MIDI-LLM: Adapting Large Language Models for Text-to-MIDI Music Generation

Shih-Lun Wu **Yoon Kim** **Cheng-Zhi Anna Huang**
Massachusetts Institute of Technology
slseanwu@mit.edu

Abstract

We present MIDI-LLM, an LLM for generating multitrack MIDI music from free-form text prompts. Our approach expands a text LLM's vocabulary to include MIDI tokens, and uses a two-stage training recipe to endow text-to-MIDI abilities. By preserving the original LLM's parameter structure, we can directly leverage the `vLLM` library for accelerated inference. Experiments show that MIDI-LLM achieves higher quality, better text control, and faster inference compared to the recent Text2midi model. Live demo at https://midi-llm-demo.vercel.app.

1 Introduction

Recent advances in text-to-music models empowered users to generate realistic-sounding audio music from simple natural language prompts (Copet et al., 2023; Evans et al., 2024). However, even with fine-grained control mechanisms (Wu et al., 2024; Tsai et al., 2025), the opaque nature of audio outputs makes them difficult to edit, restructure, or reuse in downstream workflows. This limits their ability to support the kind of iterative human-AI collaboration that Large Language Models (LLMs) have enabled in text domains. In contrast, symbolic-domain models, which most commonly generate MIDI outputs (Huang et al., 2019; Wu and Yang, 2023; Thickstun et al., 2024) that allow direct editing, rearrangement, and reuse, have been praised by musicians for promoting creative agency (Donahue et al., 2024; Kim et al., 2025). Yet, they largely lack effective free-form text control of their audio-domain counterparts, and often rely on custom architectures (Bhandari et al., 2025; Wang et al., 2025) that are difficult to accelerate for usability.

To imbue symbolic models with effective text control and fast inference, we propose MIDI-LLM, an adaptation of LLMs for text-to-MIDI generation. Our motivation is twofold: (i) LLMs encode broad world knowledge and emotional semantics, making them well-suited for conditioning on text prompts, and (ii) the LLM ecosystem is well-developed with easy-to-use inference-time optimizations. Concretely, we expand the token embedding layer of a pretrained text LLM (Grattafiori et al., 2024) to incorporate MIDI tokens (Sec. 3.1). We then train the model in two stages, first on music-adjacent text and standalone MIDIs, then on MIDIs paired with text (Sec. 3.2). Since we preserve the original LLM's parameter structure, we can easily leverage `vLLM` (Kwon et al., 2023; Shaw et al., 2024) for accelerated inference. Our experiments confirm that MIDI-LLM achieves higher quality, stronger text control, and significantly faster inference than the Text2midi baseline (Bhandari et al., 2025).

We provide a live demo website[1] to showcase MIDI-LLM's capabilities and collect feedback. We also release our code[2] and trained model weights.[3]

2 Background: MIDI Tokenization

Various approaches have been proposed to tokenize MIDI music for language modeling. Popular families include: *MIDI-like* tokens (Oore et al., 2018; Huang et al., 2019), metered Revamped MIDI

[1]Live demo: https://midi-llm-demo.vercel.app
[2]Code: https://github.com/slSeanWU/MIDI-LLM
[3]Model: https://huggingface.co/slseanwu/MIDI-LLM_Llama-3.2-1B

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: AI for Music.

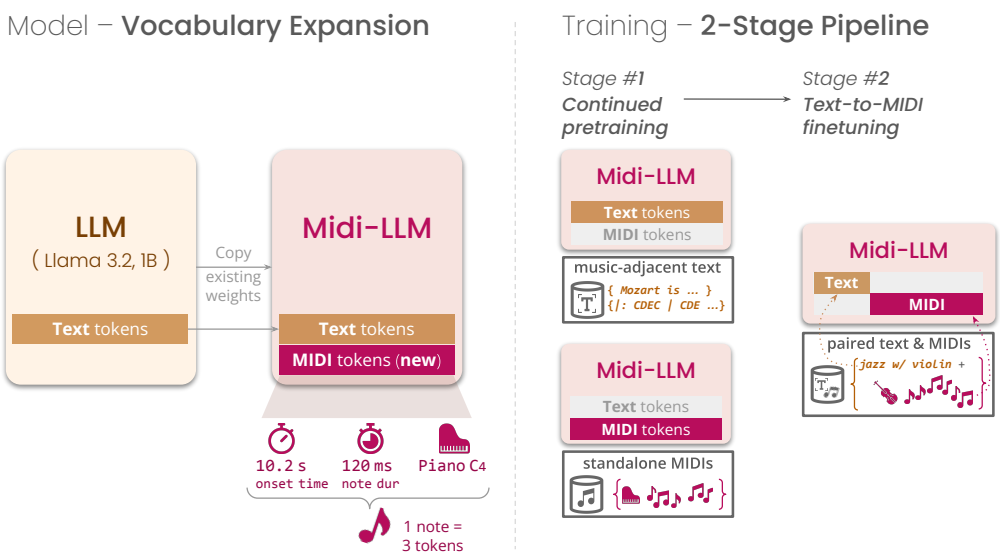

Figure 1: MIDI-LLM recipe overview. We initialize MIDI-LLM by expanding the token embeddings of *Llama 3.2* 1B LLM (Grattafiori et al., 2024) with the MIDI vocabulary defined in *Anticipatory Music Transformer (AMT)* (Thickstun et al., 2024). We then train the full model in two stages to achieve text-to-MIDI generation. See Table 1 for more information on our training data.

tokens (*REMI*) (Huang and Yang, 2020; Hsiao et al., 2021; Wu and Yang, 2023), and text-based *ABC*-derived notations (Yuan et al., 2024; Qu et al., 2024; Wang et al., 2025). For MIDI-LLM, we adopt the (arrival-time) MIDI-like tokenization from Anticipatory Music Transformer (AMT) (Thickstun et al., 2024). This approach offers more flexibility, as it does not require beat-synchronized data that is a prerequisite for REMI and ABC-based approaches.

The AMT arrival-time tokenization represents each musical note as three consecutive events:

- **Arrival (onset) time:** The note's start time – 0th to 100th second with 10 ms quantization.
- **Note duration:** How long the note is held – 0 to 10 seconds with 10 ms quantization.
- **Instrument-pitch:** A joint token for the instrument and its pitch – 129 MIDI inst. $\times$ 128 pitches.

This amounts to 27.5K possible tokens. Additionally, AMT also uses a separate set of the same tokens, called *anticipated* tokens, to represent future notes given to the language model as conditions for music infilling tasks. Hence, the total vocabulary size is 55K tokens.

3 Method

3.1 LLM Vocabulary Expansion

We use the popular *Llama 3.2* 1B LLM (Grattafiori et al., 2024) as our pretrained backbone. To incorporate the AMT music tokens (cf. Sec. 2), a naive approach is to serialize them as text (e.g., '`<onset 10.2s> <duration 120ms> <piano, C4>`') and feed to the LLM's text tokenizer. However, this would unnecessarily lengthen the sequence and hence increase compute overhead. Therefore, in MIDI-LLM, we expand the LLM's token embedding weights, $\mathbf{E}_{\text{LLM}}$, via:

$$\mathbf{E}_{\text{MIDI-LLM}} := \begin{bmatrix}\mathbf{E}_{\text{LLM}}^{\top} & \mathbf{E}_{\text{AMT}}^{\top}\end{bmatrix}^{\top} \in \mathbb{R}^{(|\mathcal{V}_{\text{LLM}}|+|\mathcal{V}_{\text{AMT}}|)\times D}, \tag{1}$$

where $|\mathcal{V}_{\text{LLM}}|$ is the original text vocabulary size (128K for Llama 3.2), $|\mathcal{V}_{\text{AMT}}|$ is the new music vocabulary size (55K), and D is the hidden-state dimension. With this design, each note becomes exactly three tokens to the LLM. We initialize $\mathbf{E}_{\text{AMT}}^{\top}$ randomly and train the entire LLM subsequently.

3.2 Two-Stage Training

We train MIDI-LLM in two sequential stages: *continued pretraining* on broad text/MIDI data, then *supervised finetuning* on targeted data. The model is trained using a standard next-token prediction objective in both stages. A summary of datasets used in training MIDI-LLM is shown in Table 1.

Table 1: MIDI-LLM two-stage training data summary.

	Continued Pretraining		*Finetuning*
	Text	MIDI	Text-to-MIDI
Domain(s)	• Music-related web articles (56%) • Synthetic music knowledge (by GPT-4) (17%) • QAs – music in ABC notation (16%) • QAs – common knowledge (12%)	*Standalone* multitrack MIDIs	Multitrack MIDIs *paired w/ text*
Dataset(s)	(Subset of) MusicPile (Yuan et al., 2024)	GigaMIDI (Lee et al., 2025)	MidiCaps (for paired text) (Melechovsky et al., 2024) + LMD (for MIDI content) (Raffel, 2016)
# Tokens	1.69 B	1.38 B	1.71 B (w/o augmentation)
Seqlen	2048	2048 (≈30 sec. music)	Text ≤256 (as prefix) MIDI 2048

Continued pre-training. Following common practices to specialize LLMs for certain domains (Gururangan et al., 2020; Ibrahim et al., 2024), we perform a continued pre-training stage with two primary goals: (i) to surface the musical knowledge (that exists in text) the LLM might have seen in its initial pre-training, and (ii) to teach it the syntax and structure of MIDI data under AMT's tokenization. We compile a dataset of around 3B tokens for this stage.

- **Music-adjacent text:** A subset of *MusicPile* (Yuan et al., 2024) containing primarily music-related articles, music knowledge, and ABC-notated music.
- **Standalone MIDIs:** The music pieces in *GigaMIDI* (Lee et al., 2025), with those reserved for text-to-MIDI finetuning removed.

Supervised finetuning. To enable music generation from textual descriptions, we finetune the model on paired text-MIDI data, teaching it to translate musical concepts expressed in text into MIDI notes. We construct each example with a text prompt from *MidiCaps* (Melechovsky et al., 2024) as the 'instruction prefix', which contains attributes like genre, mood, instrumentation, tempo, tonality, and chords of the music, followed by the AMT tokens of the corresponding MIDI in *Lakh MIDI (LMD)* (Raffel, 2016). For data augmentation, we construct music infilling examples natively supported by AMT's tokenization. We leverage *Qwen2.5-Omni* (Xu et al., 2025), an LLM capable of music captioning, to produce text prompts for these examples, adding further textual diversity. The token count for this stage is around 1.7B pre-augmentation, and 5.1B (tripled) post-augmentation.

4 Experiments and Results

4.1 Implementation Details

Training. We base our implementation on HuggingFace `transformers` package (Wolf et al., 2020) and instantiate MIDI-LLM from `LlamaForCausalLM` with minimal changes. We start from the official Llama 3.2 (1B) checkpoint, and train MIDI-LLM using `transformers`' built-in trainer with FlashAttention-2 (Dao, 2024) and BF16 precision. We perform 25K steps of gradient updates with AdamW (Loshchilov and Hutter, 2017) optimizer for both training stages. We set the effective batch size to 512K (1M) tokens for the continued pretraining (supervised finetuning) stage, and the learning rate to 2×10^{-4} with cosine decay. The entire training run (two stages combined) takes around 6 days on 4× H100 (SXM, 80G) GPUs.

Inference. We use *nucleus sampling* (Holtzman et al., 2020) with top $p = 0.98$ to balance musical diversity and coherence. We apply serving-time optimizations in `vLLM` package, including CUDA graphs, KV cache paged attention (Kwon et al., 2023), and FP8-W8A8 dynamic weight quantization (Shaw et al., 2024). In our pilot tests, these techniques accelerated inference by >50% compared to `transformers`' default setup.

4.2 Evaluation Metrics and Baseline

Following the standard for text-to-music models (Copet et al., 2023; Evans et al., 2024; Wu et al., 2024), we evaluate MIDI-LLM's outputs using the following two automatic metrics.

Table 2: Quality metrics and inference speeds on *MidiCaps* (Melechovsky et al., 2024) test set (subset, # samples = 896). MIDI-LLM achieves higher quality and faster generation than *Text2midi* (Bhandari et al., 2025). (RTF = *real-time factor*: generated music duration / wall-clock time.)

Model	Params	Prec.	Quality		Speed (2K generation seqlen on 1× L40S GPU)				
			FAD↓	CLAP↑	Time taken↓ (s)		Avg. output duration (s)	RTF↑	
					bsz=1	bsz=4		bsz=1	bsz=4
Text2midi	0.27 B	FP32	0.818	18.7	47.0	99.4	26.3	0.56	1.06
MIDI-LLM (Ours)	1.47 B	BF16	**0.173**	**22.1**	10.0	11.6	33.3	3.33	11.48
		FP8	0.216	21.8	**8.1**	**9.2**	32.6	**4.02**	**14.17**

- **FAD** (Kilgour et al., 2019): measures roughly the outputs' *quality* or *realisticness* using the Fréchet distance between the feature covariances induced by a set of model generations and those induced by a groundtruth set. We employ VGGish (Hershey et al., 2017) as the feature extractor.
- **CLAP** (Wu et al., 2023): is meant to capture each output's *relevance to text prompt* using pairwise feature cosine similarity between a contrastively trained text encoder (receiving the *prompt*) and audio encoder (receiving the *model output*). We use the `music` checkpoint provided by Laion-AI.

Since both metrics require audio inputs, we synthesize our MIDI outputs with `fluidsynth` package.[4]

We select a recent text-to-MIDI model, *Text2midi* (Bhandari et al., 2025), trained also on MidiCaps + LMD as our baseline. Unlike ours, it uses an encoder-decoder setup, with a (frozen) pretrained T5 (Raffel et al., 2020) encoder providing text conditions, and a trained-from-scratch decoder modeling REMI-like (Huang and Yang, 2020) music tokens. Since our MidiCaps held-out split is different from that of Text2midi, we identify the intersection (with 896 samples) for evaluation.[5]

4.3 Results and Discussion

The metrics are reported in Table 2. For generation quality, our MIDI-LLM outperforms Text2midi on both metrics by considerable margins. Despite being a larger model (1.47B vs. 0.27B parameters), MIDI-LLM is much faster at inference across batch sizes. Using FP8 quantization yields additional speedup (∼20%) and memory savings (50% in theory) with only a modest impact on quality. These results highlight the advantages of our LLM adaptation approach, which delivers superior output quality and text controllability, and enables using inference-time optimizations readily available in the broader LLM ecosystem. Although similar optimizations could be implemented for custom architectures like Text2midi, doing so would require substantially greater engineering effort.

We also report two negative findings. First, although MIDI-LLM is trained on infilling examples paired with text prompts (see Sec. 3.2), the text has minimal influence during inference, i.e., the infilled segments are primarily determined by the surrounding MIDI context. Second, when we replace the music-adjacent text corpus, *MusicPile*, with a general-domain text corpus, *FineWeb-Edu* (Penedo et al., 2024), for continued pretraining, we observe no noticeable change in the final text-to-MIDI performance. While potential confounders exist, e.g., differences in data curation quality, this raises questions about the necessity of music-adjacent text for continued pretraining. Both phenomena highlight promising directions for future investigation.

5 Future Work

Our MIDI-LLM opens several interesting avenues for future research. Feedback from our live demo website can be used for preference-tuning techniques such as RLHF (Ouyang et al., 2022) and DPO (Rafailov et al., 2023), and potentially to personalize the model to individual users' tastes. Beyond the current non-iterative text-to-MIDI task, a crucial direction is to develop text-guided editing capabilities (Schick et al., 2023). In parallel, engaging with musicians through interviews and co-creation sessions (Huang et al., 2020) can help identify the most valuable control mechanisms to further incorporate for real-world creative practices.

[4] `fluidsynth` soundfont: https://member.keymusician.com/Member/FluidR3_GM/.

[5] Note that Text2midi's tokenization supports varying note dynamics (i.e., loudness) while ours does not. Therefore, we compute metrics for Text2midi both *with* and *without* dynamics and report the *better* one.

References

K. Bhandari, A. Roy, K. Wang, G. Puri, S. Colton, and D. Herremans. Text2midi: Generating symbolic music from captions. In *Proc. AAAI*, 2025.

J. Copet, F. Kreuk, I. Gat, T. Remez, D. Kant, G. Synnaeve, Y. Adi, and A. Défossez. Simple and controllable music generation. In *Proc. NeurIPS*, 2023.

T. Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *Proc. ICLR*, 2024.

C. Donahue, S.-L. Wu, Y. Kim, D. Carlton, R. Miyakawa, and J. Thickstun. Hookpad Aria: A copilot for songwriters. In *Proc. ISMIR Late-breaking Demos*, 2024.

Z. Evans, C. Carr, J. Taylor, S. H. Hawley, and J. Pons. Fast timing-conditioned latent audio diffusion. In *Proc. ICML*, 2024.

A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

S. Gururangan, A. Marasovic, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proc. ACL*, 2020.

S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, et al. CNN architectures for large-scale audio classification. In *Proc. ICASSP*, 2017.

A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. The curious case of neural text degeneration. In *Proc. ICLR*, 2020.

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*, 2021.

C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck. Music Transformer: Generating music with long-term structure. In *Proc. ICLR*, 2019.

C.-Z. A. Huang, H. V. Koops, E. Newton-Rex, M. Dinculescu, and C. J. Cai. AI song contest: Human-AI co-creation in songwriting. In *Proc. ISMIR*, 2020.

Y.-S. Huang and Y.-H. Yang. Pop Music Transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proc. ACM-MM*, 2020.

A. Ibrahim, B. Thérien, K. Gupta, M. L. Richter, Q. Anthony, T. Lesort, E. Belilovsky, and I. Rish. Simple and scalable strategies to continually pre-train large language models. *Trans. on Machine Learning Research (TMLR)*, 2024.

K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi. Fréchet audio distance: A metric for evaluating music enhancement algorithms. In *Proc. Interspeech*, 2019.

Y. Kim, S.-J. Lee, and C. Donahue. Amuse: Human-AI collaborative songwriting with multimodal inspirations. In *Proc. CHI*, 2025.

W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica. Efficient memory management for large language model serving with pagedattention. In *Proc. Symposium on Operating Systems Principles (SOSP)*, 2023.

K. J. M. Lee, J. Ens, S. Adkins, P. Sarmento, M. Barthet, and P. Pasquier. The gigamidi dataset with features for expressive music performance detection. *Trans. Int. Soc. for Music Information Retrieval (TISMIR)*, 2025.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

J. Melechovsky, A. Roy, and D. Herremans. MidiCaps: A large-scale midi dataset with text captions. In *Proc. ISMIR*, 2024.

S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan. This time with feeling: Learning expressive musical performance. *arXiv preprint arXiv:1808.03715*, 2018.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. In *Proc. NeurIPS*, 2022.

G. Penedo, H. Kydlíček, A. Lozhkov, M. Mitchell, C. A. Raffel, L. Von Werra, T. Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. In *Proc. NeurIPS Datasets & Benchmarks*, 2024.

X. Qu, Y. Bai, Y. Ma, Z. Zhou, K. M. Lo, J. Liu, R. Yuan, L. Min, X. Liu, T. Zhang, et al. MuPT: A generative symbolic music pretrained transformer. *arXiv preprint arXiv:2404.06393*, 2024.

R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. In *Proc. NeurIPS*, 2023.

C. Raffel. *Learning-based methods for comparing sequences, with applications to audio-to-MIDI alignment and matching*. PhD thesis, Columbia University, 2016.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research (JMLR)*, 2020.

T. Schick, A. Y. Jane, Z. Jiang, F. Petroni, P. Lewis, G. Izacard, Q. You, C. Nalmpantis, E. Grave, and S. Riedel. PEER: A collaborative language model. In *Proc. ICLR*, 2023.

R. Shaw, M. Kurtz, S. Adkins, and B. Fineran. LLM compressor is here: Faster inference with vLLM. *Red Hat Developer Blogs*, 2024. URL https://developers.redhat.com/articles/2024/08/14/llm-compressor-here-faster-inference-vllm.

J. Thickstun, D. Hall, C. Donahue, and P. Liang. Anticipatory music transformer. *Trans. on Machine Learning Research (TMLR)*, 2024.

F.-D. Tsai, S.-L. Wu, W. Lee, S.-P. Yang, B.-R. Chen, H.-C. Cheng, and Y.-H. Yang. MuseControlLite: Multifunctional music generation with lightweight conditioners. In *Proc. ICML*, 2025.

Y. Wang, S. Wu, J. Hu, X. Du, Y. Peng, Y. Huang, S. Fan, X. Li, F. Yu, and M. Sun. NotaGen: Advancing musicality in symbolic music generation with large language model training paradigms. *arXiv preprint arXiv:2502.18008*, 2025.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proc. EMNLP Demos*, 2020.

S.-L. Wu and Y.-H. Yang. Compose & Embellish: Well-structured piano performance generation via a two-stage approach. In *Proc. ICASSP*, 2023.

S.-L. Wu, C. Donahue, S. Watanabe, and N. J. Bryan. Music ControlNet: Multiple time-varying controls for music generation. *IEEE/ACM T-ASLP*, 2024.

Y. Wu, K. Chen, T. Zhang, Y. Hui, T. Berg-Kirkpatrick, and S. Dubnov. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *Proc. ICASSP*, 2023.

J. Xu, Z. Guo, J. He, H. Hu, T. He, S. Bai, K. Chen, J. Wang, Y. Fan, K. Dang, et al. Qwen2.5-omni technical report. *arXiv preprint arXiv:2503.20215*, 2025.

R. Yuan, H. Lin, Y. Wang, Z. Tian, S. Wu, T. Shen, G. Zhang, Y. Wu, C. Liu, Z. Zhou, et al. ChatMusician: Understanding and generating music intrinsically with LLM. In *Proc. ACL Findings*, 2024.