VARIABLE-LENGTH MUSIC SCORE INFILLING VIA XLNET AND MUSICALLY SPECIALIZED POSITIONAL ENCODING

Chin-Jui Chang
Research Center for IT Innovation,
Academia Sinica
reichang182@gmail.com

Chun-Yi Lee
Department of Computer Science,
National Tsing Hua University
cylee@cs.nthu.edu.tw

Yi-Hsuan Yang
Yating Music Team
Taiwan AI Labs
yhyang@ailabs.tw

ABSTRACT

This paper proposes a new self-attention based model for *music score infilling*, i.e., to generate a polyphonic music sequence that fills in the gap between given past and future contexts. While existing approaches can only fill in a short segment with a fixed number of notes, or a fixed time span between the past and future contexts, our model can infill a variable number of notes (up to 128) for different time spans. We achieve so with three major technical contributions. First, we adapt *XLNet*, an autoregressive model originally proposed for unsupervised model pre-training, to music score infilling. Second, we propose a new, musically specialized positional encoding called *relative bar encoding* that better informs the model of notes' position within the past and future context. Third, to capitalize relative bar encoding, we perform *look-ahead onset prediction* to predict the onset of a note one time step before predicting the other attributes of the note. We compare our proposed model with two strong baselines and show that our model is superior in both objective and subjective analyses.

1. INTRODUCTION

A growing body of research work has adopted deep learning techniques to generate music sequentially, taking only the past context as a condition while generating. This paper deals with a different setting where both the past and future context are given, called *music score infilling* [1–9]. As depicted in Figure 1, the goal of this task is to generate a short piece of symbolic music that fits in the middle gap. The generated piece is expected to meet certain requirements, e.g., being coherent with the contexts, and having an appropriate duration span such that the generated piece will not overflow to the outside of the designated region. Models with such a capability are useful in a few scenarios. For instance, one may have an inspiration to write two segments of melodies but somehow do not know how to connect them [5]. Or, a music piece may have an impaired part in the middle that requires restoration [10].
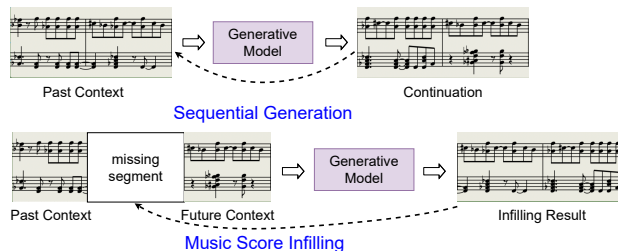
Figure 1: Comparison between sequential generation and music score infilling (a.k.a., music score inpainting).

As reviewed in Section 2, existing models for score infilling can be categorized by their representation of musical scores. Among them, we are interested in the case when music is represented as a sequence of *event tokens* such as note-on and note-off [11], for such a *token*-based representation facilitates the use of the self-attention based Transformers [12] for model building, which has been shown to outperform recurrent neural networks (RNN) in modeling sequences [13, 14]. [1] However, Transformers are originally designed for sequential generation: predicting the future given the past tokens. Adapting it to account for both the past and future contexts while generating the missing token sequence in the middle may not be trivial.

The Infilling Piano Performances (IPP) [9] is the only pioneering work adapting Transformers for music score infilling, to our best knowledge. It achieves so with a simple approach of "reordering": the past and future contexts are concatenated and placed *before* the missing segment as the *prompt*. A standard Transformer decoder is then trained to autoregressively generate a continuation of a given prompt. We note that this approach has a strong limitation: the past and future contexts need to consist of a fixed number of c notes each, and the missing segment a *fixed* number of k notes (c and k can be different). Moreover, as reported in [9], empirically their model works well only for infilling a *short* sequence with $k = 16$ missing notes; for larger k, its performance deteriorates and the infilled sequence cannot connect well with the future context.

In *text infilling* [25–30], the music infilling counterpart in natural language processing (NLP), many approaches have been based on Transformers. For instance, Infilling Language Model (ILM) [29] uses special tokens to in-

---

[1] Token-based representations have also been heavily adopted by recent Transformer models for sequential MIDI music generation [15–24].

form the language models where to infill text. FELIX [30] enables BERT [13] to solve the infilling task by letting BERT predict [PAD] tokens for redundant masked positions. Both models can perform *variable-length text infilling*, but they were both tested only on infilling short sequences with less than 10 consecutive tokens (words).

This paper proposes a new self-attention based model to attain *long* and *variable-length music score infilling*. In our experiment, the model is able to generate a variable-length polyphonic infilled sequence with up to 768 tokens, or 128 notes, given the past and future contexts. [2] Moreover, our model is able to infill spans of different length (e.g., 2 bars to 4 bars) without re-training for each span length. We achieve so with three technical contributions. First, we employ *XLNet* [31], a Transformer encoder-based model, as the model architecture for the first time for music generation. Unlike other bidirectional models such as BERT [13], XLNet can attend to the past and future contexts while maintaining its autoregressive predicting order. We show that music infilling can be attained by XLNet via a specific factorization order of the token sequence.

Second, we point out that the original XLNet can infill only a fixed-length token sequence, because it relies on the *vanilla* positional encoding [3] that requires the length of the missing segment to be known and fixed in advance. We propose a *musically specialized positional encoding* and a modification of the two-stream attention mechanism of XLNet to make it feasible for variable-length infilling. Specifically, our model represents the distance between two tokens in terms of *the number of bars* between them, rather than the exact number of intermediate tokens. Doing so, the positions of notes are also specified in a musically more meaningful way.

Third, with our special positional encoding, we need to know the musical position of the next note to be predicted in the autoregressive generation process. Therefore, we adapt the multi-output methodology of the Compound Word (CP) Transformer [24] to perform *look-ahead onset prediction* at each timestep. Specifically, each time, our model predicts the *content*-related tokens of the *current* note (i.e., `PITCH`, `DURATION`, `VELOCITY`, and `TEMPO`), and the *position*-related tokens (i.e., `BAR` and `SUB-BEAT`) of the *next* note to look one note ahead.

For evaluation, we compare our model with the two text infilling state-of-the-arts, ILM [29] and FELIX [30], that we extend to accept the same token representation as ours and to generate variable-length infilled sequences. The results show that our model outperforms these two strong baselines in both objective and subjective analyses.

For reproducibility, we open source our code at GitHub, along with examples of the infilling result. [4]

---

[2] As described in Section 3.1, we represent a musical note with 6 tokens (`PITCH`, `DURATION`, etc) in total.

[3] Unlike RNNs, the Transformers do not have a built-in notion of the sequential order of tokens, and thus need to rely on the so-called *positional encodings* that "assign" positions to each token [32, 33]. This is usually done by using a token's *absolute* position in the sequence [12], or by its *relative* distance to other tokens [34–36].

[4] https://github.com/reichang182/variable-length-piano-infilling
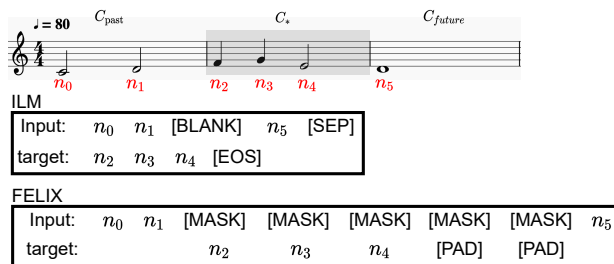
Figure 2: An illustration of how the baselines, ILM [29] and FELIX [30], solve the infilling task. For FELIX, the pre-defined mask length is set to five in this illustration.

2. BACKGROUND

2.1 Related Work on Music Score Infilling

Existing work can be divided by their data representation:

DeepBach-like. DeepBach [1] predicts a missing note based on the information from the notes around. They use two RNNs to aggregate the past and future contexts and a feedforward neural network for the notes occurring at the same temporal position as the current target note. They use pseudo-Gibbs sampling to improve the generated score gradually. Anticipation-RNN [2] introduces a constraint-RNN to the generation model, enforcing the model to consider the user-defined constraints while generating. Music InpaintNet [3] also uses an RNN to integrate the information within the context. However, they use the encoder of a bar-wise variational auto-encoder (VAE) to encode measures into latent vectors first and use the decoder of the VAE to reconstruct measures from the latent vectors. Music SketchNet [4] extends the work of Music InpaintNet to allow their model to consider user preferences.

DeepBach-like representation represents notes in an elegant way. But, it is mainly suitable for scores with a constant number of voices, e.g., four voices for Bach chorale and one voice for Irish and Scottish folk tunes [37], since representing multiple notes played at the same time requires more tracks added to the representation. Prior arts on DeepBach-like representation all use RNNs to deal with contexts, while we use Transformer-based models, which have been shown more powerful [16, 38].

Piano roll. Coconet [6] trains a convolutional neural network (CNN) to complete partial music score and uses blocked Gibbs sampling as an analogue to rewriting. Nakamura *et al.* [7] use CNN with deconvolutional layers at the end. The whole model is trained under the framework of generative adversarial networks. ES-Net [8] represents music as a sequence of *edit events*, each of which denotes either an addition or removal of a note. ES-Net is able to modify a music score while preventing the accumulation of errors that autoregressive models are prone to have.

The piano roll representation typically encodes information concerning pitch, pitch duration, and beat position only, not tempo and velocity, which are important to form an expressive piece. Moreover, as the CNN treats a piano roll as an fixed-size image, all existing piano roll-based models can only infill spans of fixed-length (e.g., 2 bars)

once the CNN was trained. On the contrary, a single model trained with our methodology can be applied to tasks with missing spans of different length (e.g., 2 bars to 4 bars).[5]

Aside from differences in data representation, some existing methods impose additional assumptions on data. For example, the use of bar-wise VAE in Music InpaintNet and Music SketchNet restrict their usage to cases where the missing segments start and end at precisely the start or end position of a bar, while our model is free of this restriction.

Event tokens. IPP [9] is the only work we are aware of that considers music as a sequence of tokens and employs the Transformer as the backbone model. Our work differs from theirs in two aspects. First, their model can only infill a fixed number of tokens, while ours can do variable-length infilling. Second, while we both group tokens related to a note to a tuple (cf. Section 3.1) [15, 24], our representation is based on the beat-based CP tokens [24], which have built-in notion of bars and beats.

Converting our data to be acceptable to the aforementioned models and making them produce variable-length infilling is not trivial. Hence, in our experiment we adopt text infilling models [29, 30] as the baselines, for they are both able to infill variable-length segment by design, not these music score infilling models.

2.2 Related Work on Text Infilling

ILM [29] replaces the missing segment with a single special token [BLANK]. The model learns to predict the original contents at the end of the sequence in an autoregressive manner. ILM only changes the input order of tokens and does not modify the attention mechanism of the vanilla Transformer. **FELIX** [30] uses BERT to derive the capacity of utilizing bidirectional contexts. For variable-length infilling, the missing segment is replaced with a series of [MASK] tokens of a pre-defined length that is sufficiently long. The model learns to predict the tokens to infill and [PAD] tokens to indicate no token here. FELIX is different from our model and ILM in that it predicts all tokens in the missing segment at once, and does not consider previously generated tokens during generation. Both ILM and FELIX were tested on infilling less than 10 words in the original papers, while we consider up to 128 notes in our task. Figure 2 illustrates how these two baselines work.
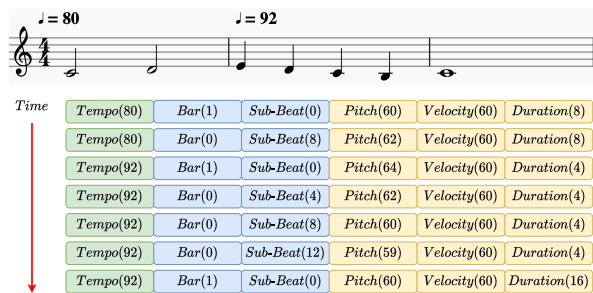
3. METHODOLOGY

We follow the definition of music score infilling in [3]: given a **past context** C_{past} and a **future context** C_{future}, the task is to generate an **infilled segment** C_*, which connects C_{past} and C_{future} in a musically meaningful way. During training, the model should maximize the likelihood:

$$P(C_* | C_{\text{past}}, C_{\text{future}}). \tag{1}$$

We consider in this paper the case where the model is a Transformer encoder and its input is a token sequence com-

Figure 3: An example of a piece of score encoded using our representation. Note that the BAR and SUB-BEAT tokens are for positioning a note event on the time grid [20].

Token type	Voc. size	Values
Tempo	47	28, 32, ..., 212
Bar	2	0, 1
SUB-BEAT	16	0, 1, ..., 15
Pitch	86	22, 23, ..., 107
Velocity	33	0, 4, ..., 128
Duration	16	1, 2, ..., 16

Table 1: The token vocabulary used in our experiments. Note that all the vocabulary sizes do not count special tokens such as <EOS> and <PAD>, since the use of the special tokens are model-dependent.

posing of $\{C_{\text{past}}$, a *masked* version of C_*, $C_{\text{future}}\}$. The target output is the sequence $\{C_{\text{past}}, C_*, C_{\text{future}}\}$. Model loss is computed over only the middle part related to C_*.

Moreover, we desire our model not to generate C_* all at once, but one token at a time in an autoregressive manner. This way, the model builds C_* progressively by considering its previously generated tokens. The training objective can be factorized to

$$\prod_{0 < i \leq T} P(n_i | C_{\text{past}}, C_{\text{future}}, n_{j, \, 0 < j < i}), \tag{2}$$

where $n_1, ..., n_T$ are the tokens in C_*. Please note that, in our setting, the number of tokens T is variable, so does the number of tokens within C_{past} and C_{future}, respectively.

3.1 Compound Word-based Token Representation

We modify the beat-based token representation REMI [20] and CP [24] to encode our music data. As illustrated in Figure 3,[6] we describe different attributes of a musical note through six different tokens—three note-related ones, PITCH, DURATION, and VELOCITY, and three metric-related ones, TEMPO, BAR, and SUB-BEAT.[7] Table 1 shows the vocabulary of the adopted token representation. BAR is encoded to 1 for encountering a new bar, while encoded to 0 for staying at the same bar. SUB-BEAT is the position of a note within one bar, represented in a resolution of the 16-th note. Following the *multi-output* method-

---

[5] Moreover, both DeepBach-like and piano roll require a token to hold at each position; e.g., 100 tokens are needed to represent one second of piano performance at a temporal resolution of 10ms, regardless of how many note events there are [15].

[6] For simplicity, we use monophonic pieces as examples in all the figures in the paper, though we actually use polyphonic pieces in experiment.

[7] Even though it is not that reasonable to associate a TEMPO event with each note, we found that the models learn to predict similar tempos for adjacent notes, thus barely lower the quality of music.
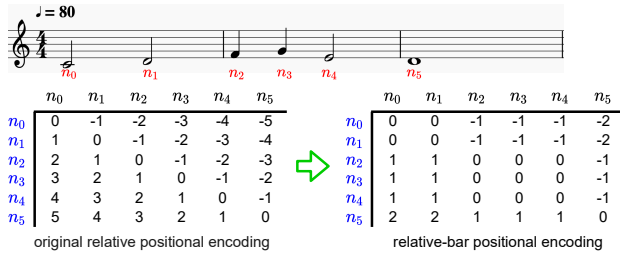
Figure 4: (Left) the widely-used relative positional encoding vs. (right) the proposed relative-bar positional encoding. Each row shows the positional encodings of a token.

Figure 5: Look-ahead onset prediction with (a) the XLNet (those shaded are masked tokens) and (b) a Transformer decoder; the input tokens in (b) are unsynchronized.

ology of the CP Transformer [24], we may consider the six tokens defining a note as a group and predict them altogether *at once* at each timestep. Accordingly, the n_i in Eq. (2) is actually a "super token" (also referred to as a "note" or a CP token herefater) comprising six tokens. When a CP token is masked, all its six constituent tokens are masked.

Below, we also refer to BAR and SUB-BEAT as the *onset* of a note, and the other four tokens as the note's *content*.

3.2 Learning Bidirectional Contexts through XLNet

We adapt XLNet [31] to predict the masked (missing) tokens of C_*. Unlike other bidirectional models such as BERT [13], XLNet can effectively address Eq. (2) due to its special *two-stream self-attention* mechanism.

Models such as BERT cannot address Eq. (2) because the missing parts of the input (i.e., n_j in Eq. (2)) is replaced with a series of masked (CP-)tokens and those parts cannot be seen by the notes to be predicted after (i.e., n_i, $j < i$). The idea of two-stream self-attention is to separate the input into two streams—the *content stream* and the *query stream*. Each masked token n_i is inferred through the query stream, which masks the content of the target token at the timestep i. But, in inferring n_i, we can attend to the content of other tokens n_j that are before n_i through the content stream, which does not mask any tokens.

The original XLNet model is general and does not requires the masked tokens to be consecutive as the case considered in Eq. (1). It covers music infilling as a special case with the following specific permutation order in its *permutation language modeling*: $C_{\text{past}} \rightarrow C_{\text{future}} \rightarrow C_*$.

3.3 A New Positional Encoding

The adapted XLNet considers both C_{past} and C_{future} and does well in fixed-length infilling, i.e., for scenarios where the number of tokens in C_* is known or pre-defined. However, to extend the model to variable-length infilling, the vanilla positional encodings [34] employed in the original XLNet (and most Transformer-based models) to realize the sequential order of the tokens become a problem. *Without knowing the number of tokens in C_*, we cannot assign proper positional encoding to the notes in C_{future}.* [8]

To address this issue, we propose a novel *relative bar encoding* to replace the original vanilla relative positional
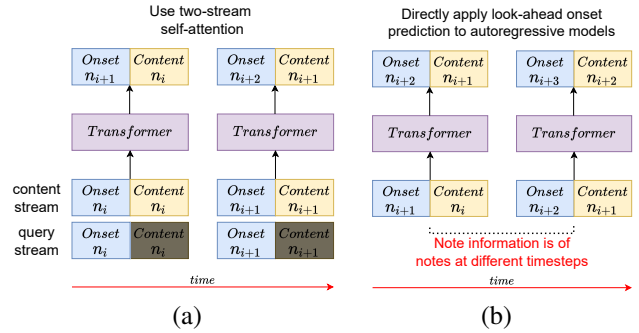
embedding [34] adopted by XLNet. While the original positional encoding represents the relative distance between two notes in terms of the number of intermediate notes, the proposed method represents the distance by the number of bars in between. For instance, tokens within the same bar are 0 bar apart, and thus get 0 for the relative-bar positional encoding. However, for notes in the next bar, the current note is one bar before, and thus will get -1 for the relative-bar positional encoding. In this way, we only need to know how many bars C_{past} and C_{future} are apart to assign the relative bar positional encoding to their and C_*'s notes. [9] See Figure 4 for an illustration.

3.4 Look-ahead Onset Prediction through XLNet

While Section 3.1 suggests that we predict the six tokens of a note n_i at the same timestep, this is actually not ideal when it comes to exploiting the relative-bar positional encoding. The problem is that we need to know the onset of n_i beforehand to assign a proper relative-bar positional encoding to its corresponding input (i.e., a masked CP token) to the Transformer. To this end, we propose *look-ahead onset prediction*, where the onset of a note n_i is inferred one timestep ahead. Specifically, we modify the XLNet such that *the onset of n_i* (i.e., BAR or SUB-BEAT) is predicted along with *the content of n_{i-1}* (i.e., four content-related tokens) at timestep $i - 1$. The onset of n_i is then fed to the model at timestep i as the query stream input, with the content part of the query stream input masked, to infer the content of n_i. This is illustrated in Figure 5(a).

There are two design details. First, the onset of the first note to be infilled should be provided by the user during inference phase, which may be desirable as the user can decide where to start infilling. Second, the onset of the next note also serves as a stop signal; once the model predicts a special [EOS] token for either BAR or SUB-BEAT of the next note, the infilling process comes to an end.

The overall architecture is shown in Figure 6. The query stream inputs are the same as the content stream input ex-

---

[8] Specifically, while we know the length of C_* at training time, we do not know its length at inference time.

[9] Moreover, relative bar encoding actually provides more musically meaningful information to models than the original positional encoding does. For example, five notes being played at the same time are considered as four notes apart for the farthest two notes by the model with the original positional encoding. However, they are actually notes with the same onset time in a score.
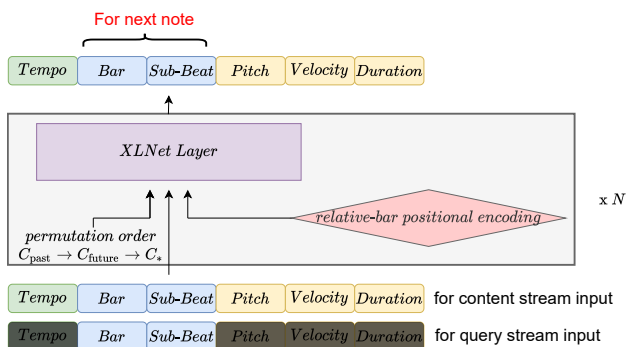
Figure 6: An illustration of the overall architecture proposed in this paper. (1) A specific permutation order of sequence is given to the model. (2) Tokens except for BAR and SUB-BEAT are masked to form the query stream input of XLNet. (3) Relative-bar positional encodings are used instead. (4) BAR and SUB-BEAT from the output of XLNet are the musical position of the next note.

cept that the content of notes are replaced by mask tokens, since those are the parts to be inferred. The onset part of the note (i.e., BAR and SUB-BEAT) is made visible, to allow the model to exploit the onset information. We note that the BAR token fed as input to the query stream only tells the model whether the note is in a new bar or stays in the same bar, but not how many bars apart the current note to the other notes in C_{past}, C_{future} and the rest of C_*. Therefore, the relative-bar positional encoding is still needed.

3.5 The Necessity of Two-Stream Self-Attention

We are now ready to elaborate more why we are in favor of XLNet instead of a Transformer decoder such as the one used by IPP [9]. As depicted in Figure 5(b) and exemplified in Figure 7, to realize the look-ahead onset prediction needed by the proposed relative bar encoding, the onset-related tokens and content-related tokens in the input to a Transformer decoder would be *unsynchronized*. For instance, the onset of the first input is for n_{i+1}, yet the content is for n_i. Such a mismatch impedes the Transformer decoder to attend to proper notes through the dot product of the input embeddings. This is not a problem for XLNet, since the input tokens in either the content stream or query stream remain *synchronized* (e.g., both for n_i).

4. EXPERIMENTAL SETUP

Dataset. We use the AILabs-Pop1k7 dataset shared publicly by Hsiao *et al.* [24], which contains 1,748 MIDI files of polyphonic pop piano performances, all in 4/4 time signature. We further quantize the tempo, beat position, duration, and velocity to reduce the vocabulary size, setting the 16-th note as our minimal temporal resolution for beat position and duration. There are on average 12.6 notes per bar. We crop the music into 16-bar pieces with an 8-bar overlap between successive pieces, yielding in total 19,789 16-bar data for model training.

Detailed Settings. We train all the models on 16-bar data with up to 512 CP tokens and design the experiment
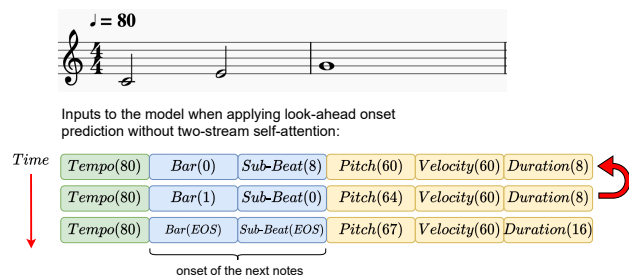
Figure 7: Illustration of the problem of a Transformer decoder such as IPP [9] for realizing look-ahead onset prediction. To infer the third note in the sequence, it may be beneficial if the model can attend to the first note, since both notes are at the same sub-beat position (though in different bars). However, because the onset-related and content-related tokens are unsynchronized (cf. Figure 5(b)), the model may not be able to attend to the proper notes.

with the following conditions in mind. First, C_{past} and C_{future} must be long enough to provide sufficient contextual information. Second, the length and the musical position of C_* should not be fixed, since we do not know where the missing segment starts and how many notes should be infilled for various cases in the real world. Consequently, for each token sequence, a range within the middle four bars, i.e., bar7 to bar10, is randomly selected to be C_*, such that C_{past} and C_{future} are at least 6 bars long, respectively. Note C_* is not required to start right at the beginning of bar7. The minimum number of CP tokens to be infilled is set to half the number of CP tokens within bar7 and bar10. [10]

A CP token is transformed before being fed to models. First, each of the six tokens composing the CP token is mapped to an embedding with size 256 using a lookup table. Then, these embeddings are concatenated and merged through a linear layer not shared across models, producing a merged embedding of size 768. All the models accept these merged embeddings as input and each has 8 heads, 12 self-attention layers, and intermediate layers of dimension 3,072. The output from these models is transformed back to probabilities with linear projection followed by softmax. At inference time, the tokens are sampled through *nucleus* [39] with temperature 1.0 and threshold 0.9.

5. EXPERIMENTAL RESULTS

5.1 Objective Evaluation

We evaluate these models with a number of metrics proposed in [22], which are *pitch class histogram entropy* and *grooving pattern similarity*. The former provides us an indicator to the usage distribution of each pitch class within 1 bar and 4 bars, resulting in metrics $\mathcal{H}_1$ and $\mathcal{H}_4$. The latter evaluates the rhythmic pattern similarity between bars ($\mathcal{GS}$). Since the goal of the task is to connect contexts and generate fluent music, these metrics calculated on C_*

---

[10] And, note that we do not expect the models to rely on the absolute musical position within an entire music piece to inference, but should only rely on the note's contexts around. Thus, when providing the 16-bar data to the models, the absolute position of these 16-bar data within an entire music piece is discarded.
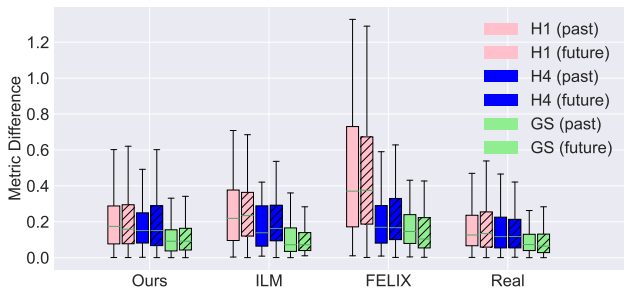
Figure 8: The difference between "C_* and C_{past}" and between "C_* and C_{future}" in 3 objective metrics: $\mathcal{H}_1, \mathcal{H}_4, \mathcal{GS}$.

should be close to those calculated on C_{past} and C_{future}. Thus we calculate the difference of metrics between C_* and C_{past}, and also between C_* and C_{future}. The lower the values are, the better the model is. Since these are bar-wise metrics, we let the models generate all four middle bars, i.e., bar7 to bar10, instead of a portion of them. From Figure 8, it seems that our model and ILM learn well to generate C_* coherent to its contexts. The results are even close to that of the real data. In contrast, FELIX performs poorly in $\mathcal{H}_1$. It is possibly due to the dependency problems between the masked tokens, since each token is unaware of what other tokens predict, leading to uncertain tonality.

We also adapt the *MIREX-like prediction test* [22,40,41] for testing the models' ability to infer the correct answer from contexts. The test includes 1,000 questions generated from a held-out validation set, with each question comprises 6-bar C_{past} and C_{future}. The goal of this test is to select the right infilling from four choices. A model makes a choice by calculating the average probability of the first few notes belonging to each choice, and selects the one with the highest probability. The number of notes to average from is dependent on the shortest length of all the choices. In addition, the choices could be randomly selected from a *different* song, or somewhere from the *same* song. The latter is harder since the choices are much more similar. We name them the **simple test** (choices come from different music) and the **hard test** (from the same music). The results in Table 2 show that our model outperforms the other two consistently in both tests. It should be noted that stability is an important factor for autoregressive models in this test, since a wrongly predicted note leads to the accumulation of errors in the following prediction. While both our model and ILM could suffer from such a problem, our model still performs slightly better.

5.2 Subjective Evaluation

A user study is conducted with in total 30 subjects, where 7 of them are deemed as professionals according to a question about their musical background. Each subject is presented with 3 sets of music randomly selected from the total 15 sets of music. Within each set, a subject listens to a music piece with a missing segment, and is then presented with 4 music pieces, where 3 of them are generated by the models (Ours, ILM, and FELIX) and 1 of them is the real music without the missing segment. The subject is

	VLI (ours)	ILM [29]	FELIX [30]
simple test	**0.940**[*]	0.796	0.919
hard test	**0.467**[**]	0.361	0.398

[**]: leads all others with $p < .01$; [*]: with $p < .05$

Table 2: Accuracy in the MIREX-like prediction test. **VLI** denotes the proposed variable-length piano infilling model.

		M	R	I	F
all	VLI (ours)	**3.27**	**3.43**	**2.93**	**27%**
	ILM [29]	2.63	2.63	2.67	17%
	FELIX [30]	2.83	2.77	2.57	16%
	Real	**3.83**	**3.73**	**2.83**	**41%**
pro	VLI (ours)	**3.08**	**3.38**	**2.77**	**24%**
	ILM [29]	2.69	2.77	2.62	10%
	FELIX [30]	2.85	2.62	2.62	14%
	Real	**3.77**	**3.92**	**2.92**	**52%**

Table 3: Results of the user study: mean opinion scores in 1–5 in **M** (melodic fluency), **R** (rhythmic fluency), **I** (impression), and percentage of votes in **F** (favorite), from 'all' the participants or only the music 'pro'-fessionals.

asked to rate each of the 4 music pieces according to its 1) **melodic fluency**: how many wrong notes are there? The fewer wrong notes, the higher the score; 2) **rhythmic fluency**: Are there notes played at the wrong time? The less, the higher the score; 3) **impression**: how much the subject is impressed? The more, the higher the score. After listening to the 4 music pieces, they are asked to choose a **favorite** one from them. From the results in Table 3, our model does beat the other baselines by a large margin. Our model even has a higher impression score than the real music when considering all subjects. However, there is still a gap between ours and the real data in the "favorite" score, suggesting that the music generated by our model is still differentiable from the real music.

6. CONCLUSION & DISCUSSION

In this paper, we have proposed a new model adapted from XLNet to address music score infilling. Specifically, to make the model able to perform variable-length infilling, we replace the token-based distance attention mechanism in Transformers with a musically specialized one considering relative bar distance. We have also reported evaluations showing that our model outperforms two strong baselines.

We do not pay attention to whether the past and future contexts are similar in style or theme in this work. It would be interesting to see in a future work whether our model can infill a nice transition between two dissimilar segments. Moreover, by changing the permutation order, our model can be applied to sequential generation in the future, to study whether the relative bar encoding improves the metrical structure of the generated music.

7. ACKNOWLEDGEMENT

8. REFERENCES

[1] G. Hadjeres, F. Pachet, and F. Nielsen, "DeepBach: a steerable model for Bach chorales generation," in *Proc. International Conference on Machine Learning (ICML)*, 2017, pp. 1362–1371.

[2] G. Hadjeres and F. Nielsen, "Anticipation-RNN: Enforcing unary constraints in sequence generation, with application to interactive music generation," *Neural Computing and Applications*, 2018.

[3] A. Pati, A. Lerch, and G. Hadjeres, "Learning to traverse latent spaces for musical score inpainting," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[4] K. Chen, C.-I. Wang, T. Berg-Kirkpatrick, and S. Dubnov, "Music SketchNet: Controllable music generation via factorized representations of pitch and rhythm," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[5] T. Bazin and G. Hadjeres, "NONOTO: A model-agnostic web interface for interactive music composition by inpainting," in *Proc. International Conference on Computational Creativity (ICCC)*, 2019.

[6] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, "Counterpoint by convolution," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[7] K. Nakamura, T. Nose, Y. Chiba, and A. Ito, "A symbol-level melody completion based on a convolutional neural network with generative adversarial learning," *Journal of Information Processing*, vol. 28, pp. 248–257, 2020.

[8] W. Chi, P. Kumar, S. Yaddanapudi, R. Suresh, and U. Isik, "Generating music with a self-correcting non-chronological autoregressive model," *ArXiv*, vol. abs/2008.08927, 2020.

[9] D. Ippolito, A. Huang, C. Hawthorne, and D. Eck, "Infilling piano performances," in *Proc. NIPS Workshop on Machine Learning for Creativity and Design*, 2018.

[10] A. Adler, V. Emiya, M. Jafari, M. Elad, R. Gribonval, and M. Plumbley, "Audio inpainting," *IEEE Transactions on Audio Speech and Language Processing*, vol. 20, pp. 922 – 932, 03 2012.

[11] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This time with feeling: Learning expressive musical performance," in *Neural Computing and Applications*, vol. abs/1808.03715, 2018.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conference on North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019, pp. 4171–4186.

[14] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang, S. Watanabe, T. Yoshimura, and W. Zhang, "A comparative study on Transformer vs RNN in speech applications," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, 2019.

[15] C. Hawthorne, A. Huang, D. Ippolito, and D. Eck, "Transformer-NADE for piano performances," in *Proc. NeurIPS Workshop on Machine Learning for Creativity and Design*, 2018.

[16] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. M. Shazeer, A. M. Dai, M. Hoffman, M. Dinculescu, and D. Eck, "Music Transformer: Generating music with long-term structure," in *Proc. International Conference on Learning Representations (ICLR)*, 2019.

[17] C. M. Payne, "MuseNet," *OpenAI Blog*, 2019.

[18] C. Donahue, H. H. Mao, Y. E. Li, G. Cottrell, and J. McAuley, "LakhNES: Improving multi-instrumental music generation with cross-domain pre-training," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 685–692.

[19] K. Choi, C. Hawthorne, I. Simon, M. Dinculescu, and J. Engel, "Encoding musical style with transformer autoencoders," in *Proc. International Conference on Machine Learning (ICML)*, 2020, pp. 1899–1908.

[20] Y.-S. Huang and Y.-H. Yang, "Pop Music Transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proc. ACM Multimedia*, 2020, pp. 1180—-1188.

[21] Y.-H. Chen, Y.-H. Huang, W.-Y. Hsiao, and Y.-H. Yang, "Automatic composition of guitar tabs by Transformers and groove modeling," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[22] S.-L. Wu and Y.-H. Yang, "The Jazz Transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[23] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, "PopMAG: Pop music accompaniment generation," in *Proc. ACM Multimedia*, 2020, pp. 1198—-1206.

[24] 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," in *Proc. AAAI Conference on Artificial Intelligence*, 2021.

[25] W. Fedus, I. Goodfellow, and A. Dai, "MaskGAN: Better text generation via filling in the_____," in *Proc. International Conference on Learning Representations (ICLR)*, 2018.

[26] W. Zhu, Z. Hu, and E. P. Xing, "Text infilling," *arXiv preprint arXiv:1901.00158*, 2019.

[27] T. Shen, V. Quach, R. Barzilay, and T. S. Jaakkola, "Blank language models," in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 5186–5198.

[28] Y. Huang, Y. Zhang, O. Elachqar, and Y. Cheng, "INSET: Sentence infilling with INter-SEntential Transformer," in *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020, pp. 2502–2515.

[29] C. Donahue, M. Lee, and P. Liang, "Enabling language models to fill in the blanks," in *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020, pp. 2492–2501.

[30] J. Mallinson, A. Severyn, E. Malmi, and G. Garrido, "FELIX: Flexible text editing through tagging and insertion," in *Proc. Findings of the Association for Computational Linguistics*, 2020, pp. 1244–1255.

[31] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[32] G. Ke, D. He, and T.-Y. Liu, "Rethinking the positional encoding in language pre-training," in *Proc. International Conference on Learning Representations (ICLR)*, 2021.

[33] B. Wang, L. Shang, C. Lioma, X. Jiang, H. Yang, Q. Liu, and J. G. Simonsen, "On position embeddings in BERT," in *Proc. International Conference on Learning Representations (ICLR)*, 2021.

[34] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019, pp. 2978–2988.

[35] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, 2018, pp. 464–468.

[36] A. Liutkus, O. Cífka, S.-L. Wu, U. Şimşekli, Y.-H. Yang, and G. Richard, "Relative positional encoding for Transformers with linear complexity," in *Proc. International Conference on Machine Learning (ICML)*, 2021.

[37] B. L. Sturm, J. F. Santos, O. Ben-Tal, and I. Korshunova, "Music transcription modelling and composition using deep learning," in *Proc. Conference on Computer Simulation of Musical Creativity*, 2016.

[38] S.-L. Wu and Y.-H. Yang, "MuseMorphose: Full-song and fine-grained music style transfer with just one Transformer VAE," *arXiv preprint arXiv:2105.04090*, 2021.

[39] A. Holtzman, J. Buys, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *Proc. International Conference on Learning Representations (ICLR)*, 2020.

[40] "The "Patterns for Prediction Challenge" of Music Information Retrieval Evaluation eXchange," [Online] https://www.music-ir.org/mirex/wiki/2019: Patterns_for_Prediction.

[41] B. Janssen, T. Collins, and I. Ren, "Algorithmic ability to predict the musical future: Datasets and evaluation," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 208–215.