Amadeus: Autoregressive Model with Bidirectional Attribute Modelling for Symbolic Music

Hongju Su[1], Ke Li[1,2*], Lan Yang[1,2], Honggang Zhang[1], Yi-Zhe Song[2]

[1]School of Artificial Intelligence, Beijing University of Posts and Telecommunications, China
[2]SketchX, CVSSP, University of Surrey, United Kingdom
{hongjusu, like1990, ylan, zhhg}@bupt.edu.cn, y.song@surrey.ac.uk
*Corresponding Author https://github.com/lingyu123-su/Amadeus

Abstract

Existing state-of-the-art symbolic music generation models represent symbolic music as a sequence of attribute tokens with fixed unidirectional dependencies. However, from the perspective of music theory, the attributes of a musical note are inherently a set rather than a sequence. Building on this insight, we propose Amadeus, a novel symbolic music generation framework that adopts a two-level architecture: an autoregressive model for note sequences and a bidirectional discrete diffusion model for note attributes. This design enables flexible attribute control and adjustable decoding speed during inference. To further enhance sequential modeling, we introduce the Conditional Information Enhancement Module (CIEM). We also constructed AMD (Amadeus MIDI Dataset)—the largest open-source symbolic music dataset to date—supporting both pre-training and fine-tuning. We trained two models of different scales, Amadeus and Amadeus-M, and conducted extensive experiments, demonstrating substantial improvements over state-of-the-art methods across both objective and subjective metrics.

1 Introduction

Algorithmic composition has long been a core challenge in music theory, intersecting technical hurdles with music aesthetics(Xenakis, 1992), creativity research(Hiller and Isaacson, 1957), and human-computer interaction(Cope, 2000). In recent years, deep learning-based music generation methods (Dhariwal et al., 2020; Agostinelli et al., 2023; Huang et al., 2018; Huang and Yang, 2020; Hsiao et al., 2021) have driven rapid advancements in this field. From a modelling perspective, existing approaches primarily fall into two categories: audio-based generation (Dhariwal et al., 2020; Agostinelli et al., 2023) and symbolic music-based generation (Huang et al., 2018; Huang and Yang, 2020; Dong et al., 2023; Qu et al., 2024).

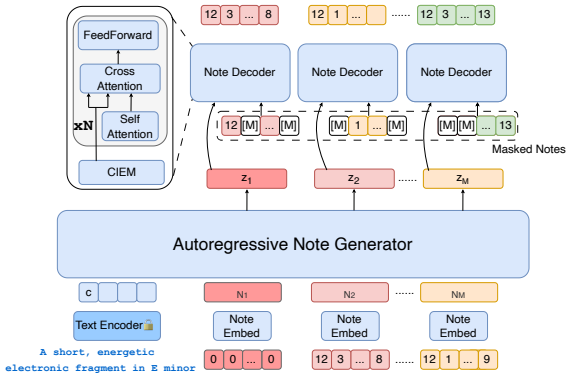

Fig. 1: Overview of the Amadeus framework. The framework consists of two main components: a note generator that auto regressively generates note-level latent variables, and a bidirectional discrete diffusion note decoder that decodes these latent variables into multi-dimensional attributes iteratively.

Compared to audio-based methods, symbolic music represents music as sequences of performance signals (*e.g.*, MIDI), offers significant advantages in lossless editing capability and compact representation.

Symbolic music can be characterized as a time series of notes, each composed of multiple attributes (*e.g.*, pitch, instrument). Consequently, auto-regressive modelling is widely used for symbolic music generation (Roy et al., 2025; Lu et al., 2023; Huang et al., 2018; Huang and Yang, 2020) and has achieved notable success. However, (Roy et al., 2025) directly applying auto-regressive modelling to generate attribute sequences step-by-step (treating intra-note attributes as independent tokens) results in excessively long sequences (max sequence length = number of notes × number of attributes), making it difficult to model long-term musical structures. Some studies (Jiwoo Ryu and Jeong, 2024) decouple the generation process into two steps: first auto regressively generating note-level latent variables, then decoding them into multi-dimensional attributes. While this approach

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 40910–40928
July 2-7, 2026

effectively reduces note sequence length, it compromises the quality of the generated symbolic music.

In existing approaches, including both autoregressive and hierarchical autoregressive models, intra-note attributes are typically assumed to follow a unidirectional dependency structure. A representative generation process first predicts pitch, followed by instrument, duration, and other attributes in a predefined order. However, according to established music theory (Bharucha and Krumhansl, 1983), fundamental note attributes such as pitch, instrument, and duration do not exhibit strictly unidirectional dependencies; instead, they are governed by complex bidirectional and interdependent relationships. In Appendix A, we provide a detailed analysis of the interactions among these attributes, and the results further support this observation.

Consequently, the theoretical foundation of autoregressively modelling intra-note attributes is questionable. Existing studies (*e.g.*, CP-Word (Hsiao et al., 2021), which uses the note type to guide generation, and NMT (Jiwoo Ryu and Jeong, 2024), which initializes generation from metric or pitch attributes) report comparable generation quality, also suggesting that imposing a strict and fixed dependency order among attributes may be unnecessary. Moreover, enforcing a sequential, attribute-by-attribute generation scheme not only introduces efficiency bottlenecks but also limits the expressiveness and controllability of symbolic music generation.

We propose Amadeus, a novel model for symbolic music generation that explicitly captures the complex interdependencies among note attributes. Amadeus adopts a two-level architecture: an autoregressive model for the note sequence and a bidirectional discrete diffusion model at the attribute level. By removing the ordering assumptions inherent in autoregressive modeling, this approach endows Amadeus with substantial flexibility during inference compared to both standard and hierarchical autoregressive models, enabling arbitrary attribute control and free adjustment of decoding speed. To improve the sequential modeling capability,we propose the Conditional Information Enhancement Module (CIEM), which amplifies discriminative features in auto regressively generated note latent vectors through attention mechanisms and incorporates global contextual information.

To validate the effectiveness of the proposed model under both pre-training and fine-tuning paradigms, we compiled, curated, and open-sourced the largest symbolic music dataset to date, termed AMD (Amadeus MIDI Dataset). The dataset consists of a 1.9-million-sample pre-training corpus and a 320,000-sample high-quality fine-tuning set with rich textual annotations. We trained Amadeus with model scales of 200M(Amadeus) and 500M(Amadeus-M) parameters.

We conducted extensive experiments using both objective and subjective evaluation metrics. The results demonstrate that Amadeus significantly outperforms existing methods in terms of musical quality, condition consistency, attribute controllability, and inference efficiency. Furthermore, we show that Amadeus enables fine-grained control over note-level attributes and allows for flexible trade-offs between decoding speed and generation quality during inference.

Our core contributions are summarized as follows:

- We propose the Amadeus architecture—performing autoregressive modelling at the note level and bidirectional modelling at the attribute level via discrete diffusion—enhanced by the CIEM to enhance sequential modeling capabilities.
- We introduce Amadeus and Amadeus-M, and demonstrate that Amadeus outperforms existing methods in terms of musical quality, condition adherence, attribute controllability, and inference speed, as validated by both subjective and objective evaluation metrics.In addition, we demonstrate model's ability to perform attribute control and speed-quality trade-offs during inference.
- We compile and open-source the largest public symbolic music dataset, AMD, comprising 1.9 million pre-training samples and 320,000 annotated fine-tuning samples.

2 Related Work

2.1 Computer-Aided Composition

The earliest attempt at computer-aided music composition was the *Illiac Suite* (later known as *String Quartet No. 4*), created using the ILLIAC I computer (Hiller and Isaacson, 1957). Its fourth movement employed probabilistic models and Markov chains to generate rhythm and melody. Subsequent research (Cope, 2000) explored rule-based and

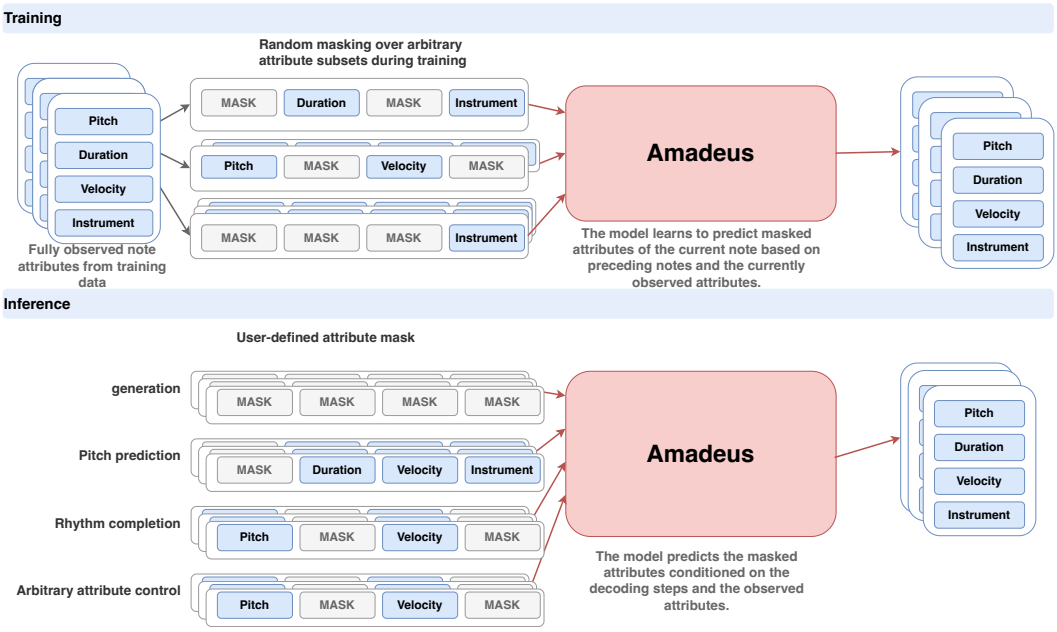

Fig. 2: Details of the training and inference processes of Amadeus.

probabilistic systems, with methods like transition matrices built on existing melodies (Collins et al., 2011). These approaches introduced mathematical formalism into music composition but suffered from poor musical quality due to limited expressive modelling capacity.

Later efforts turned to machine learning and neural networks (Nakamura et al., 2015; Mogren, 2016; Raffel, 2016), which improved fluency but still lacked convincing musicality. With deep learning, Transformer-based models (Huang et al., 2018) significantly enhanced generation performance, yet struggled with modelling long symbolic sequences. (Huang and Yang, 2020)introduced REMI tokenization and used Transformer-XL(Dai et al., 2019) to extended context but introduced training complexity and unstable results.

Attribute-merging methods (Hsiao et al., 2021) reduced sequence length by combining note features, but led to degraded expressiveness. Improvements such as attribute-level teacher-forcing (Jiwoo Ryu and Jeong, 2024) enhanced model capacity but imposed strict sequential assumptions among unordered attributes, limiting generation diversity, speed, and controllability.

2.2 Autoregressive Model Improvements

GPT-style autoregressive models are constrained by slow generation, weak long-range dependency modelling, and lack of bidirectionality. One line of work proposed partially autoregressive generation: (Yu et al., 2023) replaced tokenization with letter patches and used a two-stage autoregressive process to model mega-length sequences. Despite flexibility, generation remained dependent on sequential decoding. Patch-based improvements (Pagnoni et al., 2024) slightly boosted performance but retained these limitations.

Another line of work moved beyond autoregression. Discrete diffusion models (Austin et al., 2021) enabled parallel generation by discretizing continuous diffusion processes. Follow-up studies (Nie et al., 2025a,b) demonstrated scalability, but training remained slow and inference unstable.

Many symbolic music models (Wang et al., 2025; Plasser et al., 2023; Guo and Dixon, 2025; Wu et al., 2024) adopt these methods, yet ignore key domain properties. For instance, (Jiwoo Ryu and Jeong, 2024) treat note attributes as ordered; others (Plasser et al., 2023) neglect the temporal structure of note sequences. Such mismatches limit generation quality, control, and efficiency.

3 Methodology

3.1 Preliminary

In symbolic music generation tasks, a musical work $\mathcal{M}$ is represented as a time series $\mathcal{M} = \{\mathcal{N}_0, \mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_M\}$, where $M+1$ denotes the number of notes, $\mathcal{N}_0$ is an all-zero vector, serving as the initial note for all musical work $\mathcal{M}$. Each note $\mathcal{N}_M \in \mathbb{R}^K$ is defined by a K-dimensional feature vector, with each dimension corresponding to a

core musical attribute (detailed attribute definitions are available in the supplementary material). The joint state of these attributes uniquely determines each note in the musical sequence.

Note Embedding The note $\mathcal{N}_m$ has each attribute $\mathcal{N}_m^k$ treated as a token. The note embedding module maps every attribute $\mathcal{N}_m^k$ to a d-dimensional embedding vector $\mathbf{e}_m^k \in \mathbb{R}^d$. The composite note embedding $\mathbf{n}_m \in \mathbb{R}^d$ is then computed by aggregating all attribute embedding vectors $\mathbf{e}_m^k$ with a positional embedding vector through summation:

$$\mathbf{n}_m = \sum_{k=0}^{K-1} \mathbf{e}_m^k + \mathbf{p}_m \tag{1}$$

where $\mathbf{p}_m \in \mathbb{R}^d$ is a learnable absolute positional embedding vector corresponding to the m-th note.

Note Generator Composite note embedding sequence $\{\mathbf{n}_0, \mathbf{n}_1, \ldots, \mathbf{n}_m\}$ is fed into the note generator G, which operates as a Transformer-based decoder. Conditioned on the input sequence $\mathbf{c}$, it generates the latent vector $\mathbf{z}_{m+1} \in \mathbb{R}^d$ for the next note in an autoregressive manner, where d is the hidden dimension of the Transformer backbone:

$$\mathbf{z}_{m+1} = G(\mathbf{c}, \mathbf{n}_0, \mathbf{n}_1, \ldots, \mathbf{n}_m) \tag{2}$$

Note Decoder The note decoder in symbolic music generation methods predominantly relies on sequential autoregressive decoding of note attributes (Jiwoo Ryu and Jeong, 2024). This decoding paradigm not only violates their essential nature but also introduces significant efficiency bottlenecks by enforcing strictly sequential generation of all attributes. To address these challenges, we propose a Discrete Diffusion Model (Nie et al., 2025b)(DDM) based note decoding framework operating at the attribute granularity.

Discrete Diffusion Model (DDM) We employ Masked Diffusion Model (MDM)—a discrete diffusion model to model bidirectional dependencies between note attributes. In MDM, each note $\mathbf{n}_{m+1}$ is treated as the starting state x_0. During the forward process, the noise level at each time step t is controlled by the hyper-parameter $\alpha_t = 1 - t$. With probability p_{mask}, each token is independently replaced by the mask symbol $[M]$. The forward process is defined as:

$$q_{t|0}(\boldsymbol{x}_t|\boldsymbol{x}_0) = \prod_{k=0}^{K-1} q_{t|0}(x_t^k|x_0^k),$$

$$q_{t|0}(x_t^k|x_0^k) = \begin{cases} \alpha_t, & x_t^k = x_0^k \\ 1-\alpha_t, & x_t^k = \texttt{[M]} \end{cases} \tag{3}$$

The reverse process is initiated from a fully masked sequence $\boldsymbol{x}_T$ and progressively recovers the masked tokens:

$$q_{s|t}(\boldsymbol{x}_s|\boldsymbol{x}_t) = \prod_{k=0}^{K-1} q_{s|t}(\boldsymbol{x}_s^k|\boldsymbol{x}_t),$$

$$q_{s|t}(\boldsymbol{x}_s^k|\boldsymbol{x}_t) = \begin{cases} 1, & x_t^k \neq \texttt{[M]},\ x_s^k = x_t^k \\ \frac{s}{t}, & x_t^k = \texttt{[M]},\ x_s^k = \texttt{[M]} \\ \frac{t-s}{t} q_{0|t}(\boldsymbol{x}_s^k|\boldsymbol{x}_t), & x_t^k = \texttt{[M]},\ x_s^k \neq \texttt{[M]} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

3.2 DDM Based Note Decoder

Conditional Information Enhancement Module (CIEM) The CIEM is designed to refine the representations produced by the note generator, thereby enhancing the model's sequential modeling capability. This module reconstructs the output latent vector $\mathbf{z}_{m+1}$ from the note generator through a transformer decoder with multi-head attention, generating an enhanced representation $\hat{\mathbf{z}}_{m+1} \in \mathbb{R}^d$ infused with global structural information. Compared to direct use of the original latent vector $\mathbf{z}_{m+1}$, the enhanced representation $\hat{\mathbf{z}}_{m+1}$ enhances discriminative features in $\mathbf{z}_{m+1}$ through Self-Attention (SA) mechanism and incorporates global musical context from $\mathbf{z}_1$ into the current prediction through Cross-Attention (CA) mechanism. The computational process is formally defined as:

$$\hat{\mathbf{z}}_{m+1} = \text{CA}\Big(\text{SA}(\mathbf{z}_{m+1}), \mathbf{z}_1, \mathbf{z}_1\Big) \tag{5}$$

where $\text{SA}(\cdot)$ denotes the self-attention mechanism and $\text{CA}(\cdot)$ represents the cross-attention mechanism. Here, $\mathbf{z}_1$ refers to the hidden state of the special `[start-of-note]` token, which functions as a compact conditioning signal summarizing the global prefix context (similar to a learned global token). Alternative global pooling strategies (e.g., mean pooling) are possible extensions.

The optimized representation $\hat{\mathbf{z}}_{m+1}$ is used as a conditioning signal and injected into the discrete diffusion model via cross-attention, guiding the progressive recovery of all masked attributes.

3.3 Loss Function

To simultaneously model the temporal relationships among notes and the complex dependencies among

note attributes. We adopts a masked weighted cross-entropy loss function $\mathcal{L}_{\text{CE}}$ as the optimization objective, defined as follows:

$$\mathcal{L}_{\text{CE}} = -\sum_{k=1}^{K} \frac{1}{p_{\text{mask}}^{k}} \cdot m_k \cdot \sum_{v=1}^{V_k} \mathbb{I}[\mathcal{N}_{m+1}^{k} = v] \log p(v \mid \mathcal{N}_{m+1}^{k}) \tag{6}$$

where p_{mask}^{k} denotes the probability of applying a mask to the k-th attribute during the forward diffusion process. This probability acts as a weighting factor to balance the frequency differences in masking across attributes. $m_k \in \{0, 1\}$ is a binary mask indicator—when $m_k = 1$, it indicates that the corresponding position is non-padded and the attribute is masked. V_k represents the vocabulary size of the k-th attribute. The indicator function $\mathbb{I}[\cdot]$ outputs 1 if the ground-truth value of the k-th attribute at the $m+1$-th note equals v, otherwise 0. The probability term $p(v \mid n_{m+1}^{(k)})$ signifies the model's predicted probability. Through the synergistic mechanism of inverse probability weighting ($1/p_{\text{mask}}^{k}$) and mask filtering (m_k), this loss function focuses exclusively on optimizing prediction errors at effective masked attribute positions.

This loss can be viewed as a standard autoregressive cross-entropy loss computed only over valid masked positions; when all attributes are masked, it is equivalent to the conventional autoregressive cross-entropy loss.

3.4 Training

Fig. 2 illustrates the training processes of Amadeus. The objective of training is to enable the model to learn both the temporal relationships among musical notes and the complex dependencies among attributes within each note. During training, similar to autoregressive models, Amadeus has access to all preceding notes. However, unlike autoregressive approaches that generate note attributes sequentially, Amadeus randomly masks all attributes of the current note and predicts the attributes at the masked positions.

3.5 Inference

Amadeus can accept different masking distributions as input during inference, providing substantial flexibility in the generation process. First, as Fig. 2 illustrates, it enables a wide range of tasks to be performed without additional training. For example, in the standard generation setting, the input can be a sequence of fully masked notes; when control over specific attributes is required, the corresponding attributes can be specified as known values while the remaining attributes are masked, thereby enabling controlled generation with respect to the desired attributes. In addition, the model allows the number of decoding steps at inference time to be adjusted, facilitating a trade-off between generation quality and inference speed. Detailed descriptions are provided in the appendix.

For existing approaches that model a musical note as a sequence of attributes, including both autoregressive and hierarchical autoregressive models, inference is inherently constrained by a predefined attribute ordering. As a result, these methods are unable to perform parallel decoding over attributes during inference. This limitation fundamentally prevents flexible attribute-level control and precludes principled trade-offs between decoding speed and generation quality.

4 Experiment

We collected and utilized the AMD dataset for both pre-training and fine-tuning the model. We conducted a comprehensive and objective evaluation on the text-controlled symbolic music generation task. Training details are provided in the appendix. To ensure a fair comparison and isolate architectural effects, we compared Amadeus with three representative models, *i.e.*, Text2Midi (Bhandari et al., 2025), MuseCoco (Lu et al., 2023), and T2M-InferAlign (Roy et al., 2025), under controlled settings with comparable parameter budgets (Amadeus: 170M; Text2Midi and T2M-InferAlign: 160M; MuseCoco: 200M). We further demonstrate that our model is capable of performing attribute control at inference time. To assess human subjective perception in real-world scenarios, we conducted a human study that comprehensively evaluates the generated results across multiple dimensions. We conducted detailed ablation studies. To explore the performance ceiling of our model, we trained an expanded model Amadeus-M with 500M parameters on AMD dataset.

4.1 AMD Dataset

Pre-training Dataset: By integrating GigaMIDI (Lee et al., 2025), AriaMIDI (Bradshaw and Colton, 2025), SymphonyNet (Liu et al., 2022), MidiCaps(excluding its test set), XMIDI (Tian et al., 2025), and 80,000 self-crawled high-quality symbolic music segments, we constructed a pre-training corpus after data cleaning. This corpus comprises 1.9 million music files and

Tab. 1: Text-conditioned music generation results. Amadeus (170M) and all baseline models are evaluated under comparable parameter budgets. We evaluated the text-condition fidelity, fine-grained note attribute accuracy, and generation speed of the generated music.

	Speed(notes/s)	CLAP↑	TBT↑	CK↑	CTS↑	CI↑	CM_{top3}↑
Text2Midi	4.02	0.19	31.76	22.22	84.15	19.92	60.57
MuseCoco	1.67	0.19	34.21	14.66	94.24	22.42	38.18
T2M-inferalign	4.02	0.20	39.32	29.80	84.32	20.13	47.74
Amadeus	16.23	0.20	73.93	39.31	96.98	26.01	65.52
Amadeus-M	10.51	**0.21**	**76.31**	**43.07**	**97.02**	**27.11**	**66.39**
Amadeus(fix time-sig)	16.81	0.20	73.65	39.41	**100.0**	25.83	65.23
$p(t)$	0.0003	0.0001	0.0002	0.0001	0.0003	0.0002	0.0004

approximately 4 billion music events (equivalent to 32 billion attribute tokens). To our knowledge, this pre-training dataset is currently the largest open-source symbolic music pre-training corpus.

Supervised Fine-tuning Dataset: We integrated MidiCaps, XMIDI and 80,000 self-crawled high-quality symbolic music segments as raw materials. After data cleaning and annotation augmentation, we ultimately constructed a high-quality supervised fine-tuning dataset containing 320,000 samples. All samples include structured textual descriptions. More detailed information on the dataset is available in the appendix.

4.2 Text-Conditioned Symbolic Music Generation

The evaluation system comprises: i) CLAP Score (Wu* et al., 2023; Evans et al., 2024), measuring the similarity between the prompt text and generated audio; and ii) a fine-grained music characteristics metric set, which comprehensively validates the alignment of generated symbolic music with text-specified requirements across five dimensions: Tempo Bin with Tolerance(TBT), Correct Key (CK) (Melechovsky et al., 2024), Correct Time Signature (CTS), Coverage of Instruments (CI), and Coverage of top3 moods (CM_{top3}). We use the average number of notes generated per second (notes/s) to evaluates generation speed. Detailed definitions of all metrics are provided in the supplementary material.

The results(shown in Tab. 1). demonstrate that our method achieves breakthroughs across three critical dimensions—generation quality, control precision, and inference speed—outperforming baseline models in both text-condition fidelity (CLAP Score) and fine-grained metrics, while accelerating inference to 16.23 notes/s (tested on an RTX3090 24G GPU): a 4× speed-up over existing approaches. This comprehensive superiority validates the technical excellence of our method for efficient and controllable symbolic music generation.

Regarding control capability, our model exhibits precise modelling across multiple musical metrics: improvements in TBT and CK reach 88.02% and 31.91%, respectively, proving its robust rhythmic modelling capabilities. A 2.90% improvement in CTS confirms that the attribute-level bidirectional modelling mechanism effectively captures intrinsic harmonic functional relationships. A 16.01% enhancement in CI reflects the model's refined semantic modelling ability, while an 8.17% increase in CM_{top3} highlights its exceptional emotion modelling performance. These quantitative results collectively demonstrate that our method achieves precise modelling and decoding of note attributes through the DDM-Based Note Decoder.

Fig. 3 presents qualitative analysis results for the text-conditioned symbolic music generation task. The text prompt states: "This lengthy electronic piece, infused with a touch of easy listening, is a melodic and relaxing journey. The acoustic guitar takes the lead, accompanied by a string ensemble, piano, shana, and synth strings. Set in A minor and maintaining a 4/4 time signature, it moves at an Allegro tempo of 120 beats per minute. Throughout the composition, the chords E and A alternate frequently, creating a meditative atmosphere with a hint of Christmas spirit and a motivational undertone." The figure displays piano roll visualizations of MIDI files generated by T2M-inferalign (upper panel) and our method (lower panel), respectively. The horizontal axis represents time, each coloured block denotes a note whose length corresponds to its duration and vertical position to its pitch (detailed track-wise visualization is provided in the appendix). Based on a traditional autoregressive

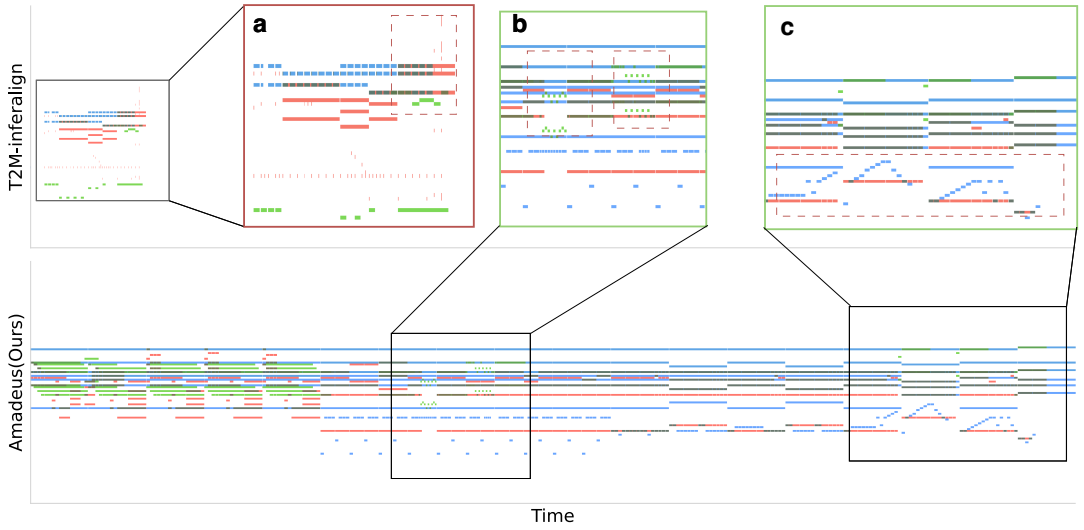

Fig. 3: Comparison of piano roll visualizations of MIDI files generated by T2M-inferalign (upper panel) and our method (lower panel).

architecture for note decoding, T2M-inferalign is constrained by limited long-sequence modelling capability—resulting in fewer generated notes, low note density per time-step, and impoverished variations in pitch-duration combinations. This indicates its inadequacy in generating complex harmonies. Notably, the scattered tomato-coloured short notes in Fig. 3 (a)'s red dashed box further reveal the model's difficulty in producing sustained, musically coherent note sequences. In contrast, the music generated by our method exhibits significantly higher textural complexity and harmonic sophistication. Within the red dashed box in Fig. 3 (b), the main development demonstrates a rhythmic evolution: transitioning from dense long notes to regular green short notes before returning to long notes. This pattern highlights our model's ability to generate intricate rhythmic structures, thereby enriching the auditory experience. The blue melodic line within the red dashed box in Fig. 3 (c) precisely aligns with the prompt's requirement for "melodic". This not only validates our model's exceptional adherence to textual instructions but also underscores its advanced generative quality in melody modelling.

4.3 Attribute Control at Inference Time

Tab. 1 also reports the results of a controlled generation experiment in which the time signature is fixed to the value specified in the text prompt, while all other attributes are generated by the model. Under this predefined control setting, the model achieves a perfect score of 100% on the CTS metric, further validating its ability to perform fine-grained, training-free control over note-level attributes at inference time. Meanwhile, the model maintains performance on other evaluation metrics comparable to those obtained under the uncontrolled generation setting, demonstrating that attribute control can be achieved without sacrificing generation quality or text–condition consistency.

4.4 Human Listening Study

We recruited 20 participants from diverse backgrounds to conduct the listening test. For each participant, five text prompts were randomly selected along with the corresponding text–MIDI pairs generated by different models (Amadeus, T2M-InferAlign, MuseCoco, and Text2Midi). All MIDI files were rendered into audio using identical synthesis settings. To clearly differentiate between core musical attributes and instruction-following capability, we categorize the evaluation into two groups. **Musicality Metrics** include (1) musical quality (MQ), evaluating the naturalness and musical coherence of melody, rhythm, and harmony. **Controllability Metrics** assess semantic alignment, including (2) overall match (OM) for global semantic consistency, (3) genre match (GM) checking if the style corresponds to the prompt, and (4) mood match (MM) judging emotional expression adherence.

As shown in Table 2, Amadeus consistently outperforms the strong baseline T2M-InferAlign, as well as MuseCoco and Text2Midi, across all evaluation dimensions. In terms of musicality,

Tab. 2: Results of the human listening study with 20 participants. Each sample was rated on a 5-point Likert scale. $\mathbf{p(t)}$ denotes the p-value from paired t-tests comparing Amadeus with the strongest baseline (T2M-InferAlign).

	Amadeus	T2M-InferAlign	MuseCoco	Text2Midi	$\mathbf{p(t)}$
MQ↑	3.82	3.47	3.32	3.45	0.0023
OM↑	3.68	3.12	3.03	3.05	0.0001
GM↑	3.80	3.32	3.17	3.25	0.0002
MM↑	3.73	2.96	2.78	2.83	0.0001

Tab. 3: Ablation study of CIEM and denoising steps on text-conditioned symbolic music generation.

CIEM	Steps	Speed(notes/s)	CLAP↑	TBT↑	CK↑	CTS↑	CI↑	CM_{top3}↑
×	8	16.24	0.17	70.01	38.76	94.33	23.79	62.35
√	1	32.12	0.19	59.84	28.19	89.30	20.08	37.93
√	4	20.11	0.20	66.46	39.00	**97.38**	25.80	**67.40**
√	8	16.23	**0.20**	**73.93**	**39.31**	96.98	**26.01**	65.52
$p(t)$		0.0001	0.0002	0.0001	0.0003	0.0002	0.0001	0.0003

Amadeus achieves higher MQ scores (3.82 vs. 3.47 for T2M-InferAlign, $p = 0.0023$), indicating more coherent and musically plausible generations compared to all baselines. Furthermore, the improvements are substantial in controllability metrics, including overall semantic consistency (3.68 vs. 3.12, $p = 0.0001$), genre matching (3.80 vs. 3.32, $p = 0.0002$), and mood matching (3.73 vs. 2.96, $p = 0.0001$). The statistically significant p-values across all metrics indicate that the observed improvements are robust rather than arising from random variation. We also evaluated inter-rater reliability among the 20 participants and observed a substantial level of agreement (e.g., Fleiss' $\kappa = 0.68$), further validating the reliability of human evaluations.

4.5 Ablation Study

To deeply investigate the impact of CIEM on generation quality, we designed systematic ablation experiments. As shown in the Tab. 4: i) CIEM significantly improves the quality of generated symbolic music. This is because CIEM introduces global information into the DDM based note decoder, which greatly ensures the continuity of the generated symbolic music sequence. Without CIEM, the overall quality of the generated audio remains poor. This proves that the CIEM module is indispensable. ii) As the number of denoising steps decreases, the generation speed significantly increases (up to 32.12 notes/s), but the model performance generally exhibits a declining trend. The iterative parallel decoding process enables the model to fully integrate currently decoded information and gradually optimize generation results—increasing steps enhances quality at the cost of speed; iii) Higher steps do not always correspond to better performance. For example, when CIEM is enabled, the model achieves superior results on CTS and CM_{top3}metrics when step is 4. The core reason is that the confidence-based retention mechanism at each step resembles a local greedy search. Moderately reducing steps helps the model escape local optima to achieve globally superior solutions; iv) Whether pursuing the highest quality (where Amadeus operates at 16.23 notes/s, significantly outperforming the comparative methods' 4.02 notes/s) or prioritizing maximum speed (where generation quality is lowest but still comparable to the comparative methods), Amadeus demonstrates comprehensive superiority.

4.6 Effects of Dataset Scale and Parameter Scale

We adopted a two-stage training strategy: pre-training on the AMD pre-training dataset, followed by supervised fine-tuning on the fine-tuning dataset. To further enhance model capability, we scaled up the model size, denoted as Amadeus-M (500M parameters). As shown in the Tab. 1, Amadeus-M achieves significant improvements across all performance metrics in the text-controlled music generation task. Notably, the increased model parameter count resulted in a decreased generation speed of 10.51 notes/s, yet this speed remained significantly higher than the 4.02 notes/s of other comparative methods. Experiments verified that its performance can be further enhanced with continuously increasing data volume and parameter count.

5 Conclusion

In this paper, we challenge the conventional view of musical note attributes as fixed unidirectional sequences, proposing instead that they are more accurately modeled as sets. This insight led to the development of Amadeus, a hybrid framework that synergistically combines autoregressive modeling for global note sequences with bidirectional discrete diffusion for local attribute generation. To bolster this architecture, we introduced the Conditional Information Enhancement Module (CIEM) and curated **AMD**, currently the largest open-source symbolic music dataset. Our extensive evaluations demonstrate that Amadeus not only achieves state-of-the-art performance across objective and subjective metrics but also offers unprecedented flexibility in attribute control and inference efficiency. By bridging the gap between structural coherence and attribute-level flexibility, Amadeus sets a new benchmark for controllable symbolic music generation and opens promising avenues for future research in hierarchical generative modeling.

6 Limitations

Despite its performance, our study has several limitations. First, although the AMD dataset is extensive, it may exhibit stylistic biases toward Western classical and popular music due to its web-crawled nature, potentially underrepresenting diverse ethnic or avant-garde genres. Second, our subjective evaluation, while statistically significant, relied on a relatively small group of 15 participants; a larger and more specialized cohort (e.g., professional composers) would provide more robust validation of musical artistry. Lastly, like many large-scale generative models, Amadeus faces potential "memory effects" where generated sequences might inadvertently resemble training samples. The model currently lacks an automated mechanism to detect and mitigate such copyright and ethical risks, which remains a focus for future work.

References

Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. 2023. Musiclm: Generating music from text. Cite arxiv:2301.11325Comment: Supplementary material at https://google-research.github.io/seanet/musiclm/examples and https://kaggle.com/datasets/googleai/musiccaps.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993.

Keshav Bhandari, Abhinaba Roy, Kyra Wang, Geeta Puri, Simon Colton, and Dorien Herremans. 2025. Text2midi: Generating symbolic music from captions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23478–23486.

Jamshed Bharucha and Carol L. Krumhansl. 1983. The representation of harmonic structure in music: Hierarchies of stability as a function of context. *Cognition*, 13(1):63–102.

Louis Bradshaw and Simon Colton. 2025. Aria-midi: A dataset of piano midi files for symbolic music modeling. *Preprint*, arXiv:2504.15071.

Tom Collins, Robin Laney, Alistair Willis, and Paul H Garthwaite. 2011. Chopin, mazurkas and markov: Making music in style with statistics. *Significance*, 8(4):154–159.

David Cope. 2000. *The algorithmic composer*, volume 16. AR Editions, Inc.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. 2020. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*.

Hao-Wen Dong, Ke Chen, Shlomo Dubnov, Julian McAuley, and Taylor Berg-Kirkpatrick. 2023. Multitrack music transformer. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Zach Evans, Julian D. Parker, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons. 2024. Stable audio open. *Preprint*, arXiv:2407.14358.

Zixun Guo and Simon Dixon. 2025. Moonbeam: A midi foundation model using both absolute and relative music attributes. *Preprint*, arXiv:2505.15559.

Lejaren Arthur Hiller and Leonard M Isaacson. 1957. *Experimental Music; Composition with an electronic computer*. Greenwood Publishing Group Inc.

Wen-Yi Hsiao, Jen-Yu Liu, Yin-Cheng Yeh, and Yi-Hsuan Yang. 2021. Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 178–186.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. 2018. Music transformer. *arXiv preprint arXiv:1809.04281*.

Yu-Siang Huang and Yi-Hsuan Yang. 2020. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, page 1180–1188, New York, NY, USA. Association for Computing Machinery.

Jongmin Jung Jiwoo Ryu, Hao-Wen Dong and Dasaem Jeong. 2024. Nested music transformer. In *25th International Society for Music Information Retrieval Conference (ISMIR)*.

Keon Ju Maverick Lee, Jeff Ens, Sara Adkins, Pedro Sarmento, Mathieu Barthet, and Philippe Pasquier. 2025. The gigamidi dataset with features for expressive music performance detection. *Transactions of the International Society for Music Information Retrieval*, 8(1).

Jiafeng Liu, Yuanliang Dong, Zehua Cheng, Xinran Zhang, Xiaobing Li, Feng Yu, and Maosong Sun. 2022. Symphony generation with permutation invariant language model. *Preprint*, arXiv:2205.05448.

Peiling Lu, Xin Xu, Chenfei Kang, Botao Yu, Chengyi Xing, Xu Tan, and Jiang Bian. 2023. Musecoco: Generating symbolic music from text. *Preprint*, arXiv:2306.00110.

Jan Melechovsky, Zixun Guo, Deepanway Ghosal, Navonil Majumder, Dorien Herremans, and Soujanya Poria. 2024. Mustango: Toward controllable text-to-music generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8286–8309.

Olof Mogren. 2016. C-rnn-gan: A continuous recurrent neural network with adversarial training. In *Constructive Machine Learning Workshop (CML) at NIPS 2016*, page 1.

Eita Nakamura, Philippe Cuvillier, Arshia Cont, Nobutaka Ono, and Shigeki Sagayama. 2015. Autoregressive hidden semi-markov model of symbolic music performance for score following. In *16th International Society for Music Information Retrieval Conference (ISMIR)*.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2025a. Scaling up masked diffusion models on text. *Preprint*, arXiv:2410.18514.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025b. Large language diffusion models. *arXiv preprint arXiv:2502.09992*.

Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srinivasan Iyer. 2024. Byte latent transformer: Patches scale better than tokens. *Preprint*, arXiv:2412.09871.

Matthias Plasser, Silvan Peter, and Gerhard Widmer. 2023. Discrete diffusion probabilistic models for symbolic music generation. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, IJCAI '23.

Xingwei Qu, Yuelin Bai, Yinghao Ma, Ziya Zhou, Ka Man Lo, Jiaheng Liu, Ruibin Yuan, Lejun Min, Xueling Liu, Tianyu Zhang, and 1 others. 2024. Mupt: A generative symbolic music pretrained transformer. *arXiv preprint arXiv:2404.06393*.

Colin Raffel. 2016. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. Ph.D. thesis, Columbia University, USA.

Abhinaba Roy, Geeta Puri, and Dorien Herremans. 2025. Text2midi-inferalign: Improving symbolic music generation with inference-time alignment. *arXiv preprint arXiv:2505.12669*.

Sida Tian, Can Zhang, Wei Yuan, Wei Tan, and Wenjie Zhu. 2025. Xmusic: Towards a generalized and controllable symbolic music generation framework. *IEEE Transactions on Multimedia*.

Yashan Wang, Shangda Wu, Jianhuai Hu, Xingjian Du, Yueqi Peng, Yongxin Huang, Shuai Fan, Xiaobing Li, Feng Yu, and Maosong Sun. 2025. Notagen: Advancing musicality in symbolic music generation with large language model training paradigms. *Preprint*, arXiv:2502.18008.

Shangda Wu, Yashan Wang, Xiaobing Li, Feng Yu, and Maosong Sun. 2024. Melodyt5: A unified score-to-score transformer for symbolic music processing. *Preprint*, arXiv:2407.02277.

Yusong Wu*, Ke Chen*, Tianyu Zhang*, Yuchen Hui*, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. 2023. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*.

Iannis Xenakis. 1992. *Formalized music: thought and mathematics in composition*. 6. Pendragon Press.

Lili Yu, Dániel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. 2023. Megabyte: Predicting million-byte sequences with multiscale transformers. *Advances in Neural Information Processing Systems*, 36:78808–78823.

A Mutual Information Analysis of Token Attributes

To assess the dependency structure among the attributes, we conducted a mutual information (MI) and conditional entropy analysis. This empirical study is intended to inform the modeling strategy of musical event sequences, particularly the suitability of autoregressive vs. bidirectional generation paradigms.

A.1 MIDI Protocol, Tokenization Method, and Note Attributes

The Musical Instrument Digital Interface (MIDI) is an industry-standard electronic communication protocol that defines a comprehensive set of codes for musical notes and performance actions, enabling electronic instruments, computers, mobile devices, and other stage equipment to connect, synchronize, and exchange performance data in real time. MIDI provides a complete framework for describing the performance states of different instruments and plays a crucial role in modern music composition workflows. Due to the flexibility of MIDI signals, artists often use Digital Audio Workstations (DAWs) to rapidly iterate on musical ideas by editing MIDI data and assigning virtual instruments. Once the desired effect is achieved, the final version may be recorded with real instruments. This workflow significantly lowers the barrier to music production and democratizes music creation.

There are various methods for converting MIDI signals into tokens. We recommend interested readers refer to the `miditok` library, which provides a detailed introduction to specific tokenization processes. In our work, we adopt an improved note-based encoding proposed by NMT. Specifically, each note is represented by the following attributes:

- **Type**: each representing a different combination of metrical changes or continuations.
- **Beat**: The relative position of the note within the same measure.
- **Chord**: The chord to which the current note belongs.
- **Tempo**: The playback speed of the note; generally, a higher tempo indicates a faster song.
- **Instrument**: The instrument performing the current note.
- **Pitch**: The pitch of the note, represented by 128 discrete values according to the MIDI specification.
- **Duration**: The duration for which the note is played.
- **Velocity**: The intensity with which the note is played, determining its loudness.

These attributes uniquely specify a note. Importantly, each attribute describes an independent aspect of the note, and the set of attributes is inherently free from unidirectional sequential dependencies. Technically, the order of these attributes can be arbitrarily permuted without affecting the representation of the note itself, which clearly goes beyond the scope of unidirectional dependency. As observed in prior work, different methods may use different attributes (such as `type` or `pitch`) as the first token, yet achieve comparable performance. This further demonstrates the insufficiency of assuming unidirectional dependencies among note attributes.

A.2 Experimental Setup

We tokenized corpus of LakhClean dataset MIDI files into notes, from which we extracted the following attribute dimensions: `Beat Position`, `Pitch`, `Velocity`, `Duration`, `Instrument`, `Chord`, `Tempo`, and `Type`.

The total amount of note tokens in the dataset is 82985704. For each attribute, we applied label encoding and computed the empirical mutual information between all attribute pairs:

$$I(X;Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

We further normalized MI using the geometric mean of marginal entropies to obtain a symmetric measure in $[0, 1]$:

$$\mathrm{NMI}(X;Y) = \frac{I(X;Y)}{\sqrt{H(X)H(Y)}}$$

A.3 Results and Observations

The normalized mutual information matrix $\mathrm{NMI}(X;Y)$ reveals several trends. First, dependencies among `Pitch`, `Velocity`, `Duration`, and `Instrument` are generally weak to moderate (for example, $\mathrm{NMI}(\texttt{Pitch}, \texttt{Instrument}) \approx 0.28$). Such complex dependencies can be explained by basic musical knowledge: different instruments

Tab. 4: Raw Mutual Information (nats)

	Beat	Pitch	Velocity	Duration	Instrument	Chord	Tempo	Type
Beat	1.1703	0.5128	0.5128	0.5128	0.5128	0.0008	0.4905	0.0964
Pitch	0.5128	3.5981	0.5308	0.5401	0.8679	0.0007	0.4205	0.0314
Velocity	0.5128	0.5308	1.6784	0.5222	0.5356	0.0007	0.4205	0.0314
Duration	0.5128	0.5401	0.5222	2.4884	0.5961	0.0007	0.4205	0.0314
Instrument	0.5128	0.8679	0.5356	0.5961	2.6845	0.0007	0.4205	0.0314
Chord	0.0008	0.0007	0.0007	0.0007	0.0007	0.0046	0.0008	0.0000
Tempo	0.4905	0.4205	0.4205	0.4205	0.4205	0.0008	1.0186	0.0042
Type	0.0964	0.0314	0.0314	0.0314	0.0314	0.0000	0.0042	0.1040

Tab. 5: Normalized Mutual Information

	Beat	Pitch	Velocity	Duration	Instrument	Chord	Tempo	Type
Beat	1.0000	0.2499	0.3659	0.3005	0.2893	0.0114	0.4492	0.2764
Pitch	0.2499	1.0000	0.2160	0.1805	0.2793	0.0056	0.2197	0.0513
Velocity	0.3659	0.2160	1.0000	0.2555	0.2523	0.0082	0.3216	0.0752
Duration	0.3005	0.1805	0.2555	1.0000	0.2306	0.0068	0.2641	0.0617
Instrument	0.2893	0.2793	0.2523	0.2306	1.0000	0.0065	0.2543	0.0594
Chord	0.0114	0.0056	0.0082	0.0068	0.0065	1.0000	0.0120	0.0004
Tempo	0.4492	0.2197	0.3216	0.2641	0.2543	0.0120	1.0000	0.0128
Type	0.2764	0.0513	0.0752	0.0617	0.0594	0.0004	0.0128	1.0000

have distinct pitch ranges and playing styles, which influence the distribution of these attributes. The strongest dependency is observed between `Beat Position` and `Tempo` (NMI ≈ 0.45), reflecting the rhythmic regularity commonly found in MIDI data.

We further analyzed asymmetric conditional entropy to evaluate directional predictability:

$$H(Y|X) = H(Y) - I(X;Y)$$

The estimated values:

$$H(\texttt{Duration}|\texttt{Pitch}) \approx 1.95,$$
$$H(\texttt{Pitch}|\texttt{Duration}) \approx 3.06$$

indicate that knowing `Pitch` reduces uncertainty in `Duration` more effectively than the reverse, suggesting a directional statistical influence from `Pitch` to `Duration`.

A.4 Implications for Modeling

The results indicate that the dependencies among note attributes are generally weak and not strictly unidirectional, as evidenced by the low maximum NMI values and the absence of any attribute that strongly determines the others. Furthermore, the observed dependencies are often mutual rather than hierarchical, suggesting that imposing a fixed generation order, as in conventional autoregressive (AR) models, may not be optimal. These findings support the adoption of bidirectional or non-autoregressive modeling strategies, which are better suited to capture the symmetrical and overlapping relationships among token attributes in symbolic music data.

B Training and Inference Details

B.1 Training Setup

In our training setup, we employ gradient clipping with a threshold of 1 and utilize the AdamW optimizer in conjunction with a cosine learning rate scheduler. For unconditional generation, the maximum learning rate is set to 1×10^{-4}; for conditional generation and large-scale pretraining, it is 2×10^{-4}; and for text-guided fine-tuning, it is 5×10^{-5}. The maximum number of training iterations is 100,000 for both unconditional and conditional generation tasks, and 300,000 for pretraining. Notably, we do not use the final checkpoint from pretraining, but rather select an intermediate checkpoint for subsequent experiments. Training is accelerated using distributed data parallelism (DDP) and the `accelerate` library with mixed-precision computation. For conditional generation, pretraining, and fine-tuning, we further employ gradient accumulation with a step size of 4.

B.2 Pretraining Data

Our pretraining dataset mainly combines the GigaMIDI, AriaMIDI, SymphonyNet corpora. The total number of files is 1.9 million, comprising about 4 billion events (approximately 32 billion attribute tokens). To our knowledge this is the largest open-source symbolic music pretraining dataset. To enhance data quality, we filter out GigaMIDI files that contain only drum tracks. Additionally, due to the relatively poor quality of some pretraining data, we also incorporate the MIDI files from the fine-tuning datasets into the pretraining collection. We do not include the MidiCaps test split in training to ensure fairness.

We follow a processing procedure partly inspired by NMT: first, we apply rule-based filtering to the MIDI files and detect attributes such as chords, time signatures, note durations, and instruments. We convert these attributes into event sequences, compile a vocabulary of tokens for each attribute, and then convert the event sequences into index files corresponding to this vocabulary. Notably, for any file missing tempo or time signature information, we set the default tempo to 120 BPM and the default time signature to 4/4, in accordance with the MIDI specification.

B.3 Supervised Fine-Tuning Data

For the supervised (fine-tuning) dataset with text annotations, the processing differs from that of the pretraining data. We use the MidiCaps and XMIDI datasets, as well as an in-house dataset, as the raw corpora. XMIDI contains around 100k symbolic music pieces annotated by experts with emotion and genre labels, and our in-house dataset includes roughly 80k high-quality symbolic music pieces.

The overall processing pipeline is divided into two stages: text annotation and conversion to index sequences.

In the text annotation stage, we largely follow the MidiCaps methodology. We first extract information from the MIDI sequences directly using tools such as Music21 and Mido, capturing key (mode), time signature, tempo, durations, and instruments. We then synthesize each MIDI file into audio and extract features such as genre, mood, and chords. Based on the extracted information, we design prompt templates with examples and use DeepSeek-v3 to convert these prompts into textual annotations. For XMIDI, we simply use its provided emotion and genre labels as annotations.

In the second stage, we convert each piece into index files similarly to the pretraining data, with one major difference: we do not perform any instrument merging or trimming. Instrument merging (e.g., combining distorted guitar, overdrive guitar, and clean electric guitar into a single "electric guitar" category) is sometimes used to reduce sequence length, but it can severely harm the alignment between the generated music and the text control signal in the conditional generation task. Therefore, following the approach of `text2midi`, we assign each instrument to its own track to avoid merging. After this two-stage processing, we obtain a high-quality supervised symbolic music dataset of 320k examples with text annotations.

B.4 Text-MIDI Fusion via Cross-Attention

To effectively align linguistic descriptions with symbolic music signals, we employ a cross-attention mechanism that treats the text embedding as the "context" for MIDI generation. This ensures that the generated musical attributes are strictly conditioned on the input prompt.

B.4.1 Architectural Implementation

We use Flan-T5 as text encoder. The fusion process is integrated into the Transformer-based decoder layers. Given the text prompt's hidden states H_{text} and the MIDI sequence's latent representations H_{midi}, the cross-attention is defined as:

$$\text{CrossAttn}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{7}$$

In this setup:

- **Query (Q):** Derived from the MIDI latent states H_{midi}, representing the musical tokens currently being generated.
- **Key (K) and Value (V):** Derived from the text encoder's output H_{text}, providing the global linguistic constraints.

B.4.2 Modality Alignment

By calculating the attention scores between each MIDI token and all text tokens, the model can "attend" to specific keywords (e.g., "fast tempo") while generating corresponding musical events. This mechanism allows for long-range dependency modeling between the two modalities, which is crucial for the AMD dataset's 320,000 fine-tuning samples.

B.5 Evaluation Metrics

We use the MidiCaps test set of 500 text annotations to guide the generation of symbolic music, render the outputs to WAV, and trim them to 10 seconds (again generating 1024 tokens per sample). We evaluate from two perspectives: overall similarity between the prompt text and the generated audio, and accuracy of the conditioned attributes.

Overall Text–Audio Similarity We use the CLAP score, which measures the cosine similarity between text and audio embeddings, bigger CLAP score indicates better alignment between the text and audio. We follow the implementation of stable-audio.

Control Accuracy Metrics The text annotation in MidiCaps includes attributes such as key, time signature, tempo, and instruments. Following previous works, We evaluate the accuracy of these attributes in the generated symbolic music using the following metrics:

- **Tempo Bin with Tolerance (TBT)**: the ratio of samples where the predicted tempo falls within a tolerance range of 10 BPM from the true tempo. This metric accounts for slight variations in tempo that may still be musically acceptable.
- **Correct Key (CK)**: the ratio of samples where the predicted key matches the true key, ignoring the octave. This metric evaluates the model's ability to capture the tonal center of the music. By convention, an undefined key is treated as C major.
- **Correct Time Signature (CTS)**: represents the ratio of generated MIDI files where the predicted time signature matches the true time signature. This metric evaluates the model's ability to capture the rhythmic structure of the music.
- **Coverage of Instruments (CI)** is the fraction where predicted instruments fully cover the ground truth; $\mathrm{CI}_{\mathrm{top1}}$ is the fraction where at least one true instrument appears in the prediction.
- **Coverage of Mood (CM)**: the fraction of samples where the predicted mood matches the true mood. This metric evaluates the model's ability to capture the emotional content of the music;$\mathrm{CM}_{\mathrm{top3}}$ is the fraction where at least three true moods appear in the prediction.(all mood if the number of moods is less than 3)

B.6 Step-Adjustable Note Decoding

The forward process of our note decoding framework adheres to the standard pipeline of discrete diffusion models (Nie et al., 2025b). Its core operation involves applying time-step-dependent independent masking operations to each note attribute (treated as an independent token) in the sequence. The reverse process initiates from a partially masked sequence (during training) or a fully masked sequence (during inference). After the note latent vector $\mathbf{z}_m$ generated by the note generator is processed by the Conditional Information Enhancement Module (CIEM) to obtain the enhanced conditional signal $\hat{\mathbf{z}}_m$, this signal serves as a conditioning input to guide the progressive recovery of all masked attributes via cross-attention mechanisms. At each time-step t (total steps T), the system precisely computes the number of tokens $\mathtt{num}_{\mathrm{tk}}$ to decode using an approximate uniform recovery schedule:

$$\mathtt{num}_{\mathrm{tk}} = \begin{cases} \left\lfloor \frac{\mathtt{num}_{\mathtt{m}}}{T} \right\rfloor + 1, & \text{if } t < \mathtt{num}_{\mathtt{m}} \bmod T \\ \left\lfloor \frac{\mathtt{num}_{\mathtt{m}}}{T} \right\rfloor, & \text{otherwise} \end{cases} \tag{8}$$

where $\mathtt{num}_{\mathtt{m}}$ is the total number of masked tokens in the note. For a masked attribute k at time step t, the model predicts a probability distribution $p(v \mid \hat{\mathbf{z}}_{m+1}, x_t^k)$ over the vocabulary V_k. The confidence score for this prediction is defined as the maximum softmax probability:

$$\text{Confidence} = \max_v p(v \mid \hat{\mathbf{z}}_{m+1}, x_t^k) \tag{9}$$

During each decoding step, the model selects the $\mathtt{num}_{\mathrm{tk}}$ tokens with the highest confidence scores, fixes their attributes for all subsequent iterations, and re-masks all remaining tokens.

Fig. 4 visually illustrates the flexible step-adjustable decoding mechanism of our method during the reverse process. Using a sequence containing three notes (each with four attributes) as an example (the horizontal axis denotes token sequence length; the vertical axis indicates decoding iterations—higher rows correspond to more iterations/longer latency): REMI (Huang and Yang, 2020) and NMT (Jiwoo Ryu and Jeong, 2024) prevent parallel attribute decoding by enforcing sequential constraints. Each attribute is decoded one

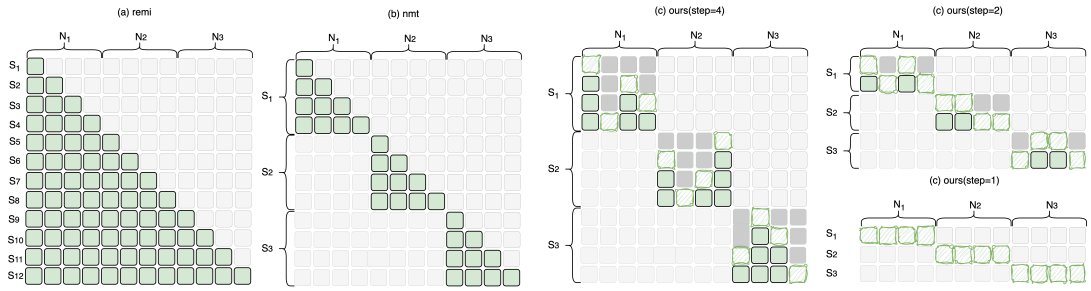

Fig. 4: Comparative schematics of note decoding processes: (a) REMI; (b) NMT; (c-e) Our step-adjustable note decoding (step=4, 2, 1).

at a time, requiring exactly 12 iterations for 3 notes and 4 attributes per note. In contrast, our method eliminates inter-attribute sequential dependencies, leveraging discrete diffusion to enable attribute-parallel decoding while permitting flexible control over total steps T (where T also adjusts $\texttt{num}_{\text{tk}}$). By setting different T values: $T = 4$ (more steps), $T = 2$, or $T = 1$ (fewest steps), overall decoding latency can be drastically altered, thereby achieving an effective trade-off between decoding speed and quality.

Figure 4 visually illustrates the flexible step-adjustable decoding mechanism of our method during the reverse process. Using a sequence containing three notes (each with four attributes) as an example (the horizontal axis denotes token sequence length; the vertical axis indicates decoding steps—higher rows correspond to more steps/longer latency): REMI requires exactly 12 iterations (3 notes × 4 attributes) to complete all attributes due to its global sequence dependencies, decoding only one attribute per step. Similarly, NMT relies solely on intra-note attribute dependencies but still decodes one attribute per step, requiring 12 iterations. Both methods enforce strict sequential constraints among attributes, preventing parallel decoding. In contrast, our method eliminates inter-attribute sequential dependencies, leveraging discrete diffusion to enable attribute-parallel decoding while permitting flexible control over total steps T (where T also adjusts `num_tk`). By setting different T values: $T = 4$ (more steps), $T = 2$, or $T = 1$ (fewest steps), overall decoding latency can be drastically altered, thereby achieving an effective trade-off between decoding speed and accuracy.

C Human Evaluation Details

C.1 Evaluation Interface and Platform

We developed a web-based evaluation platform to conduct the subjective listening tests. The interface was designed to be intuitive, ensuring that participants could focus entirely on musical quality without technical distractions. As shown in Figure 5, the platform presents a text prompt and two randomized audio clips (Audio 1 and Audio 2) to eliminate sequence bias. All clips were rendered from MIDI files using a consistent, high-quality synthesizer to ensure that differences in audio quality did not interfere with the evaluation of the underlying symbolic music.

C.2 Participants and Recruitment

We recruited 15 participants for this study. The cohort primarily consisted of university students with varying levels of musical background.

- **Demographics:** Approximately **[30%]** of participants identified as having formal musical training (e.g., instrument proficiency or music theory knowledge), while the remainder were general listeners.
- **Compensation:** Participants were compensated at a rate of half an hour. On average, the evaluation session lasted 10 minutes.

C.3 Evaluation Criteria (Translated)

The interface was presented in Chinese to accommodate the primary language of the participants. The participants were asked to rate the music on a 5-point Likert scale (1: Lowest to 5: Highest) based on the following four dimensions (translated from the interface shown in Figure 5):

1. **Musical Quality:** Whether the melody, rhythm, and harmony are natural and exhibit musicality.

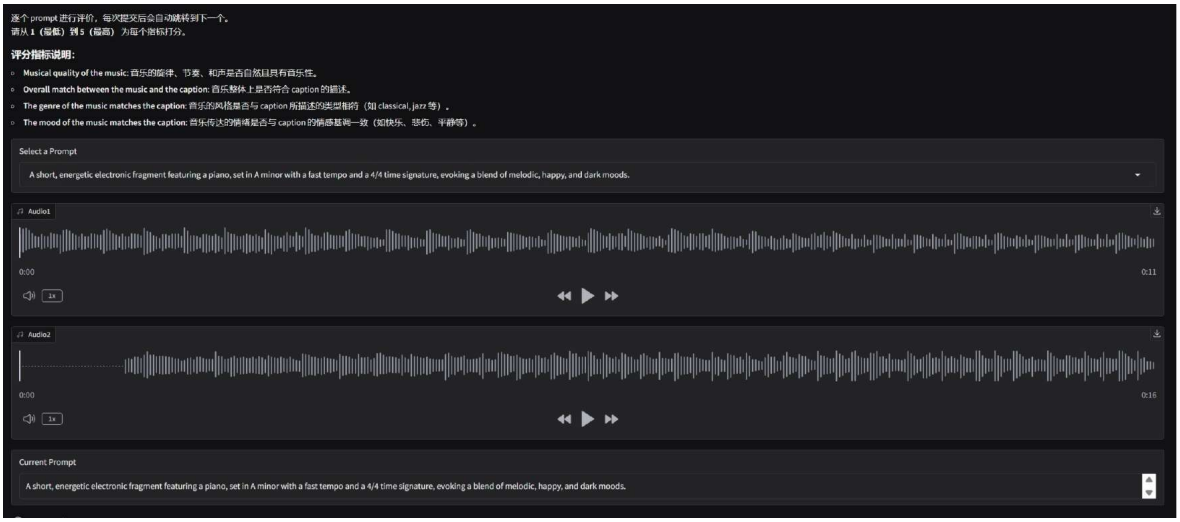

Fig. 5: The web interface used for the human study. Participants evaluate generated music based on specific text prompts across four standardized metrics.

2. **Overall Match:** Whether the generated music aligns with the overall description of the text prompt.

3. **Genre Correspondence:** Whether the musical style (e.g., classical, jazz) matches the genre specified in the prompt.

4. **Mood Consistency:** Whether the emotional tone (e.g., happy, dark) matches the mood described in the prompt.

C.4 Consent and Ethical Compliance

At the beginning of each session, an Informed Consent Form was presented to the participants. The form detailed: (1) the research purpose; (2) the complete anonymization of ratings and demographic data; (3) the voluntary nature of participation with the right to withdraw without penalty; and (4) the strict absence of Personally Identifying Information (PII) storage.

Following local institutional policy, this study was determined to be **exempt from formal IRB approval** due to its anonymous nature, minimal risk to participants, and focus on subjective artistic evaluation.

Tab. 6: Unconditional music generation results on the LakhClean dataset and SOD dataset.

	LakhClean			SOD		
Model	SC ↑	PE ↓	PCE ↓	SC ↑	PE ↓	PCE ↓
CPWord	0.93	4.19	2.64	0.90	4.29	2.89
MMT	0.94	4.09	2.61	0.91	4.38	2.90
NMT	0.91	4.36	2.82	0.91	4.32	2.93
REMI	0.95	3.45	2.38	0.90	4.60	3.05
Amadeus	**0.97**	**2.20**	**1.97**	**0.92**	**4.24**	**2.82**

D Additional Results

D.1 Unconditional Symbolic Music Generation

For evaluation, we use metrics from both the audio domain and the symbolic domain to comprehensively assess model performance. In the unconditional generation task, for each model we generate 500 samples of a fixed length of 1024 tokens, render them to WAV audio, and uniformly trim each audio clip to 30 seconds.

We introduce a set of symbolic-domain metrics to comprehensively evaluate the quality and consistency of generated symbolic music. These include:

- `scale_consistency`: Measures the maximum proportion of notes that fit within any assumed musical scale, reflecting the overall tonal coherence of the generated piece.

- `pitch_entropy`: The Shannon entropy of the

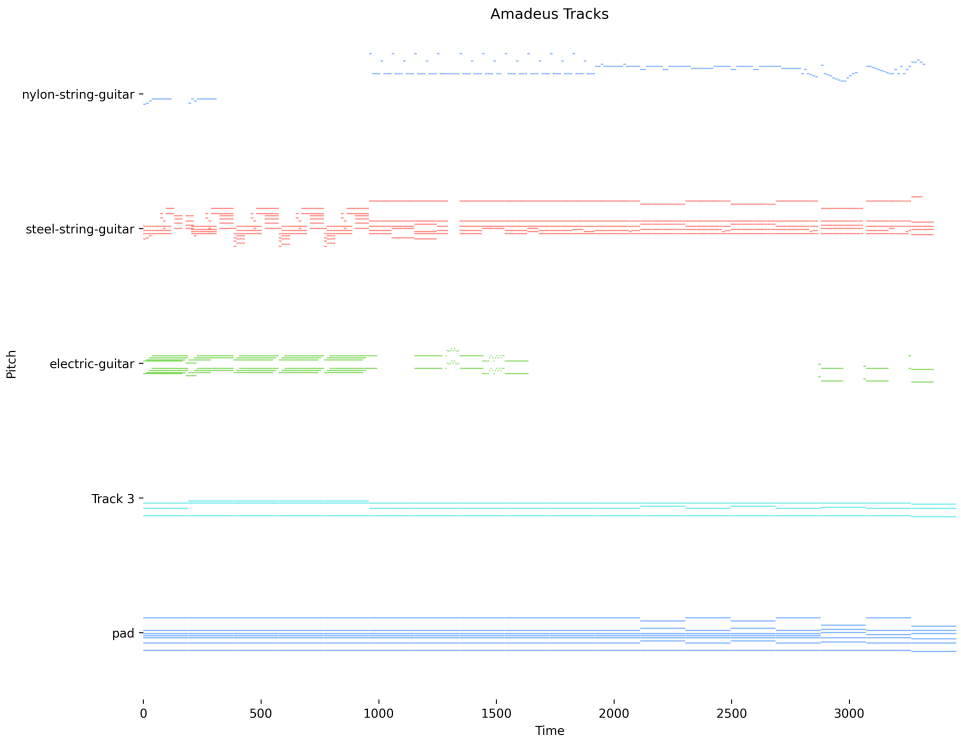

Fig. 6: Visualization of generated symbolic music by Amadeus.

pitch histogram, quantifying the diversity and concentration of pitch usage. Lower entropy indicates that pitches are concentrated on a few core notes, suggesting clearer tonality and higher generation quality.

- `pitch_class_entropy`: The Shannon entropy of the pitch-class histogram, further characterizing the stability of pitch usage across pitch classes.

We compute these metrics using the methods provided by MusPy.

The unconditional symbolic music generation results shown in Tab. 6 demonstrates that: i) The model achieves optimal performance across all evaluation metrics on both datasets—demonstrating significant superiority on the multi-genre Lakh Clean dataset while also exhibiting exceptional performance on the SOD dataset, which primarily features orchestral and classical music. Given that orchestral and classical music demand stronger fine-grained attribute modelling and long-range structural modelling capabilities, these results fully confirm that our model significantly outperforms traditional autoregressive and other baseline models in capturing the intrinsic development of symbolic music; ii) The highest SC score indicates that the model possesses more stable structural coherence when generating long-sequence music; iii) Significant advantages in PE and PCE further corroborate the model's precise modelling ability for harmonic complexity, with the generated music surpassing baseline methods in both harmonic richness and quality.

E Visualization of Generated Music

To provide a visual representation of the generated symbolic music, we use the `pypianoroll` library to render the MIDI files into sheet music. Below are examples of generated tracks from both Amadeus and T2M-inferalign, showcasing the diversity and complexity of the outputs.

F Environmental Impact

To reduce carbon emissions and make better use of computational resources, we employed mixed-precision training with the `accelerate` library. Compared to standard Distributed Data Parallel (DDP) training under the same power consumption, this setup achieved nearly 40% faster training. This efficiency gain indirectly mitigates the environmental impact. We plan to explore additional energy-efficient training strategies in future work to further reduce environmental costs.

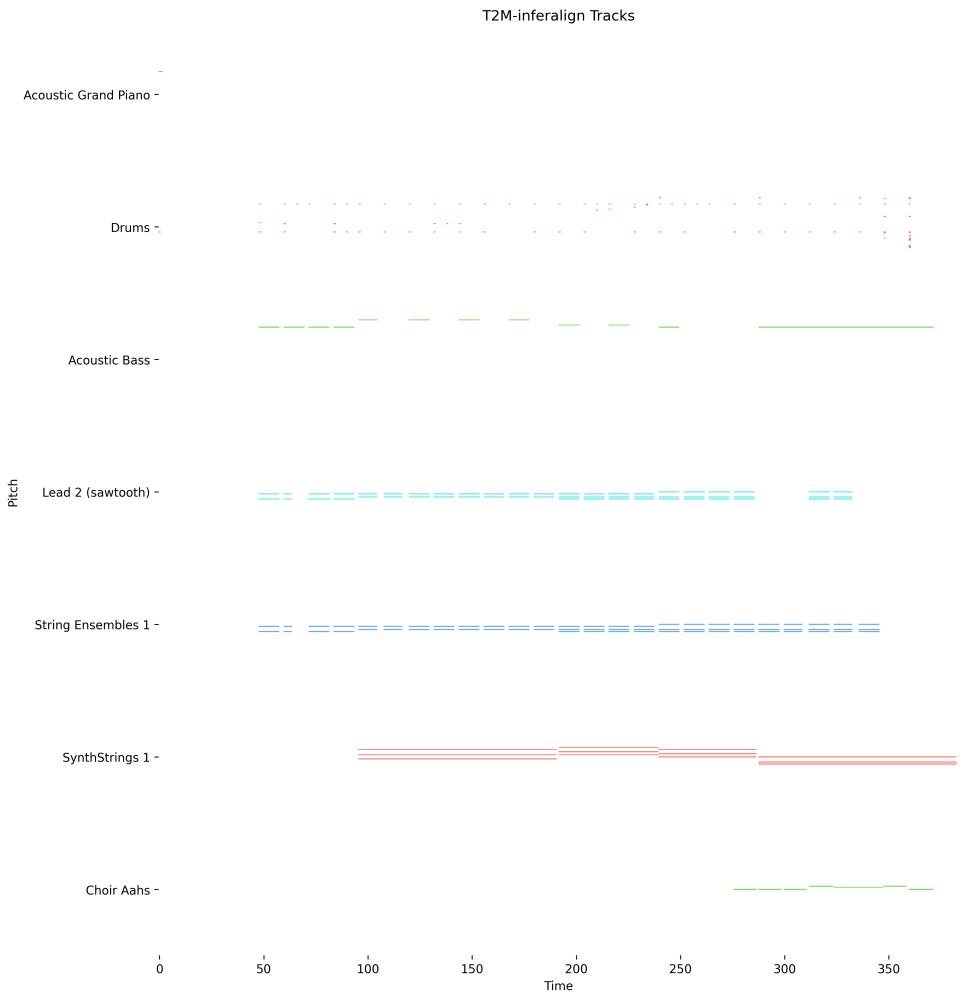

Fig. 7: Visualization of generated symbolic music by T2M-inferalign.

G Ethics Statement

This research involves the use of human-composed music data for training and evaluation of a generative model. All music data were obtained from publicly available datasets or licensed sources and were used solely for academic research under fair use or equivalent scholarly exemptions. No personally identifiable information was involved. We did not conduct any experiments involving human subjects or listeners. All experiments were carried out in compliance with institutional guidelines on ethical research using existing data.

H Adverse Impact Statement

This work presents a generative music model trained on datasets containing copyrighted music. Although the model is developed purely for academic research, we acknowledge the potential risks associated with unauthorized reproduction of copyrighted styles or melodies. To mitigate such risks, we refrain from releasing any trained models or audio outputs that may resemble specific copyrighted works. Our publication focuses on methodological contributions rather than commercial deployment. We explicitly discourage any misuse of our approach for content piracy, unauthorized distribution, or infringement of intellectual property rights.