CS230

# Diffusion-LM on Symbolic Music Generation with Controllability

**Hao Sun**
Stanford University
haosun@stanford.edu

**Liwen Ouyang**
Stanford University
ouyang42@stanford.edu

## 1 Novelty

Providing controllability to generative tasks has been a long-standing problem. In a symbolic music generation task, controllable generation effectively aligns with the artist's expectations via different indicators, such as instrument, genre, or melody trend, which could be helpful in real-world tasks, including film score composition and accompaniment generation. Thus, we suggest an end-to-end symbolic music generation model with plug-and-play controllability. The most outstanding novelties of our project are utilizing the latent diffusion model on symbolic music generation using a modified Transformer[21] with time embedding and training classifiers as plug-and-play classifiers.

## 2 Introduction

Continuous diffusion model[8][1][19][12] is effective in vision and audio domains and has been recently adapted to the text generation task[13]. We use a similar approach by Diffusion-LM in our end-to-end model to achieve similar distribution of outputs in the generation tasks; by training and providing a classifier to guide denoising steps, we can achieve controllability on the output. In detail, we use a dedicated tokenizer and "Bar-Block" padding method to parse symbolic notes into fixed-length sequences. Through a customized Transformer[21] model, we predict the latent variables yielded in each step of the denoising process and round the output to note tokens by the K-NN algorithm, thus, generating output midi files. Due to the limited scale of this project, we emphasize the controllable generation of single MIDI tracks.

You can find our model's code implementations on `https://github.com/SwordElucidator/Diffusion-LM-on-Symbolic-Music-Generation`.

## 3 Relate Work

### 3.1 Diffusion Models for Discrete Data

Diffusion-LM[13] adapted continuous diffusion to discrete data generation by using the Transformer[21] model and achieved structural and semantic controls by iteratively performing gradient updates on the continuous latent variables. We use a similar method for discrete tokenized midi data.

### 3.2 Diffusion Models for Symbolic Music Generation

Diffusion model In the research paper "SYMBOLIC MUSIC GENERATION WITH DIFFUSION MODELS" [14], the author considered MusicVAE [18] to build continuous latent variables, linking to the diffusion procedure in DDPM [8] and received excellent results. While their attempts used

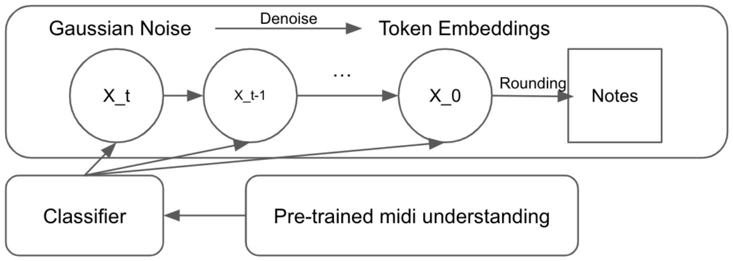

Figure 1: General Architecture of the Model

VAE to predict continuous latent variables, we hope to use a transformer to predict continuous latent variables so that we could use a classifier sharing the same embeddings to guide the latent variable.

### 3.3 Other Symbolic Music Generation Methods

Traditionally, RNN and GAN structured music generators such as MelodyRNN[22] and MuseGAN[4] were popular. There were also many Transformer based generation models such as Music Transformer[9], OpenAI's MuseNet[16], and PopMAG[17] focusing on musical compositions.

## 4 Dataset and Features

Due to the project's size, we concentrate on single-track music data. We tried GiantMIDI-Piano[11], a classical piano MIDI dataset containing 10,855 MIDI files of 2,786 composers. Since it is challenging to learn the complexity of classic piano tunes, we also use another mono-midi-transposition-dataset[6], a more manageable midi dataset for monophonic music containing 15,972 shorter MIDI files from different instruments. We separate them into 9552/2400/3980 for training/validation/evaluation.

## 5 Methods

A general architecture of the model is shown in fig1:

### 5.1 Tokenizer, Padding and Embedding

We used MidiTok [5], a popular Python package to tokenize MIDI music files into sequences of tokens. We experimented with several different tokenizers including Midi-like[15], REMI[10], and Structured[7] in the early stages. Specifically, we will use the REMI [10] tokenizer, in which the time is represented with "Bar" and "Position" tokens that indicate respectively when a new bar is beginning and the current position within a bar, and a note is represented as a succession of a Pitch, Velocity and Duration tokens.

With limited computation resources, we limited the sequence to 256 in the diffusion steps. We invented a new padding method called "Bar-Block," which splits a midi file into several blocks with a maximum length of 256 that start with a "Bar" token and end with padding tokens. Those blocks with more than 20% of the padding tokens will be omitted. Besides, the first few continuous "Bar" tokens in each block will also be removed to reduce empty bars in the output sequences.

Based on the Diffusion-LM experience [13], We used 16-dim and 32-dim random embeddings to convert tokenized notes to embeddings, and a sequence of notes will be defined as $EMB(notes) = [EMB(note_1), ..., EMB(note_n)] \in \mathbb{R}^{dim}$.

### 5.2 Diffusion Model

A diffusion model[8] is a latent variable model inspired by non-equilibrium thermodynamics, which defines a Markov chain $x_T...x_1$ of diffusion steps with $x_T$ a Gaussian and $x_0 \in \mathbb{R}^d$. The diffusion model gradually denoises the sequence of latent variables $x_{T:1}$ to approximate samples from the

| Encoder Structure | Dataset | Tokenizer | Embedding Dim | Eval Loss (MSE) |
|---|---|---|---|---|
| Bert | GiantMIDI-Piano | Midi-Like | 16 | 0.326 |
| Bert | GiantMIDI-Piano | Midi-Like | 32 | 0.1813 |
| Bert | GiantMIDI-Piano | REMI | 32 | 0.1373 |
| Bert | mono | Structured | 32 | 0.07542 |
| Bert | mono | REMI | 32 | 0.05387 |
| Music Transformer | mono | REMI | 32 | 0.2127 |

Table 1: Generation Task Metrics on 2800 steps

target data distribution. Each denoising transition $x_t \to x_{t-1}$ is parametrized by a Transformer. As derived by Diffusion-LM [13], in the generation task, we use mean-squared error loss

$$L_{simple}(x_0) = \sum_{t=1}^{T} \underset{q(x_t|x_0)}{\mathbb{E}} \|\mu_\theta(x_t, t) - \hat{\mu}(x_t, x_0)\|^2$$

where $\hat{\mu}(x_t, x_0)$ is the mean of the posterior $q(x_{t-1}|x_0, x_t)$ and $\mu_\theta(x_t, t)$ is each latent variable predicted computed by Transformer. The MSE loss is calculated on each single time step and perform gradient updates on the Transformer parameters.

### 5.3 Time-Aware Transformer

We use a modified version of Transformer[21] inspired by Diffusion-LM[13] to predict the latent variables in each denoising step. To prevent GPU resources from being exhausted by introducing too large parameters in the diffusion process, we used 32-dim embeddings in our experiments. However, to ensure that the Transformer is still large enough for self-attention and dense layers, we created a learnable up-projection layer and a down-projection layer after the encoder to project the embedding dimension to 768 and vice versa. Inspired by Diffusion-LM[13], we also introduced a sinusoidal time embedding that indicates the denoising step in the embedding process. We experimented three encoder structures from Bert[3], Music Transformer[9], and Longformer[2].

### 5.4 Time-Aware Classifier

A classifier is used to achieve the desired controllability of the diffusion model's generated output. In each denoising step, the diffusion model needs to query the classifier, do the backpropagation, and quantitatively update the predicted latent variables by the Langevin function. We tried different classifier structures in our experiments, including fully connected neural network, LSTM, Transformer, and Music Transformer[9] on instrument and music composition labels. Because we were not satisfied with the experimental results, we also pre-trained a modified BERT[3] by Masked-LM with more than 300,000 midi blocks of length 256 obtained by "Bar-Block" padding on the GiantMIDI-Piano[11] dataset to achieve better midi understanding. This pre-trained model also has an up projection layer and time embeddings but has no down projection layer since it was then connected to a fully connected layer for classification.

To add time-aware ability, we integrated the Gaussian diffusion to the classifier model to generate random noises on different time steps and added those noises to our training and validation samples during training. This method also has some data augmentation effect.

## 6 Experiments

### 6.1 Denoising Steps and Generation

We experimented on two datasets as shown in table1: GiantMIDI-Piano[11] and mono-midi-transposition-dataset[6], and three tokenizers Midi-like[15], REMI[10], and Structured[7], with embedding dim = [16, 32]. Limited to the GPU provided on Colab and AWS, we could only experiment with a token length of 256. The Gaussian diffusion uses step $T = 2000$, and we used Bert Encoder[3] in our model. We found that the REMI[10] out-performs Midi-like[15] and Structured[7] on average loss and output quality as in fig2.

| Structure | Task | Time Aware | Accuracy (eval) |
|---|---|---|---|
| FC | Instrument | no | 78% |
| Shallow Bert (hidden size=32) | Instrument | yes | 59% |
| Shallow Bert (hidden size=32) | Composition | yes | 51% |
| Bert (Transfer Learning with hidden size=768) | Composition | yes | 42% |
| Bert (Pre-train + Fine-tune with hidden size=768) | Composition | yes | 27% |
| Music Transformer + FC | Composer | no | 23% |

Table 2: Classifier Metrics

For the encoders, we tried Bert Encoder[3], Music Transformer[9] and Longformer[2] in our modified Transformer Model. We found that by hidden layer size = 768, intermediate layer size = 3072, 12 encoder layers, and 12 self-attention heads, the Bert encoder outperforms the Music Transformer encoder on 256-token sequences as in fig4. Unfortunately, our modified Longformer encoder was too large to be trained on the provided AWS GPU. Thus, we used Bert Encoder in our further experiments. We also found that the learned embeddings are meaningful by PCA, as shown in fig3.

### 6.2 Basic Controllability

We experimented with two easy controllability mechanisms: length control and infilling without introducing classifier models, although they can be used together or with other classifiers. By providing a sequence with desired note tokens on specific positions (masked on another input sequence) and leaving all other undetermined notes as -1, the output sequence can achieve harmony on provided and generated notes on the chord, velocity, and pitch. Similarly, by setting pad tokens to positions greater than the maximum length, or the [EOS] note to the position that the sequence should exactly end, we also achieved length control.

We did a classic LM-style experiment that asked the model to generate samples starting with [E4, E4, E4, D4, C4] and no longer than 230 tokens. We found that the model successfully generated samples with creativity on how to deal with those initial notes.

### 6.3 Classifier

We experimented with different classifier architectures as shown in table2 to classify instruments on mono-midi-transposition-dataset[6] and music composition types on GiantMIDI-Piano[11] dataset. The fully connected classifier was trained on the mono-midi-transposition-dataset[6] with Adam optimizer, a batch size of 16, and a learning rate of 7e-4. The classifier achieves a 78% accuracy on the test set on the 128 instrument classes. We found a significant label imbalance in these 128 classes, so we narrow the labels down to 34 classes with at least 1000 samples. Similarly, we only chose {Sonata, Variations, Prelude, Valse, Waltz, Morceaux, Mazurka, Etude, Rondo, Ballade, Fantasia, Christian, Sonatina, Romance, Polka, unknown} as the labels of composition type on piano midi.

We then attempted a modified Transformer with time embeddings and transfer learning using the parameters learned on our diffusion model and connecting it to a fully connected layer to classify the instrument. By this approach, we found the model severely over-fitted on the validation set.

### 6.4 Pre-trained Model and Time-Aware Classifiers

We finally attempted to use a pre-trained model and fine-tune the model on the classification task. Although there are some sophisticated pre-trained models for midi understanding, such as MusicBert[23], we found it used a different tokenizer than ours and was trained on much longer sequences with large embedding dimensions and without time-aware structures. Thus, we defined a new pre-training model with time embeddings, up projections, and noises superimposed on the input note embeddings. We pre-trained the model from scratch on Masked-LM, with 12 hidden layers and 12 heads using more than 300,000 midi blocks of length 256 obtained by "Bar-Block" padding on the GiantMIDI-Piano[11] dataset.

Then, we added a fully connected layer using the last hidden state to predict the composition classification sharing the same parameters learned in the pre-training. Unfortunately, we only achieved accuracy = 27% on the validation set.

### 6.5 Classifier Guided Generation

We used the trained classifiers to do backpropagation and update the samples on each denoising step. The sampling procedure is prolonged and takes more than 15 minutes to create a batch of 32 samples using a Transformer-based classifier. In an experiment, we used a poor classifier to guide sampling with Instrument = Piano (with label = 0) and got bad results. We then used a better classifier to guide Composition = Waltz, Etude, and Fantasia and achieve relatively better results.

## 7 Results / Evaluations

### 7.1 Results

We offer music samples generated by our diffusion model on different tasks that can be listened to. `https://drive.google.com/drive/folders/1M5UF2O8Otml11vf9uEVYXiqVVJZkRABO?usp=sharing`

### 7.2 Generation Tasks

As we are generating 256-note blocks of the midi files due to the lack of computation resources, we suppose it is hard directly compare to the baseline models' performance score. Compared to the outputs of RNN, our outputs successfully maintain harmony on chord, velocity, and pitch on the 256 tokens, while RNN could hardly maintain consistency in several bars. On the other hand, while the Transformer[21] maintains consistency in short output sequences, the Music Transformer[9] out-performs on long sequences as in fig6. Compared to these Transformer architectures, our diffusion model's outputs have a closer distribution to the original distribution. It rarely generates inharmonious samples but sometimes dull samples (e.g., repetitions or existing tunes) than the Transformer models, especially without controlling guidance, but with much more creativity on challenging styles as in fig7.

### 7.3 Controllability

We achieved controllability based on length, infilling, and classifications. To compare our infilling with previous models, we infilled the notes at the beginning of the sequence to simulate a language model. Our outputs more creatively melt the preset notes into the melody than the Transformer models as in fig8. For example, while one output uses the first four notes as one pattern and the last note as the start of another pattern, another output adds notes on the given notes and makes a long melody. We speculate that this is because our Diffusion model only considers preset notes as a known part of the target distribution, while Transformers rely on these preset notes to predict later notes, thus tending to reuse this known pattern and losing creativity.

We did not achieve good performance on the classifier guided generation task. As we have frozen the embeddings from the diffusion model, we suppose it might be sub-optimal for the classifier model. We also find that the classifier-guided generations can work terribly under the guidance of a bad classifier. For example, using an instrument classifier with an accuracy of 42% will cause the output to be irregular and worse than vanilla diffusion model generations without guidance.

## 8 Contributions

We worked separately on finding available datasets. While Hao focused more on the Transformer diffusion model and experimented on Colab and AWS EC2, Liwen focused on structuring and training the classifier model. Hao also implemented and experimented with different Transformer structures, the pre-trained model for classification, and the different controllabilities of the diffusion model. Liwen researched and experimented with different classifiers and tried to utilize MusicBert. We both handled midi data preprocessing, including tokenizing, padding, and embedding methods.

## References

[1] Aditya Ramesh et al. *Hierarchical Text-Conditional Image Generation with CLIP Latents*. 2022.

[2] Iz Beltagy, Matthew E. Peters, and Arman Cohan. "Longformer: The Long-Document Transformer". In: *arXiv:2004.05150* (2020).

[3] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[4] Hao-Wen Dong et al. "MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment". In: *arXiv:1709.06298* (2018).

[5] Nathan Fradet et al. "MidiTok: A Python package for MIDI file tokenization". In: *Extended Abstracts for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference*. 2021.

[6] Sebastian Garcia-Valencia, Alejandro Betancourt, and Juan G. Lalinde-Pulido. "Sequence Generation using Deep Recurrent Networks and Embeddings: A study case in music". In: *ArXiv e-prints* abs/2012.0 (2020). arXiv: `2012.01231 [cs.SD]`. URL: `https://arxiv.org/abs/2012.01231`.

[7] Gaëtan Hadjeres and Léopold Crestel. *The Piano Inpainting Application*. 2021. arXiv: `2107.05944 [cs.SD]`.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models". In: *arXiv preprint arxiv:2006.11239* (2020).

[9] Cheng-Zhi Anna Huang et al. "Music Transformer: Generating Music with Long-Term Structure". In: *arXiv preprint arXiv:1809.04281* (2018).

[10] Huang et al. "Pop Music Transformer: Beat-based modeling and generation of expressive Pop piano compositions". In: *Proceedings of the 28th ACM International Conference on Multimedia*. 2020.

[11] Qiuqiang Kong et al. *GiantMIDI-Piano: A large-scale MIDI Dataset for Classical Piano Music*. arXiv: `2010.07061 [cs.IR]`.

[12] Keon Lee. *DiffSinger*. `https://github.com/keonlee9420/DiffSinger`. 2021.

[13] Xiang Lisa Li et al. "Diffusion-LM Improves Controllable Text Generation". In: *ArXiv* abs/2205.14217 (2022).

[14] Gautam Mittal et al. "Symbolic Music Generation with Diffusion Models". In: *Proceedings of the 22nd International Society for Music Information Retrieval Conference*. 2021. URL: `https://archives.ismir.net/ismir2021/paper/000058.pdf`.

[15] Sageev Oore et al. "This time with feeling: Learning expressive musical performance". In: *Neural Computing and Applications* (2018).

[16] C. Payne. "MuseNet". In: (2019). URL: `https://openai.com/blog/musenet`.

[17] Yi Ren et al. "PopMAG: Pop Music Accompaniment Generation". In: *ArXiv* abs/2008.07703 (2020).

[18] Adam Roberts et al. "A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, July 2018, pp. 4364–4373. URL: `https://proceedings.mlr.press/v80/roberts18a.html`.

[19] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2021. arXiv: `2112.10752 [cs.CV]`.

[20] Ian Simon and Sageev Oore. *Performance RNN: Generating Music with Expressive Timing and Dynamics*. `https://magenta.tensorflow.org/performance-rnn`. Blog. 2017.

[21] Ashish Vaswani et al. *Attention Is All You Need*. arXiv: `1706.03762 [cs.CL]`.

[22] Elliot Waite. "Generating Long-Term Structure in Songs and Stories". In: (2016). URL: `https://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn`.

[23] Mingliang Zeng et al. "MusicBERT: Symbolic Music Understanding with Large-Scale Pre-Training". In: *arXiv:2106.05630* (2021).

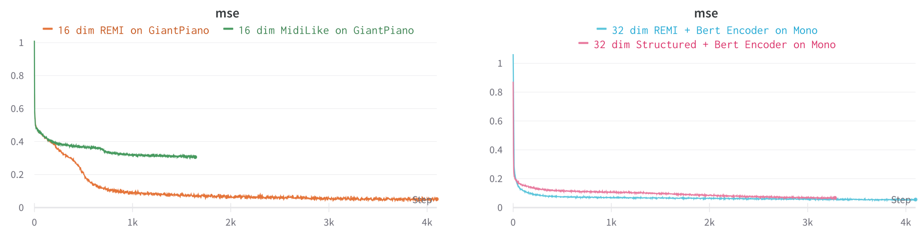

Figure 2: Comparing Midi-like[15], Structured[7], and REMI[10]

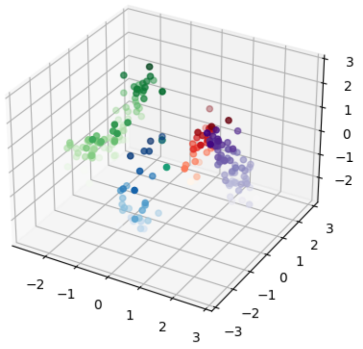

Figure 3: PCA on embeddings learned by the diffusion model: greens, reds, purples, blues indicate Pitch, Position, Duration and Velocity tokens

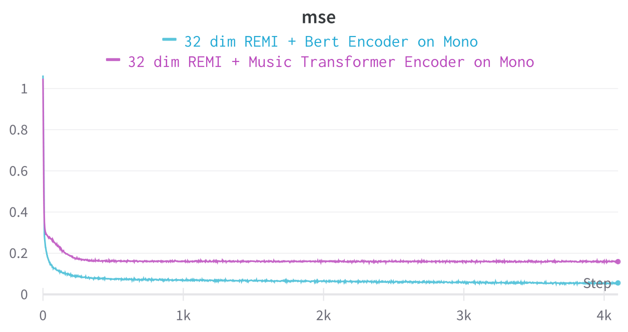

Figure 4: BERT encoder[3] VS Music Transformer encoder[9]

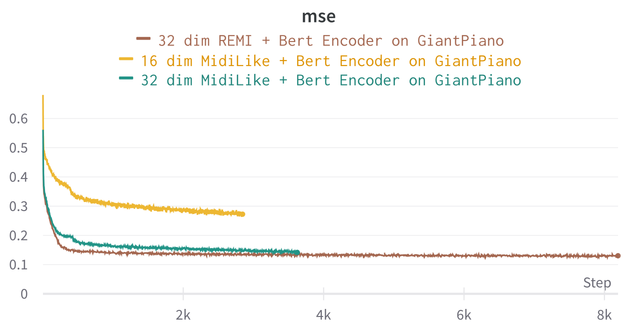

Figure 5: Experiments on GiantPiano[11] dataset

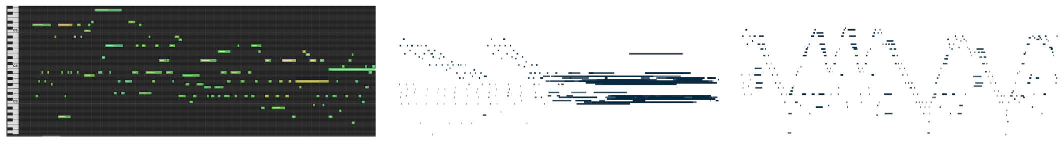

Figure 6: Samples generated by Performance RNN[20], Transformer[21] and Music Transformer[9]

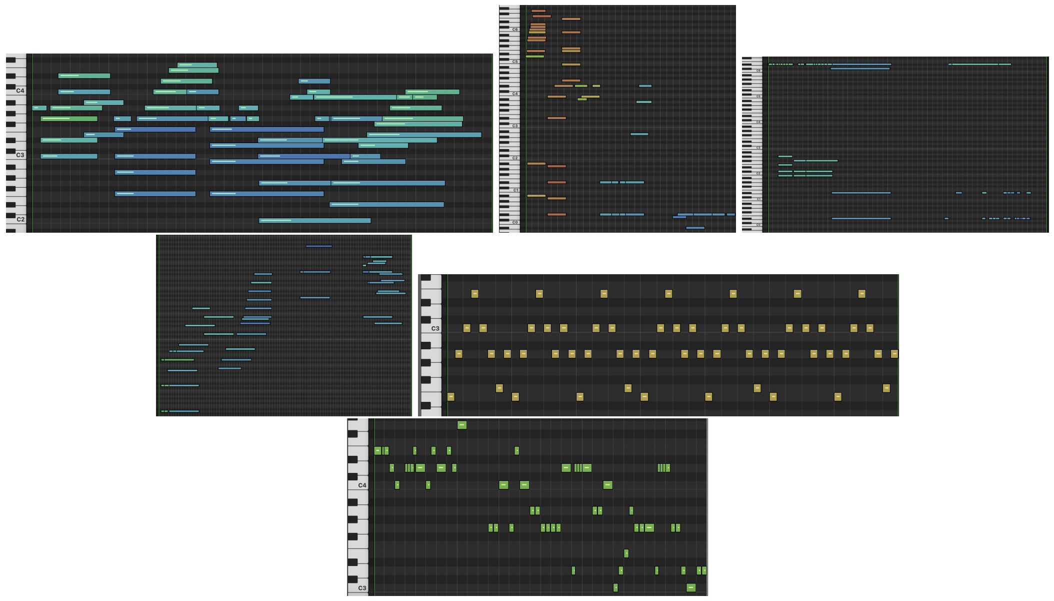

Figure 7: Good samples generated by our diffusion model. Note that the composition style varies with more creativity than the previous models

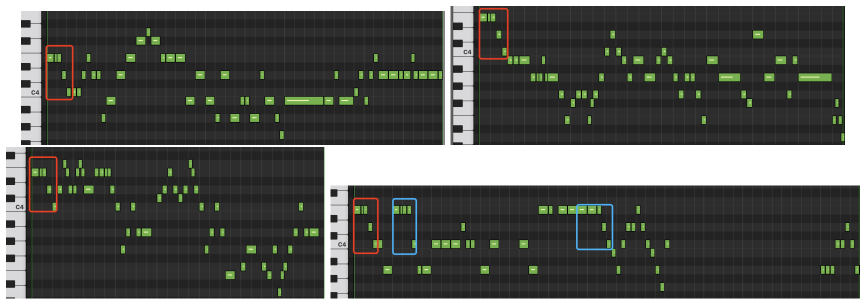

Figure 8: Different Infills starts with [E4, E4, E4, D4, C4]

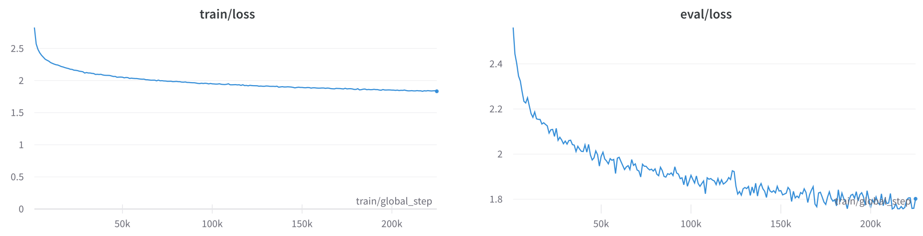

Figure 9: Pre-training on Time Aware Transformer