bars. $\boldsymbol{X}_{N(i)}$ corresponds to the matrix formed by stacking $\{\mathbf{X}_i \,|\, i \in N(i)\}$, and similarly $\tilde{\boldsymbol{S}}_{\bar{N}(i)}$ corresponds to the matrix formed by stacking $\{\tilde{\boldsymbol{s}}_i \,|\, i \in \bar{N}(i)\}$. Finally, $\boldsymbol{X}_{i,k\leq j}$ corresponds to the matrix that stacks $\{\boldsymbol{x}_{i,k} \,|\, \boldsymbol{x}_{i,k} \in \boldsymbol{X}_i,\, k \leq j\}$. In other words, the music token only attends to those music tokens that are in the structure-related bars, or its previous tokens within the current bar. For other bars, it only attends to their summary tokens.

3.3 Merits of Museformer

We explain how the designs of FC-Attention make Museformer more suitable for the music generation tasks compared to the existing Transformers.

First, the structure-related bar selection ensures that FC-Attention can capture both the short-term and the long-term structures of music. Compared to most long-sequence Transformers that only focus on the most recent content (e.g., recurrent Transformers and sliding window attention), the selected bars include both the neighboring ones to capture the short-term structures and the distant ones to capture the long-term structures.

Secondly, the combination of the fine-grained attention and the coarse-grained attention makes it capable of concentrating on the structure-related bars, and also remaining the necessary information of the other bars through the summarization. In contrast, sparse attention models leverage strong inductive biases to make each specific token only attend to a subset of tokens and simply drop the others, which may loses necessary information of the whole sequence.

Thirdly, Museformer is efficient for handling long music sequences. A music token of a specific bar only directly attends to those music tokens in the structure-related bars, and indirectly attends to the other bars via the summary tokens. Compared to the number of all the music tokens, the number

of summary tokens is much smaller, which drastically decreases the time complexity and memory consumption.

4 Experiments

4.1 Experiment Settings

In this subsection, we give a brief introduction to the experiment settings. Please refer to Appendix A for more details.

Dataset and Music Representation We select the widely-used Lakh MIDI (LMD) dataset [25], which is a multi-instrument dataset. We preprocess it and obtain the final dataset for the experiments, which contains $29{,}940$ songs ($1{,}727$ hours), each song contains 95 bars on average, and each bar is represented by 158 tokens on average. We randomly split them by $8/1/1$ for training/validation/test, respectively. To represent symbolic music as token sequences, we use a REMI-like representation method [16] for its simplicity. Figure 1 shows an example.

Implementation We implement our model with PyTorch[1] and fairseq[2]. For efficiently computing FC-Attention, we transfer the attention layouts into the blocksparse forms and compute them with OpenAI Triton[3]. We also write CUDA kernels to generate the attention layouts efficiently.

Model and Training Configurations The core parameters of our model include layer number $= 4$, hidden size $= 512$, number of attention heads $= 8$, and FFN hidden size $= 2{,}048$. During training, the batch size is set to 4 songs. Following [23], we use the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\varepsilon = 10^{-9}$, and the learning rate is warmed up over the first $16{,}000$ steps linearly to a peak value of 0.00005, and then decreases based on the inverse square root of the steps. The L2 weight decay is set to 0.01. During inference, we use the top-k sampling with $k = 8$, and generate the sequence tokens continuously until the end-of-sentence token is generated.

Baselines We select four typical and widely-used baseline models, most of which have been adopted in music generation:

- **Music Transformer** [3] is a vanilla Transformer with a memory-efficient "skewing" relative position embedding implementation.
- **Transformer-XL** [10] is a recurrent Transformer that encodes the sequence chunk by chunk, using the previous chunks as the memory to carry the past information.
- **Longformer** [11] leverages a sliding window sparse attention to encode the local information.
- **Linear Transformer** [12] uses a kernel-based attention to decrease the computation complexity and memory consumption.

All the baseline models are set with comparable hyper-parameters as Museformer. For Music Transformer that adopts a full attention and thus cannot process long sequences due to the memory limit, we chunk the input into multiple samples during training, while allows it to generate longer sequences during inference, to test the generalization on long music sequences.

Objective Evaluation In the objective evaluation, we use **Perplexity (PPL)** to evaluate the performance, which is a very common metric in the generation tasks to measure whether a model can correctly predict the tokens. The lower the value is, the better the model can correctly predict. In our experiment, perplexity is calculated on the test set and over the first $1{,}024$, $5{,}120$, or $10{,}240$ tokens of each sample.

Subjective Evaluation The most canonical way to evaluate a music generation model is the human listening test. We apply each model to randomly generate 100 music pieces, and invite 10 people, where 7 of them have some music-related learning experiences, to score these music pieces. Specifically, for each participant, we randomly construct 5 groups, where each group contains 5 music pieces that are generated by Museformer and the 4 baseline models, respectively. Participants are

[1] `https://pytorch.org`

[2] `https://github.com/pytorch/fairseq`

[3] `https://github.com/openai/triton`

Table 1: The results of objective evaluation. The numbers in the parentheses are the sequence lengths.

	PPL (1,024)	PPL (5,120)	PPL (10,240)
Music Transformer [3]	1.66	1.77	2.55
Transformer-XL [10]	**1.64**	1.45	1.43
Longformer [11]	1.65	1.46	1.45
Linear Transformer [12]	1.86	1.67	1.64
Museformer (ours)	**1.64**	**1.41**	**1.35**

Table 2: The scoring results of subjective evaluation. ST and LT stand for short-term and long-term respectively. Mean $\pm$ standard deviation is reported.

	Musicality	ST structure	LT structure	Overall
Music Transformer [3]	6.00 ± 2.21	6.90 ± 1.76	5.30 ± 2.58	5.90 ± 1.90
Transformer-XL [10]	6.10 ± 2.19	7.40 ± 1.81	6.26 ± 2.78	6.44 ± 2.01
Longformer [11]	6.46 ± 1.81	7.60 ± 1.47	6.18 ± 2.54	6.44 ± 1.72
Linear Transformer [12]	6.06 ± 1.99	6.92 ± 2.03	5.78 ± 2.64	6.30 ± 1.84
Museformer (ours)	$\mathbf{6.88 \pm 1.95}$	$\mathbf{7.86 \pm 1.51}$	$\mathbf{6.72 \pm 2.74}$	$\mathbf{7.12 \pm 1.81}$

asked to score these music pieces from 1 (lowest) to 10 (highest) over the following subjective metrics: 1) **Musicality**: whether the music is pleasant and interesting, and is real enough just like human-made music. 2) **Short-term structure**: whether the music shows good structure in neighboring content, such as good repetitions and reasonable development. 3) **Long-term structure**: whether the music shows good structure in long distance, such as song-level repetitions and long-distance connections. 4) **Overall**: an overall score. We also calculate the **preference score** based on the overall score, which is defined as the ratio of the winning times.

4.2 Comparison with Previous Models

Table 1 shows the results of the objective evaluation. As we can see: 1) Music Transformer achieves comparable performance with other models on generating short music pieces ($1{,}024$ tokens), but undergoes a severe declining when moving to longer ones, which implies that a model trained on short music sequences cannot well generalize to long music sequences. Accordingly, an appropriate long-sequence Transformer model is crucial in music generation tasks. 2) Although Linear Transformer drastically decreases the complexity, its performance is relatively poor compared to other long-sequence Transformers on the music generation task, since it loses accurate information of both the short- and long-term structure-related bars. 3) The proposed Museformer can consistently achieve the best results on various sequence lengths, especially on generating long music sequences, which demonstrates the effectiveness of Museformer on music generation tasks.

Furthermore, we present the results of the subjective evaluation in Table 2 and Figure 4, which show that Museformer achieves the best performance compared to all the best models. Specifically, 1) Museformer gets the highest scores on all the metrics, and the improvement is statistically significant ($p \leq 0.05$, see Appendix C for the details). 2) On the structure metrics, especially the long-term structure, Museformer exceeds other models by a large gap, indicating that the proposed FC-Attention can empower the model to capture the correlations in distant bars.

4.3 Ablation Study

We verify the effectiveness of the Museformer components as follows: 1) To show the effectiveness of the coarse-grained attention, we compare Museformer with the setting **w/o coarse-grained attention**, where each music token only directly attends to the music tokens in the structure-related bars. 2) To show the effectiveness of the structure-related bar selection for the fine-grained attention, we compare

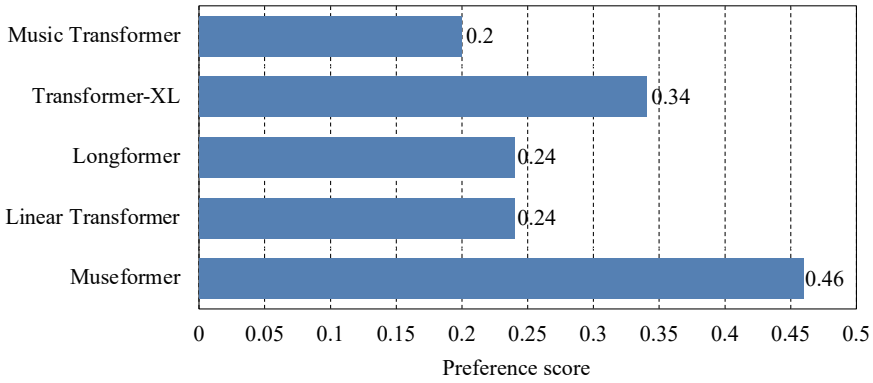


Figure 4: The preference scores of the subjective evaluation.

Table 3: The results of ablation study.

	PPL (1,024)	PPL (5,120)	PPL (10,240)
Museformer (ours)	**1.64**	**1.41**	**1.35**
w/o coarse-grained attention	1.65	1.42	1.38
w/o structure-related bar selection	1.65	1.43	1.39

Museformer with the setting **w/o structure-related bar selection**, where we directly use the most recent 8 bars for the fine-grained attention instead of the selected structure-related bars.

The results are presented in Table 3. Museformer consistently achieves better performance with coarse-grained attention and structured-related bar selection. The contribution of these two settings grows when the sequence length is getting longer, and this is reasonable because: 1) The coarse-grained attention helps Museformer obtain summarization of more previous bars, if the sequence is long and there are more bars in the sequence. 2) There exist more structures in a long music sequence, where the selected bars can help capture the correlations.

4.4 Complexity Analysis

We further analyze the complexity of Museformer. Suppose the sequence length is n, the average bar number is b, the average bar length is m, and the number of selected structure-related bars is k. For FC-Attention, the time complexity of the summarization step is $O(n)$, and that of the update step is $O((km + b)n)$, so the overall complexity of FC-Attention is $O((km + b)n)$. Since k and m are small numbers, and b is quite small in the real world, the complexity of FC-Attention is linear to the number of tokens.

With the decreased complexity, Museformer can encode much longer sequence than the full attention, allowing us to generate full-song music. We train Museformer and vanilla Transformer on the validation set to test the memory consumption and the running time. With batch size $= 1$, we increase the maximum sequence length until we use up the 32GB of memory on a V100 GPU, and record the peak memory consumption[4] at each length and the running time for training one epoch. Figure 5 shows the results. As we can observe, the memory consumption of Museformer increases linearly with respect to the maximum sequence length, and can process over $4\times$ longer music sequences than vanilla Transformer, making it feasible in generating full-song music. Museformer also achieves faster training times when the maximum sequence length is greater than around $5{,}000$, which is a common length for music sequences. Note that, after the maximum sequence length exceeds $32{,}768$, the memory consumption of Museformer does not increase as fast as in the previous range, and this may because there is only a few samples in the dataset that are longer than $30{,}000$, and the memory consumption is also related to the lengths of the bars.

[4]We use the result returned by torch.cuda.max_memory_allocated().

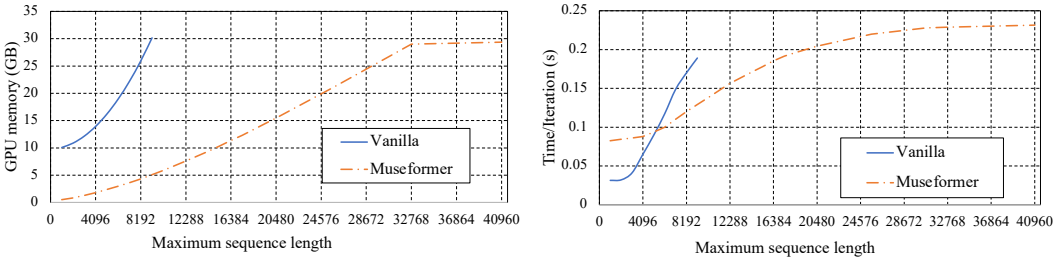


Figure 5: Memory consumption and running time of Museformer compared to vanilla Transformer (full attention).

4.5 Case Study

We present the piano track of a generated music piece in Figure 6, which exhibits the reasonable short-term structure and long-term structure. As we can see, the 21st bar repeats the 19th bar, which is a short-term structure commonly existing in music. Moreover, the 35th–38th bars repeat the 19th–22nd bars, and the interval is large enough (16 bars) to make it a song-level long-term structure.

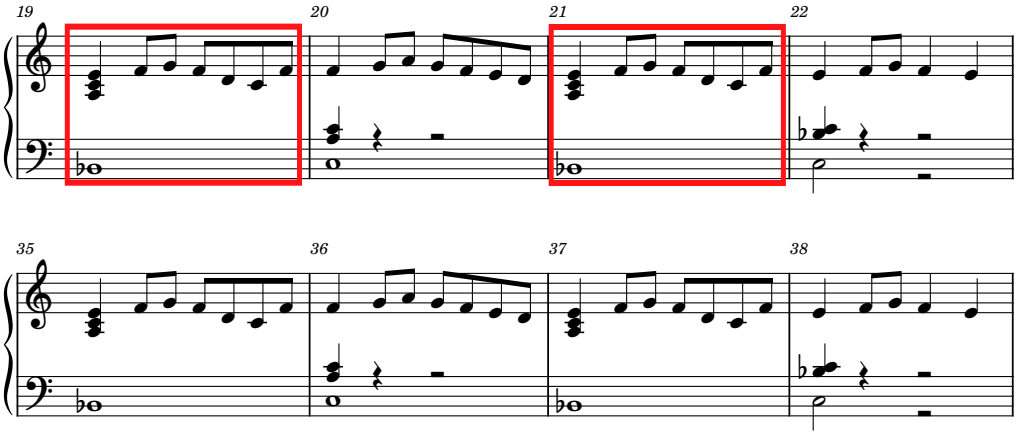


Figure 6: Case study.

5 Conclusion

Music generation is an interesting and challenging problem. In this paper, we propose Museformer to tackle the long-sequence music generation task with a carefully designed fine- and coarse-grained attention. The fine-grained attention is applied to selected structure-related bars, which can successfully capture both the short- and long-term music structures, while the coarse-grained attention can get a sketch of all the other bars based on the summary tokens. Experimental results show that the proposed Museformer is superior than previous music generation models and long-sequence transformer models, especially on generating music with song-level structure.

The limitation of Museformer is that, the efficiency of the dynamic sparse attention largely depends on a very sophisticated implementation, and the running speed and the memory consumption still have large room for improvement compared to the theoretical complexity. In future work, we will explore the possibility of the further efficiency improvement, and apply the FC-Attention to the music understanding tasks.

References

[1] H. A. García Salas, A. Gelbukh, H. Calvo, and F. Galindo Soria, "Automatic music composition with simple probabilistic generative grammars," *Polibits*, no. 44, pp. 59–65, 2011.

[2] A. K. Yanchenko and S. Mukherjee, "Classical music composition using state space models," *arXiv preprint arXiv:1708.03822*, 2017.

[3] 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 *International Conference on Learning Representations*, 2018.

[4] 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.

[5] A. Muhamed, L. Li, X. Shi, S. Yaddanapudi, W. Chi, D. Jackson, R. Suresh, Z. C. Lipton, and A. J. Smola, "Symbolic music generation with transformer-gans," in *AAAI*, 2021, pp. 408–417.

[6] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *ICML*, 2018, pp. 4364–4373.

[7] Z. Ju, P. Lu, X. Tan, R. Wang, C. Zhang, S. Wu, K. Zhang, X. Li, T. Qin, and T.-Y. Liu, "Telemelody: Lyric-to-melody generation with a template-based two-stage method," *arXiv preprint arXiv:2109.09617*, 2021.

[8] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, *PopMAG: Pop Music Accompaniment Generation*. New York, NY, USA: ACM, 2020, pp. 1198–1206.

[9] Z. Sheng, K. Song, X. Tan, Y. Ren, W. Ye, S. Zhang, and T. Qin, "Songmass: Automatic song writing with pre-training and alignment constraint," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 15, 2021, pp. 13 798–13 805.

[10] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," *arXiv preprint arXiv:1901.02860*, 2019.

[11] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[12] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *ICML*, 2020, pp. 5156–5165.

[13] J. de Berardinis, S. Barrett, A. Cangelosi, and E. Coutinho, "Modelling long-and short-term structure in symbolic music with attention and recurrence," in *Proceedings of The 2020 Joint Conference on AI Music Creativity*, 2020, pp. 1–11.

[14] M. Farbood and B. Schöner, "Analysis and synthesis of palestrina-style counterpoint using markov chains," in *ICMC*, 2001.

[15] M. Allan, "Harmonising chorales in the style of johann sebastian bach," *Master's*, 2002.

[16] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *ACM MM*, 2020, pp. 1180–1188.

[17] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, "Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs," *arXiv preprint arXiv:2101.02402*, 2021.

[18] J. W. Rae, A. Potapenko, S. M. Jayakumar, C. Hillier, and T. P. Lillicrap, "Compressive transformers for long-range sequence modelling," in *ICLR*, 2019.

[19] S. Ding, J. Shang, S. Wang, Y. Sun, H. Tian, H. Wu, and H. Wang, "Ernie-doc: A retrospective long-document modeling transformer," in *ACL*, 2021, pp. 2914–2927.

[20] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[21] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang *et al.*, "Big bird: Transformers for longer sequences," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 283–17 297, 2020.

[22] H. Peng, N. Pappas, D. Yogatama, R. Schwartz, N. Smith, and L. Kong, "Random feature attention," in *International Conference on Learning Representations*, 2020.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, l. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017, pp. 5998–6008.

[24] 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," *arXiv preprint arXiv:2008.07142*, 2020.

[25] C. Raffel, *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Columbia University, 2016.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We do not see any potential negative societal impacts of our work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [Yes]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]