EVALUATING TOKENIZATION STRATEGIES FOR EXPRESSIVE CLASSICAL PIANO PERFORMANCE GENERATION

Qingyang Lyu *

The University of Hong Kong
Hong Kong, China
u3612707@connect.hku.hk

Brian Cruz

University of California, Berkeley
Berkeley, CA, USA
brian.cruz@berkeley.edu

Jeremy Wagner

University of California, Berkeley
Berkeley, CA, USA
jeremy.wagner@berkeley.edu

*Carmine-Emanuele Cella**

University of California, Berkeley
Berkeley, CA, USA
carmine.cella@berkeley.edu

ABSTRACT

Expressive piano performance generation needs symbolic pitch, timing, and dynamics. We evaluate six tokenization strategies for a Transformer that generates classical piano performances. Our tokenizations add velocity, beat annotations, and sustain pedal, from note-only to full representations. We pretrain on MAESTRO, then finetune on ASAP with beat-level annotations. The model uses anticipatory-style note encoding with cross-attention on composer and genre. FAD on the ASAP test set shows that **note + velocity + pedal** and **full** modes achieve the lowest mean FAD (1.76 and 1.97). Both beat the note-only baseline (3.10). Beat tokens show mixed, category-dependent effects and do not improve the best modes on average.

1. INTRODUCTION

Symbolic music generation has seen remarkable progress with the advent of Transformer-based language models applied to MIDI token sequences [1, 2, 3]. However, most work has focused on either generating musical scores (symbolic compositions) or generating performances in genres where timing follows a fixed grid. Classical piano performance presents a unique challenge: the performer introduces expressive tempo variations (rubato), dynamic shaping, and pedaling that go far beyond what is notated in the score.

A critical design choice in any symbolic music generation system is the *tokenization*—how continuous performance attributes such as onset time, duration, velocity, beat position, and pedal state are discretized into tokens. Different tokenizations encode different amounts of musical information, and the choice directly affects what the model can learn and generate.

In this paper, we present a systematic study of six tokenization strategies for a conditional Transformer model that generates expressive classical piano performances. Our contributions are:

- A compact, prefix-hierarchical vocabulary that supports six tokenization modes—from note-only to full—by truncating a shared token ID space.
- A two-stage training pipeline: pretraining on MAESTRO [4] to learn basic musical structure, followed by finetuning on ASAP [5] for beat-aware expressive performance.
- A systematic FAD-based evaluation showing that velocity and pedal are most useful jointly, while beat tokens have mixed, category-dependent effects.

2. RELATED WORK

2.1. MIDI Tokenization

The design of MIDI token vocabularies has evolved significantly. The REMI tokenization [1] introduced bar and beat tokens to provide metrical structure, improving generation quality for pop music. However, REMI quantizes note onsets to fixed beat subdivisions, making it unsuitable for classical performance where rubato is essential. The Anticipatory Music Transformer (AMT) [2] and MIDI-LLM [3] use compact note-tuple encodings for symbolic music generation. Inspired by this tuple structure, our representation encodes each note as a time, duration, and pitch, with optional velocity; we further add beat and pedal event tokens, enabling a systematic study of their individual contributions.

2.2. Conditional Music Generation

Conditioning on metadata such as composer or genre allows controllable generation. PopMAG [6] used track symbols and meta embeddings for instrument and tempo conditioning for simultaneous multi-track generation. MusicBERT [7] explored masked language modeling on symbolic music. Our model uses cross-attention between the token sequence and learned embeddings of composer and genre, enabling fine-grained conditioning.

2.3. Evaluation Metrics

Fréchet Audio Distance (FAD) [8] measures distributional distance between generated and reference audio in a VGGish embedding space. It correlates better with human perception than signal-level metrics. We adopt FAD as our primary metric on rendered audio from generated MIDI.

* Corresponding authors.

3. METHOD

3.1. Tokenization

We design a compact vocabulary where token IDs are allocated sequentially:
`[PAD, BOS, EOS]` (0–2),
`Time` (3–10,002), `Duration` (10,003–11,002),
`Pitch` (11,003–11,130), `Velocity` (11,131–11,258),
`Beat` (11,259–11,261), and `Pedal` (11,262–11,263). Modes that omit velocity or beat tokens still allocate the full embedding matrix. Only the relevant token IDs are predicted during training and generation. This enables loading a pretrained checkpoint from a larger mode into a smaller one. We truncate the embedding and LM head to the target vocabulary size.

Table 1 summarizes the six modes. Each note event is represented as (t, d, p) or (t, d, p, v) when velocity is included. Here t, d, p, and v denote time, duration, pitch, and velocity tokens. Beat events are represented as (t, b). The beat type b is `Beat`, `Downbeat`, or `Beat-R` (rubato). Pedal events are represented as $(t, \texttt{Pedal-On/Off})$.

Time is discretized at a resolution of 100 ticks per second (maximum 10,000 time steps, i.e. 100 seconds per window). Duration is discretized with the same resolution, up to a maximum of 999 ticks ($\approx$10 seconds). Pitch spans the standard MIDI range (0–127). Velocity covers 128 MIDI levels. Beat tokens include three types: `Beat` (regular beat), `Downbeat`, and `Beat-R` (rubato beat, where the exact beat position cannot be determined), derived from ASAP beat annotations. Pedal tokens indicate sustain pedal on/off.

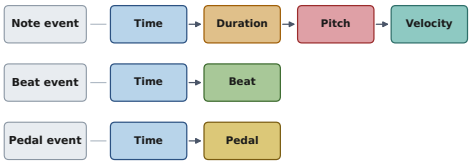

Figure 1: Three event types in the tokenization. A *note event* consists of time, duration, and pitch tokens (plus velocity when enabled). A *beat event* pairs a time with a beat type token (Beat, Downbeat, or Beat-R). A *pedal event* pairs a time with a pedal on/off token.

3.2. Model Architecture

Our model is a decoder-only Transformer (Fig. 2). Cross-attention conditions generation on composer and genre. The input sequence passes through learned token and positional embeddings, which are summed. Each of the 24 decoder layers consists of:

1. **Causal self-attention** with causal masking.
2. **Cross-attention** over a condition memory vector derived from composer and genre embeddings.
3. **Feed-forward network** (FFN) with GELU activation and dropout.

Layer normalization is applied before each sub-layer (pre-norm), and residual connections are used throughout.

3.2.1. Conditioning mechanism

Composer and genre IDs are each mapped to a 128-dimensional embedding. They are concatenated and projected through a two-layer MLP (with GELU) to a 768-dimensional memory vector ($d_{\text{model}} = 768$). This memory is shared across all layers via cross-attention.

3.2.2. Generation

During inference, tokens are generated autoregressively with nucleus sampling (top-p $=$ 0.98, temperature $=$ 0.95). A constrained decoding mechanism restricts the output vocabulary at each step based on the current phase of the token sequence (time $\rightarrow$ duration/beat/pedal $\rightarrow$ pitch $\rightarrow$ velocity), ensuring syntactically valid token sequences.

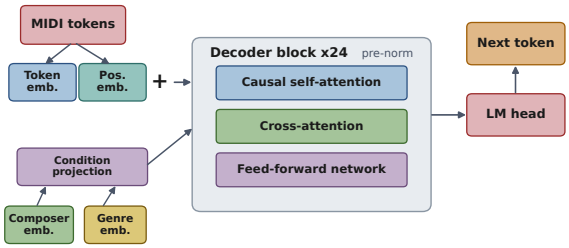

Figure 2: Model architecture. MIDI tokens are embedded and processed by 24 decoder layers with causal self-attention and cross-attention over composer/genre condition memory.

3.3. Training Pipeline

3.3.1. Pretraining on MAESTRO

We pretrain on MAESTRO [4] using the `Note+Vel+Pedal` mode. This is the richest mode compatible with MAESTRO, which lacks beat annotations. MAESTRO provides over 170 hours of piano performances with aligned MIDI and audio. The model learns pitch relationships, rhythm, dynamics, and pedal usage. We train for 10 epochs with a batch size of 32 per GPU on 8 GPUs, using AdamW with a cosine learning rate schedule.

3.3.2. Finetuning on ASAP

We finetune on the ASAP dataset [5]. This dataset provides 1,067 performances of 236 classical piano pieces with beat and downbeat annotations from 15 composers. The finetuning uses each of the six tokenization modes separately, producing six finetuned models. For modes smaller than `Note+Vel+Pedal`, we truncate the LM head and the pretrained embedding to the target vocabulary size; missing keys (e.g. beat embeddings) are randomly initialized. This runs for 50–62 epochs with a batch size of 28 per GPU.

3.3.3. Data augmentation

We augment training MIDI with pitch shift, time stretch, and velocity scaling. For MAESTRO pretraining, each original receives six pitch-shifted copies (±1, ±2, ±3 semitones), five time-stretched copies (factors uniform in $[0.8, 1.2]$), and five velocity-scaled copies ($[0.9, 1.1]$). For ASAP finetuning, we apply the

Table 1: Six tokenization modes with their token components and vocabulary sizes.

Mode	Pitch	Vel	Beat	Pedal	Evt. size	$\lvert\mathcal{V}\rvert$
Note	✓				3	11,131
Note+Pedal	✓			✓	3 / 2	11,264
Note+Vel	✓	✓			4	11,259
Note+Vel+Beat	✓	✓	✓		4 / 2	11,262
Note+Vel+Pedal	✓	✓		✓	4 / 2	11,264
Full	✓	✓	✓	✓	4 / 2	11,264

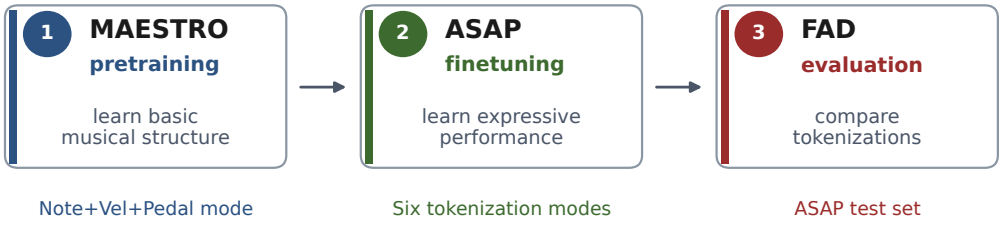

Figure 3: Training and evaluation pipeline. Pretraining on MAESTRO learns basic musical structure; finetuning on ASAP with six tokenization modes produces six models; FAD evaluation on the ASAP test set compares generation quality.

same pitch and velocity augmentations; performance MIDIs are time-stretched in $[0.9, 1.1]$. Both stages train on originals and augmented copies ($\sim$16$\times$ more rows than the base sets). FAD evaluation uses non-augmented reference performances only.

4. EVALUATION

4.1. Setup

We evaluate each of the six finetuned models on the ASAP test set. All ASAP test pieces and their corresponding MAESTRO source recordings were excluded from the MAESTRO pretraining split. We generate at least 20 MIDI performances autoregressively per composer/genre category in the test set, conditioned on the piece's composer and genre, with a maximum sequence length of 8,192 tokens. Generated MIDI files are rendered to audio using FluidSynth with a standard piano SoundFont; reference MIDI uses the same approach. We compute Fréchet Audio Distance (FAD) [8] with VGGish embeddings per composer/genre category and overall. Representative samples of the generated MIDI and rendered WAV files are available at github.com/Mikemenny/Evaluating-Tokenization-Strategies.

4.2. Results

Table 2 reports the FAD scores for each tokenization mode across 19 composer/genre categories in the ASAP test set, along with the mean FAD.

4.3. Discussion

Our results suggest that velocity and pedal are most useful jointly rather than independently. `Note+Vel` and `Note+Pedal` do not improve over `Note` in terms of mean FAD, while `Note+Vel+Pedal` substantially lowers FAD. Velocity and pedal appear mutually reinforcing in rendered audio: dynamics shape each attack, while pedal shapes decay and overlap. Beat tokens show a mixed, category-dependent effect, increasing mean FAD when added to `Note+Vel` (3.14 $\rightarrow$ 3.45) and not improving the strongest mode on average (`Full`: 1.97 vs. `Note+Vel+Pedal`: 1.76).

4.3.1. The Kreisleriana anomaly

Because Schumann/Kreisleriana is a large outlier for several modes, we also report a one-category-per-tail trimmed mean in Table 2. This robust summary preserves the main conclusion that `Note+Vel+Pedal` and `Full` are the strongest modes, while showing that their ordering is not decisive: `Note+Vel+Pedal` has the best ordinary mean FAD (1.76), whereas `Full` has the best trimmed mean FAD (1.33 vs. 1.37).

5. LIVE DEMO

We will present a live demonstration at the conference (power outlet, table, and audio cable required). Attendees can condition generation on a composer–genre pair and hear FluidSynth-rendered performances in real time; sample outputs for all six modes are provided in `artifacts/` (see Sec. 4.1).

6. CONCLUSION

We presented a systematic evaluation of six tokenization strategies for expressive classical piano performance generation. Jointly encoding velocity and pedal yields the lowest mean FAD on rendered piano audio; neither channel alone reliably improves over a note-only baseline. Adding beat tokens to Note+Vel slightly increases mean FAD (3.14 $\rightarrow$ 3.45), with large category-to-category variation; beat does not improve the best modes on average (`Full`: 1.97 vs. `Note+Vel+Pedal`: 1.76). Overall, `Note+Vel+Pedal` and `Full` achieve the best FAD scores, with `Note+Vel+Pedal` lowest by ordinary mean and `Full` slightly lowest by trimmed mean, indicating that velocity and pedal should be encoded jointly for this evaluation setup.

Table 2: FAD scores per composer/genre category for each tokenization mode. Lower is better. Best results per row are shown in **bold**.

Composer	Genre	Note	Note+Pedal	Note+Vel	Note+Vel+Beat	Note+Vel+Pedal	Full
Bach	Fugue	2.23	5.60	4.63	2.52	**1.45**	1.60
Bach	Prelude	1.62	6.88	0.70	2.19	0.68	**0.54**
Beethoven	Sonata	2.37	3.20	3.43	4.24	1.56	**1.32**
Chopin	Ballade	1.46	6.59	2.12	2.53	**0.84**	1.22
Chopin	Etude	2.70	5.03	**2.05**	2.63	2.89	2.90
Chopin	Sonata	1.66	3.06	3.00	3.86	**0.52**	0.57
Debussy	Images	4.40	1.57	3.38	5.12	1.56	**1.49**
Debussy	Suite	2.31	2.83	2.30	2.99	2.28	**1.60**
Haydn	Sonata	1.15	3.88	1.47	1.77	**0.91**	0.97
Liszt	Etude	2.29	**1.49**	2.33	4.81	2.01	1.67
Liszt	Rhapsody	4.27	2.87	**2.22**	3.59	2.71	2.62
Liszt	Sonata	3.57	1.33	4.71	4.65	**0.72**	0.81
Ravel	Miroirs	1.56	2.83	1.60	2.17	**0.77**	0.78
Schubert	Fantasie	1.31	4.85	4.29	2.46	**0.57**	1.29
Schubert	Impromptu	2.38	4.82	2.33	4.57	1.61	**1.31**
Schubert	Sonata	2.72	5.31	5.69	6.01	1.42	**1.31**
Schumann	Arabeske	2.70	4.64	1.39	1.99	0.69	**0.57**
Schumann	Kreisleriana	15.22	8.87	8.19	**3.54**	9.72	14.48
Scriabin	Sonata	3.05	1.38	3.74	3.94	0.54	**0.44**
Mean		3.10	4.05	3.14	3.45	**1.76**	1.97
Trimmed mean		2.51	3.93	2.98	3.40	1.37	**1.33**

Trimmed mean drops each mode's min/max category FAD; FAD can be sensitive to sample size and outlier generations.

Future work could investigate why beat tokens help some composer/genre categories but not the overall average. We also plan richer controllable generation—for example, an interactive web tool with composer–genre conditioning, as outlined in our live demo. Other directions include continuous time representations, finer pedal modeling (half-pedal and una corda), larger-scale pretraining on diverse repertoire, and extending the two-stage paradigm to other instruments.

Acknowledgments

The authors would like to thank the Center for New Music and Audio Technologies (CNMAT) at the University of California, Berkeley, for providing research facilities and institutional support.

7. REFERENCES

[1] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proceedings of the 28th ACM international conference on multimedia*, 2020, pp. 1180–1188.

[2] J. Thickstun, D. Hall, C. Donahue, and P. Liang, "Anticipatory music transformer," *Transactions on Machine Learning Research*, 2024, accepted by TMLR. [Online]. Available: https://openreview.net/forum?id=EBNJ33Fcrl

[3] S.-L. Wu, Y. Kim, and C.-Z. A. Huang, "MIDI-LLM: Adapting large language models for text-to-midi music generation." arXiv, 2025, NeurIPS 2025 Workshop on AI for Music. [Online]. Available: https://arxiv.org/abs/2511.03942

[4] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the maestro dataset," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://arxiv.org/abs/1810.12247

[5] F. Foscarin, A. McLeod, P. Rigaux, F. Jacquemard, and M. Sakai, "ASAP: A dataset of aligned scores and performances for piano transcription," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 534–541. [Online]. Available: https://archives.ismir.net/ismir2020/paper/000127.pdf

[6] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, "PopMAG: Pop music accompaniment generation," in *Proceedings of the 28th ACM international conference on multimedia*, 2020, pp. 1198–1206.

[7] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T.-Y. Liu, "MusicBERT: Symbolic music understanding with large-scale pre-training," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021, pp. 791–800.

[8] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms," in *Interspeech 2019*, 2019. [Online]. Available: https://doi.org/10.21437/Interspeech.2019-2219