SongBloom: Coherent Song Generation via Interleaved Autoregressive Sketching and Diffusion Refinement

Chenyu Yang[1,4] **Shuai Wang**[3,4,*] **Hangting Chen**[2] **Wei Tan**[2]
Jianwei Yu **Haizhou Li**[1,4,*]

[1]The Chinese University of Hong Kong, Shenzhen [2]Tencent AI Lab
[3]Nanjing University [4]Shenzhen Research Institution of Big Data

Abstract

Generating music with coherent structure, harmonious instrumental and vocal elements remains a significant challenge in song generation. Existing language models and diffusion-based methods often struggle to balance global coherence with local fidelity, resulting in outputs that lack musicality or suffer from incoherent progression and mismatched lyrics. This paper introduces **SongBloom**[1], a novel framework for full-length song generation that leverages an interleaved paradigm of autoregressive sketching and diffusion-based refinement. SongBloom employs an autoregressive diffusion model that combines the high fidelity of diffusion models with the scalability of language models. Specifically, it gradually extends a musical sketch from short to long and refines the details from coarse to fine-grained. The interleaved generation paradigm effectively integrates prior semantic and acoustic context to guide the generation process. Experimental results demonstrate that SongBloom outperforms existing methods across both subjective and objective metrics and achieves performance comparable to the state-of-the-art commercial music generation platforms. Audio samples are available on our demo page: `https://cypress-yang.github.io/SongBloom_demo`.

1 Introduction

Music, as one of the most expressive forms of human art, serves as a universal language that transcends cultural and linguistic boundaries. Lyric-to-song generation, which involves producing both vocals and accompaniment from given lyrics, requires comprehensive modeling of diverse musical elements, including instrumentation, vocal expressiveness, structural arrangement, and emotional dynamics. These complexities pose significant challenges for existing generative approaches [1, 2]. In particular, end-to-end models that directly synthesize music face considerable challenges in maintaining sound quality, as music's wide frequency range and complex temporal dynamics impose stringent demands on generative fidelity and model capacity.

Advanced approaches to long-form generation typically adopt either a unified non-autoregressive (NAR) architecture [3] that models the entire process with a diffusion transformer, or an autoregressive (AR) framework [4, 5] that employs language models (LMs) and separates the generation into semantic and acoustic stages. However, NAR models often struggle to capture precise alignments between phonemes and audio frames, whereas AR approaches typically predict quantized tokens that suffer from low bitrates, resulting in weak musicality and degraded audio quality.

*Corresponding authors.

[1]Code: `https://github.com/Cypress-Yang/SongBloom`

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

System	Architecture	Size	Max Length	Sample Rate (kHz)	Structure
SongGen [9]	LM	1.1B	30s	16	✗
SongEditor [5]	LM + Diff.	0.7B + 1B	120s	44.1	✓
DiffRhythm-base [3]	Diff.	1.1B	95s	44.1	✗
DiffRhythm-full [3]	Diff.	1.1B	285s	44.1	✗
YuE [4]	LM + LM	7B + 1B	300s	44.1	✓
SongBloom-tiny	LM & Diff.	1.3B	60s	48	✓
SongBloom-full	LM & Diff.	2B	150s	48	✓

Table 1: Comparison of existing song generation models. "+" indicates two cascaded models, while "&" indicates one unified model with jointly optimized objectives.

To address these challenges, we propose SongBloom, a novel lyric-to-song generation approach that combines the strengths of both AR and NAR approaches. SongBloom is built upon an autoregressive diffusion architecture [6], which is well-suited for continuous-valued modalities such as music and speech, and has shown strong scalability and robust long-range generation capabilities [7] comparable to that of audio language models. Given structured lyrics and a 10-second reference audio clip, SongBloom can generate full-length songs with diverse sections up to 150 seconds long.

To further improve the semantic alignment in long-form song generation, we decouple high-level sketch planning from low-level acoustic synthesis, and integrate both within the unified autoregressive framework that supports joint modeling in the style of Chain-of-Thought (CoT) prompting [8]. Notably, we observe that the entire sketch sequence is not always necessary for predicting each acoustic frame. Instead, the acoustic context itself can provide valuable guidance for shaping subsequent sketch planning. Based on this insight, we partition both semantic and acoustic sequences into patches and generate them in an interleaved manner. This paradigm not only facilitates bidirectional contextual exchange between sketching and refinement stages but also significantly reduces the sequence length needed during acoustic synthesis.

Both subjective and objective evaluations are conducted in our experiments. The assessment employs a comprehensive suite of metrics covering multiple dimensions, including musicality, audio quality, structural coherence, and overall musical aesthetics. The results show that SongBloom consistently outperforms all open-source baselines and surpasses most commercial systems, achieving state-of-the-art performance across several key metrics.

The main contributions of this paper can be concluded as:

- We propose SongBloom, the first autoregressive diffusion-based model for full-length song generation, which is capable of producing high-quality and expressive songs with coherent structure and rich acoustic details.
- We design a unified framework that combines coarse and fine stages into a single, jointly optimized model, and treats sketch tokens as CoT-like prompts for acoustic generation directly. Additionally, we introduce a novel interleaved generation paradigm that alternates between sketch and acoustic patches, effectively utilizing both semantic and acoustic contexts.
- Experiments demonstrate that SongBloom surpasses all non-commercial baselines and achieves competitive performance with the state-of-the-art Suno-v4.5 in terms of both subjective and objective metrics. Additionally, our approach achieves a relatively low real-time factor (RTF) compared to other LM-based methods, indicating a favorable trade-off between generation quality and computational efficiency.

2 Related work

2.1 Song Generation

Song generation models aim to produce both singing voice and accompanying music. Previous studies, such as Melodist [10], SongGen [9], and MelodyLM [11], have shown promising results in generating sentence-level singing and accompaniment. However, generating long-form song pieces that encompass diverse musical structures, such as verses, choruses, and instrumental-only sections,

remains a significant challenge. JukeBox [12] was one of the earliest attempts at generating long-form songs, though it suffers from limited genre and timbre control. SongComposer [13] supports long-form lyric-to-melody generation but struggles to preserve certain aspects such as timbre.

Recent research has increasingly focused on long-form and structure-guided song generation. DiffRhythm [3] introduces a single diffusion model capable of generating songs up to 285 seconds in length while maintaining a low real-time factor. YuE [4] employs a coarse-to-fine generation strategy, leveraging two large-scale language models to produce acoustic tokens with improved fidelity and structure. SongEditor [5] proposes a two-stage framework that supports both full song generation and infilling-based editing tailored to specific lyrics, offering greater flexibility for user-driven customization. In addition, commercial platforms such as Suno[2] and Udio[3] have demonstrated promising capabilities in complete song generation. However, due to the lack of publicly available technical details, these systems cannot be thoroughly evaluated or compared.

2.2 Autoregressive Diffusion Models

While language models predominantly operate in discrete-valued space, this paradigm might be suboptimal for inherently continuous audio signals. A common workaround involves transforming waveforms into discrete token sequences through vector quantization (VQ)[14, 15] or residual vector quantization (RVQ)[16]. Recent work, however, challenges the necessity of discretization. For instance, Li et al. [6] demonstrates that discrete tokens may not be essential for autoregressive language models in image generation. Some research has also explored autoregressive diffusion techniques for text [17] and speech [18] generation. ARDiT [19] introduces a decoder-only diffusion transformer for zero-shot text-to-speech generation, achieving impressive results without relying on discrete tokens. DiTAR [7] further simplifies the architecture through a divide-and-conquer strategy, making the model more adaptable for large-scale deployment.

However, the approaches mentioned above remain restricted to the speech domain, and direct migration faces some limitations. First, songs are typically longer than spoken utterances, imposing stricter requirements on contextual consistency across extended durations. Second, the presence of musical accompaniment leads to a higher semantic density of each frame, making generation more difficult. Recent works [20, 21] introduce a shared transformer architecture capable of handling both discrete and continuous tokens across different modalities. Inspired by this design, we propose a unified autoregressive framework that integrates both sketch prediction and audio synthesis with distinct training objectives.

3 Preliminary

Next-Token Prediction Given a sequence of tokens $\mathbf{x} = (x_1, x_2, \ldots, x_{t-1})$, the objective is to model the conditional probability of the next token $P(x_t \mid x_1, x_2, \ldots, x_{t-1})$. For an entire sequence $(x_1, x_2, \ldots, x_T)$, the joint probability is factorized as:

$$P(x_1, x_2, \ldots, x_T|C) = \prod_{t=1}^{T} P(x_t \mid x_1, x_2, \ldots, x_{t-1}; C), \tag{1}$$

where C represents the conditions.

Rectified Flow Matching Rectified flow-matching [22] defines a linear interpolation between a source point z_0 and a target point z_1 over time $t \in [0, 1]$:

$$z_t = (1-t)z_0 + tz_1 \tag{2}$$

$$\mathrm{d}z_t = (z_1 - z_0)\mathrm{d}t = v(z_t, t)\mathrm{d}t \tag{3}$$

where z_0 denotes the original data and $z_1 \sim \mathcal{N}(0, 1)$ is a random noise. $v(z_t, t)$ represents the velocity field guiding the transformation from z_0 to z_1. The reverse process enables the generation by inverting this transformation:

$$z_{t-\Delta t} = z_t - v(z_t, t) \cdot \Delta t \tag{4}$$

This formulation allows the model to sample from the data distribution by iteratively denoising from the noise distribution.

[2] `https://suno.com`

[3] `https://udio.com`

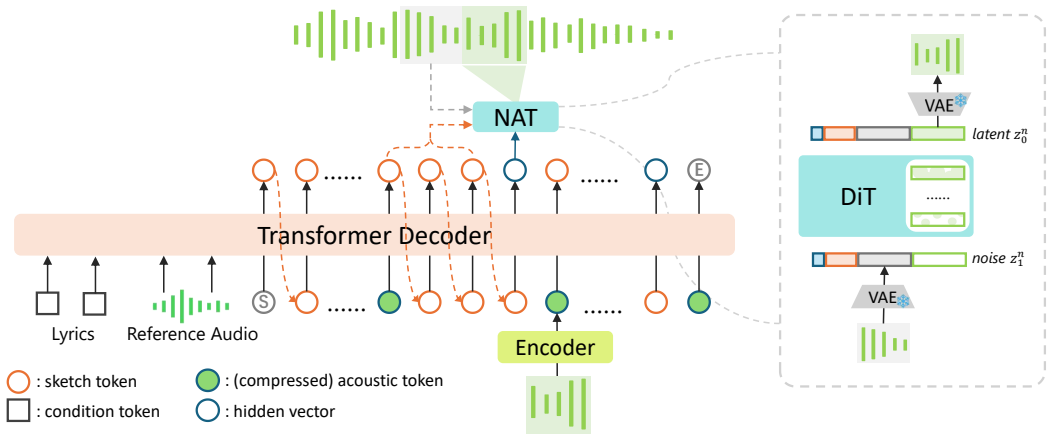

Figure 1: Overall architecture of SongBloom.

4 SongBloom

4.1 Task Formulation

MusicLM [23] proposes a two-stage framework that leverages semantic tokens, referred to as *sketch*, extracted from self-supervised learning (SSL) models such as MERT [24] as intermediate representations. In the first stage, an LM is employed to generate the sketch sequence from scratch, as its variable length and discrete nature make it well-suited for language modeling approaches, which have demonstrated strong scalability. In the second stage, either diffusion or LM subsequently generates *acoustic* tokens in a coarse-to-fine manner. The overall generation process can be formulated as:

$$p(a_{(0:T]}, s_{(0:T]} \mid C) = p_\theta(a_{(0:T]} \mid s_{(0:T]}, C) \cdot p_\phi(s_{(0:T]} \mid C), \tag{5}$$

where s_i denotes the sketch token at frame i, and a_i denotes the acoustic token. T is the total length of the frame sequence. The parameters ϕ and θ represent the models responsible for generating semantic and acoustic tokens, respectively. This two-stage architecture has been widely adopted in previous long-form music generation approaches such as [5, 4, 25].

However, the aforementioned formulation faces several limitations: 1) Future sketch tokens contribute little to the prediction of current acoustic tokens due to the strict token-level alignment between the two representations. As noted in Yang et al. [5], Xu et al. [26], using chunk-level semantic-to-acoustic reconstruction introduces minimal degradation in fluency, suggesting that generating the complete semantic sequence in advance may be unnecessary. 2) Given the rich and expressive nature of music, prior acoustic latents can provide valuable contextual cues for sketch token generation. However, existing approaches typically condition only on the semantic history, thereby overlooking potentially useful acoustic information.

To address the aforementioned challenges, we propose a novel generation paradigm that interleaves the generation of sketch and acoustic sequences. Both types of tokens are first segmented into fixed-size patches. The modified generation process is formulated as:

$$p(a_{(0:T]}, s_{(0:T]} \mid C) = \prod_{i=0}^{N} p_\theta(s_{(iP:(i+1)P]} | s_{(0:iP]}, a_{(0:iP]}, C) \cdot p_\phi(a_{(iP:(i+1)P]} | s_{(0:(i+1)P]}, a_{(0:iP]}, C), \tag{6}$$

where P denotes the patch size, $N = \lceil T/P \rceil - 1$ denotes the number of patches. θ and ϕ correspond to the parameters responsible for generating sketch tokens and acoustic features, respectively. Although a sequential dependency between the two stages still exists, the interleaved generation paradigm enables bidirectional information exchange between the semantic and acoustic representations. The two stages share a subset of model parameters and are jointly optimized within a unified framework, facilitating coherent and high-fidelity song generation.

4.2 Data Representation

Lyric preprocessing To incorporate structural information, we introduce two categories of structure flags into the lyric sequence. Vocal-based flags, which indicate the structure of vocal sections (e.g., verse, chorus), are prepended to the beginning of each paragraph. Accompaniment-based flags, which represent non-vocal regions such as intros and outros, are inserted between paragraphs and expanded proportionally according to their exact durations. All lyric text is normalized and then transformed into a phoneme sequence to serve as input for the subsequent sketch generation stage.

Sketch tokens In this paper, we adopt the embeddings extracted from MuQ [27] as our sketch representation. A single vector quantization layer is attached to discretize the sketch embeddings into flattened tokens. Additionally, we have explored various sketch alternatives including musical signals, which are further discussed in Appendix A.

Acoustic latents Due to the broader frequency spectrum of music compared to speech, modeling and reconstruction become significantly more challenging. Codec-based discretization methods [28, 29] inherently suffer from a trade-off between reconstruction fidelity and the depth of codebooks, often increasing generation complexity. To tackle this issue, we substitute the discrete acoustic tokens with continuous latents derived from an autoencoder, which compresses 2-channel 48 kHz music into continuous-valued acoustic latent sequences at a reduced frame rate. This continuous representation preserves high-frequency detail while simplifying the generation process, making it more suitable for high-fidelity music synthesis.

4.3 Architecture

4.3.1 Overview

The overall model architecture is illustrated in Figure 1, which consists of three main modules: an autoregressive transformer decoder, a non-autoregressive diffusion transformer, and an acoustic encoder. Both semantic and acoustic sequences are first segmented into fixed-length patches and then generated in an interleaved manner, enabling the model to maintain consistency between high-level structure and fine-grained acoustic detail.

4.3.2 Autoregressive Sketch Generation

The autoregressive sketch generation stage is designed to generate both sketch tokens $p_\theta(s_{(iP:(i+1)P]}|\cdot)$ and a corresponding hidden vector $p_\theta(h_i|\cdot)$ per patch, which serve as conditioning inputs for the downstream diffusion stage. The architecture is composed of a stack of transformer decoders with causal masks, enabling left-to-right autoregressive prediction over the sketch tokens. Conditions, including lyric text and style prompt, are prepended to the semantic stream. Each semantic patch is generated based on all previously generated sketch tokens, as well as the acoustic latent features from preceding patches. Specifically, the acoustic features of the current patch are compressed via an acoustic encoder and then inserted as the next token at the position of the hidden vector. This design eliminates the need for custom attention masks, thereby facilitating compatibility with acceleration techniques such as FlashAttention2 [30].

The training objective for the autoregressive sketch generation is formulated as a cross-entropy loss:

$$\mathcal{L}_{\text{LM}} = -\frac{1}{NP}\sum_{i=0}^{N-1}\sum_{j=1}^{P}\log p_\theta(s_{iP+j} \mid s_{<iP+j}, a_{<iP}, C), \tag{7}$$

where s_t denotes the sketch token at timestep t.

4.3.3 Non-Autoregressive Latent Diffusion

The non-autoregressive diffusion module predicts acoustic latents within each patch in parallel using a full-attention DiT [31] architecture. The model is trained with the Rectified Flow-Matching (RFM) objective [22], which aims to predict the velocity field $v_\phi(\cdot)$ governing the dynamics of the latent trajectories:

$$\frac{\mathrm{d}\mathbf{z}_t^i}{\mathrm{d}t} = v_\phi(t, \mathbf{z}_t^i \mid h_i, s_{(iP:(i+1)P]}, \mathbf{z}_0^{i-1}). \tag{8}$$

Here, t denotes the diffusion time step. In our experiments, we sample t from a logit-normal distribution π_t following Esser et al. [32]. $\mathbf{z}_t^i$ represents the latent variables at time t for the i-th patch, where $\mathbf{z}_0^i = a_{(iP:(i+1)P]}$ and $\mathbf{z}_1^i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The variable h_i is the hidden vector inherited from the first stage. $s_{(iP:(i+1)P]}$ refers to the sketch tokens for the current patch, and $\mathbf{z}_0^{i-1}$ refers to the acoustic latents from the preceding patch.

The loss function for training is defined as:

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_{t\sim\pi_t,\mathbf{z}^i}\left[||v_\phi(t, \mathbf{z}_t^i \mid \cdot) - (\mathbf{z}_1^i - \mathbf{z}_0^i)||^2\right], \tag{9}$$

where π_t denotes the distribution over diffusion steps. The total training objective combines the sketch generation loss and the diffusion loss:

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \lambda \cdot \mathcal{L}_{\text{flow}}, \tag{10}$$

where $\lambda = 0.1$ is an empirically chosen weighting factor.

Gradients are backpropagated from the diffusion stage to the sketch generation stage via the hidden vector h_i. Additionally, the sketch tokens share the same embedding layer across both stages, which improves training efficiency and accelerates convergence.

Jia et al. [7] introduced a classifier-free guidance (CFG) mechanism for autoregressive diffusion models, which is applied only to the hidden vector. In our work, we treat the hidden vector and sketch tokens as a unified conditioning signal, and jointly mask them during training. Additional masks are applied solely to the sketch tokens to further encourage the hidden vector to carry more semantic information.

5 Experimental Setup

5.1 Dataset

We use a large-scale song dataset totaling 100K hours, including both Chinese and English songs, to train our model. The original lyrics are of relatively low quality, prompting the use of a dedicated data cleaning pipeline. First, we separate the vocal and instrumental tracks using the Demucs model [33]. Then we use the WhisperX toolkit [34] to filter out mismatched lyric sentences and recover missing words. The time boundaries of lyrics are also refined based on the results of automatic speech recognition (ASR). Additionally, structural information is extracted from the original tracks using the structure analyzer proposed by Kim and Nam [35].

For inference, considering that most current song generation models are trained on large-scale but closed-source datasets, it is crucial to ensure that the test samples are not seen during training. To this end, the lyrics are generated using GPT-4o, and the reference audio is randomly clipped from unseen real songs with diverse genres, followed by random shuffling to prevent potential data leakage. A subset comprising 20 samples is used for subsequent human evaluation.

5.2 Configuration & Training Setup

In our experiments, the codebook size for semantic tokens is set to 16384, at a frame rate of 25. The stable-audio-vae [36] is adopted as the implementation of our waveform autoencoder. Minor modifications are made to its hyperparameters to ensure they have the same frame rate, thereby facilitating synchronized sketches and latents.

The core component of SongBloom is based on the LLaMA-2 decoder architecture [37], utilizing causal attention as the backbone of the autoregressive LM. This architecture is further modified to support bidirectional attention, forming our diffusion transformer. Rotary Positional Embeddings (RoPE) [38] are employed in both autoregressive and non-autoregressive transformers to encode positional information. The acoustic encoder is a simple two-layer convolutional network. All conditioning inputs are prepended to the input sequence. Attention modules in each layer use 24 heads with a hidden dimension of 1536, consistent across both autoregressive and non-autoregressive components. The patch size is set to 16, spanning 0.64 seconds. We evaluate two model configurations in our experiments: (1) **SongBloom-tiny**, comprising 16 layers for the autoregressive LM and 8 layers for the non-autoregressive diffusion transformer, capable of generating songs up to 60 seconds in length; and (2) **SongBloom-full**, comprising 24 autoregressive layers and 12 non-autoregressive

Table 2: Objective evaluation results across all models. The upper section reports the performance of commercial platforms (valid until June 23, 2025), while the lower section presents results for open-source baselines as well as our proposed models.

Models	Prompt	PER(%)↓	MCC↑	FAD↓	SER(%)↓	Aesthetic Score				RTF↓
						CE↑	CU↑	PC↑	PQ↑	
Suno-v4.5 †	text	24.67	0.69	3.39	**10.43**	7.77	7.93	6.03	8.40	-
Udio-v1.5	10s wav	20.04	0.79	4.04	17.92	7.47	7.63	6.29	8.20	-
Haimian	text	10.03	0.63	5.45	13.39	7.55	7.87	5.75	8.28	-
Mureka-O1	10s wav	7.79	0.86	3.39	31.37	7.69	7.84	**6.41**	8.45	-
ACE-step [46]	text	54.34	0.62	8.02	12.27	7.37	7.52	6.26	7.85	-
YuE-7B [4]	text + 30s wav	27.30	0.62	5.99	13.93	7.25	7.59	5.96	8.03	13.724
DiffRhythm-full [3]‡	10s wav	15.77	0.70	5.04	46.62	5.81	7.29	4.52	7.73	**0.034**
SongEditor [5]	10s wav	16.20	0.77	4.85	18.06	7.44	7.80	6.06	8.27	1.717
SongBloom-full	10s wav	6.75	**0.88**	3.43	17.67	7.71	7.88	5.86	8.43	1.649
SongBloom-full-ft	10s wav	**5.49**	0.86	**3.20**	14.50	**7.79**	**7.96**	5.88	**8.47**	1.649

†DiffRhythm requires additional sentence-level timestamps. We first extract paragraph-level timestamps from ASR results of generated samples, then employ GPT-4o to predict the start time of each sentence.

layers, enabling song generation up to 150 seconds. The former is used for analysis and ablation studies, while the latter is included to enable fair comparisons with other baselines. The model is trained with 16 A100 GPUs for approximately one week.

All models are trained using the AdamW optimizer [39] with a learning rate of 1e-4. A cosine learning rate scheduler [40] with 2000 warm-up steps is employed to stabilize early training. Each model is trained for approximately 150K steps with a batch size of 128. The DeepSpeed strategy [41] is adopted to support efficient training. For inference, both stages share a classifier-free guidance coefficient of 1.5. Next-token prediction is performed using top-k sampling with $k = 200$ and a temperature of 0.9. The diffusion process employs the Euler ODE solver with 36 diffusion steps.

5.3 Evaluation Metrics

We evaluate the proposed models using the following objective metrics: (1) *Phoneme Error Rate (PER)*, computed based on the separated vocal tracks and the corresponding lyrics; (2) *MuLan Cycle Consistency (MCC)*, which measures the cosine similarity of MuLan [42] embeddings between generated samples and reference audio or textual descriptions; (3) *Fréchet Audio Distance (FAD)*[43], which quantifies the distributional similarity between generated samples and real songs from which the clips are intercepted; and (4) *Structural Error Rate (SER)*, which measures the mismatch between detected structural patterns and the target lyric structure. The Dynamic Time Warping (DTW) algorithm [44] is first employed to obtain the optimal temporal alignment, and then we calculate the proportion of error duration. Additionally, we leverage the Audiobox-Aesthetic [45] to assess musical aesthetics, including content enjoyment (CE), content usefulness (CU), production complexity (PC), and production quality (PQ).

For subjective evaluation, we conduct a Mean Opinion Score (MOS) listening test. A group of at least 10 participants with musical expertise are invited to rate each sample on a scale from 1 to 5. The evaluation focuses on several aspects: musicality (MUS) and audio quality (QLT) assessed separately for both the vocal and accompaniment components, correctness (CRR) of lyrics, and consistency (CST) between samples and prompts.

6 Results

6.1 Full-Length Lyric-to-Song Generation

Based on the SongBloom-full model, we fine-tuned it for 1,000 additional steps on synthesized data with a clear alternating structure of verses and choruses, denoted as **SongBloom-full-ft**. This fine-tuning process enhances the model's ability to emulate synthesized lyric compositions, leading to improved performance on lyrics with similar stylistic patterns.

Comparison of objective metrics As shown in Table 2, SongBloom demonstrates superior performance across various metrics and is competitive with Suno-v4.5, the state-of-the-art commercial song

Table 3: Subjective evaluation results. Consistency (CST) is reported only for models that accept a 10-second reference audio as the style prompt.

Models	$\text{MUS}_V\uparrow$	$\text{MUS}_A\uparrow$	$\text{QLT}_V\uparrow$	$\text{QLT}_A\uparrow$	CRR↑	CST↑
Suno-v4.5	<u>3.87</u> ± 0.28	**4.04** ± 0.18	3.83 ± 0.12	**3.96** ± 0.06	2.95 ± 0.14	-
Udio-v1.5	3.28 ± 0.27	3.55 ± 0.25	3.62 ± 0.23	3.74 ± 0.21	2.57 ± 0.19	2.76 ± 0.31
Haimian	3.39 ± 0.26	3.65 ± 0.24	<u>3.86</u> ± 0.17	3.90 ± 0.11	3.09 ± 0.20	-
Mureka-O1	**3.91** ± 0.26	<u>3.93</u> ± 0.14	3.85 ± 0.18	3.89 ± 0.12	<u>3.38</u> ± 0.16	3.41 ± 0.39
ACE-step [46]	2.95 ± 0.29	2.93 ± 0.32	2.66 ± 0.27	2.68 ± 0.25	2.11 ± 0.16	-
YuE-7B [4]	2.93 ± 0.27	3.15 ± 0.24	3.16 ± 0.27	3.28 ± 0.23	2.54 ± 0.19	-
DiffRhythm-full [3]	2.99 ± 0.27	3.48 ± 0.27	2.97 ± 0.21	3.33 ± 0.26	2.51 ± 0.23	2.45 ± 0.28
SongEditor [5]	2.90 ± 0.32	3.11 ± 0.27	3.03 ± 0.32	3.24 ± 0.28	2.89 ± 0.16	2.44 ± 0.28
SongBloom-full	3.59 ± 0.25	3.60 ± 0.23	3.83 ± 0.09	3.81 ± 0.08	3.27 ± 0.17	**3.62** ± 0.27
SongBloom-full-ft	**3.91** ± 0.24	3.92 ± 0.12	**3.95** ± 0.04	<u>3.93</u> ± 0.10	**3.42** ± 0.18	<u>3.45</u> ± 0.31

generation platform. After fine-tuning on downstream data, SongBloom-full-ft even outperforms Suno-v4.5 in several metrics.

During our evaluation, we observed that Suno tends to follow rigid structural patterns, such as redundantly repeating the chorus at the end, which leads to structural hallucinations and degraded PER performance. In contrast, SongBloom adheres more faithfully to the structure of the input lyrics, enabling flexible and structurally coherent song generation that significantly reduces the PER. In terms of the MCC metric, waveform-based style prompts generally possess a natural advantage over text descriptions, as they always provide a comprehensive and unbiased representation of all musical components. Among these approaches, SongBloom-full achieves the highest MCC score. In terms of the automated-evaluated aesthetic scores, SongBloom-full-ft outperforms all other baselines in three out of the four metrics, further demonstrating its ability to generate high-quality, musically coherent, and aesthetically pleasing songs.

Apart from generation performance, we also assessed the inference speed of SongBloom compared to other open-source models. The integrated design of SongBloom enables outstanding inference efficiency, yielding a lower RTF than all other autoregressive baselines. Compared to SongEditor, our model achieves a similar RTF despite having a larger size, since SongEditor takes the entire semantic sequence as input during the diffusion stage, leading to unnecessary computational overhead. Meanwhile, YuE utilizes two LMs and generates multi-layer codec tokens in a flattened pattern, significantly extending the generation sequence. While autoregressive models are naturally slower than their non-autoregressive counterparts, SongBloom achieves an excellent trade-off between efficiency and performance.

Comparison of subjective metrics Table 3 shows that SongBloom and Suno-v4.5 dominate the human evaluation results, outperforming all other models by a clear margin. Suno-v4.5 demonstrates particular strength in accompaniment generation, whereas SongBloom excels in metrics related to vocal tracks. Notably, SongBloom achieves the highest correctness score, indicating a stronger adherence to the provided lyrics compared to other systems. This suggests that its interleaved generation paradigm effectively preserves semantic intent throughout the song. Furthermore, SongBloom's superior performance in consistency is corroborated by its high MCC score, reflecting its ability to maintain thematic coherence over every section of songs.

6.2 Ablation Study on Diffusion Conditions

Table 4 compares the performance of SongBloom-tiny under different combinations of diffusion conditions. In the absence of sketch tokens, the LM stage produces only a sequence of hidden vectors, and the diffusion stage reconstructs the audio solely based on them. As shown in the table, the sketch plays a critical role in generation. Without it as a coarse-grained CoT, the model fails to learn proper alignment between phoneme sequences and audio, resulting in extremely high PER. In contrast, when the sketch is provided, even without additional conditions, the diffusion transformer can generate intelligible vocals solely based on the hidden vector from the LM stage. Meanwhile, incorporating both the acoustic context and sketch tokens as input conditions during the diffusion stage further enhances both objective and subjective performance, leading to higher-quality generation results.

Table 4: Ablation study of SongBloom-tiny under different diffusion conditions. "H" represents the hidden vector, "C" represents the acoustic context from the previous patch, and "S" represents the sketch tokens of the current patch.

w/ sketch	Conditions	Objective		Aesthetic Score				Subjective	
		PER(%)↓	FAD↓	CE↑	CU↑	PC↑	PQ↑	MUS↑	QLT↑
✓	H+C+S	**9.44**	5.60	**7.57**	**7.66**	6.00	**8.25**	**3.55**	**3.77**
	H+C	10.49	**5.16**	7.43	7.57	**6.42**	8.15	3.43	3.46
	H	11.35	7.58	7.46	7.50	5.80	8.09	3.14	3.41
✗	H+C	109.76	8.86	7.06	7.40	5.92	7.71	-	-

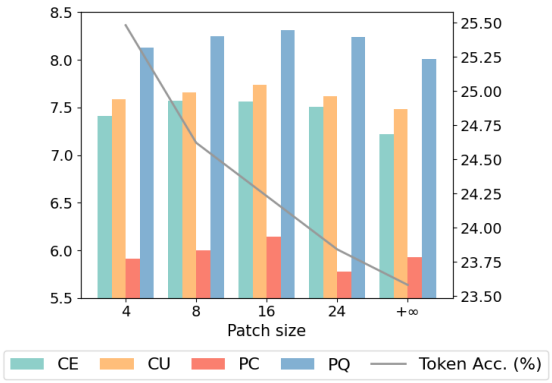

Figure 2: Aesthetic scores and sketch token accuracy of SongBloom-tiny with various patch sizes.

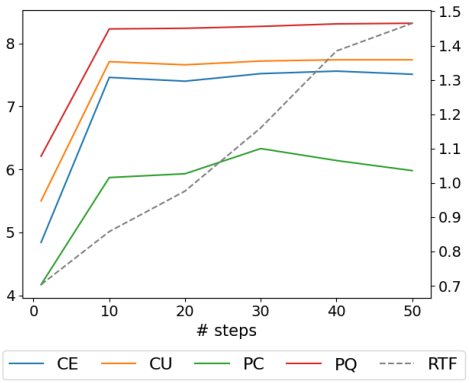

Figure 3: Aesthetic scores of SongBloom-tiny with increasing diffusion steps.

6.3 Effect of Hyper-Parameters

Figure 2 illustrates the impact of various patch sizes, where "$+\infty$" denotes separating the generation of sketch and acoustic sequence. For speech-oriented autoregressive diffusion models [7, 19], a small patch size is essential, as larger patch sizes make the model difficult to converge. For SongBloom, since the sketch tokens have served as the CoT of hidden vectors and guide the diffusion directly, a small patch size is no longer the optimal choice. Although smaller patch sizes provide more acoustic information for sketch generation, thereby improving sketch token accuracy, they also hinder the fluency of acoustic latents during the diffusion stage, as the window for preceding contexts becomes smaller. Figure 3 illustrates the relationship between performance and diffusion steps during inference. The RTF increases proportionally with the number of diffusion steps, while near-optimal generation performance is achieved in as few as 10 steps, indicating that the inference process can be further accelerated.

7 Discussions and Limitations

In this paper, we propose SongBloom, a novel approach for full-song generation that produces expressive, high-quality songs and achieves state-of-the-art performance across multiple metrics. Nevertheless, several open challenges remain. First, the current sketch representation is derived from SSL models, which lack interpretability. Replacing these with some symbolic formats could enable more fine-grained control and user customization, which will be our future work. Second, we believe that some reinforcement learning-based techniques, such as DPO [47] or PPO [48], can be applied to SongBloom, which aligns the generation process with user preferences, thereby enabling outputs that better match human aesthetic judgments.

We fully acknowledge the potential ethical risks associated with music generation models. We ensure that both our models and training data are strictly used for academic research purposes only. We respect the intellectual property rights of all original artists and content creators. Every effort has been made to avoid the use of copyrighted material without proper authorization.

8 Acknowledgement

This work is supported by National Natural Science Foundation of China (Grant No. 62401377 and No. 62271432), Shenzhen Science and Technology Program (Shenzhen Key Laboratory, Grant No. ZDSYS20230626091302006), Program for Guangdong Introducing Innovative and Entrepreneurial Teams, Grant No. 2023ZT10X044.

References

[1] J.-P. Briot and F. Pachet, "Deep learning for music generation: challenges and directions," *Neural Computing and Applications*, vol. 32, no. 4, pp. 981–993, 2020.

[2] C. Hernandez-Olivan and J. R. Beltran, "Music composition with deep learning: A review," *Advances in speech and music technology: computational aspects and applications*, pp. 25–50, 2022.

[3] Z. Ning, H. Chen, Y. Jiang, C. Hao, G. Ma, S. Wang, J. Yao, and L. Xie, "Diffrhythm: Blazingly fast and embarrassingly simple end-to-end full-length song generation with latent diffusion," *arXiv preprint arXiv:2503.01183*, 2025.

[4] R. Yuan, H. Lin, S. Guo, G. Zhang, J. Pan, Y. Zang, H. Liu, Y. Liang, W. Ma, X. Du, X. Du, Z. Ye, T. Zheng, Y. Ma, M. Liu, Z. Tian, Z. Zhou, L. Xue, X. Qu, Y. Li, S. Wu, T. Shen, Z. Ma, J. Zhan, C. Wang, Y. Wang, X. Chi, X. Zhang, Z. Yang, X. Wang, S. Liu, L. Mei, P. Li, J. Wang, J. Yu, G. Pang, X. Li, Z. Wang, X. Zhou, L. Yu, E. Benetos, Y. Chen, C. Lin, X. Chen, G. Xia, Z. Zhang, C. Zhang, W. Chen, X. Zhou, X. Qiu, R. Dannenberg, J. Liu, J. Yang, W. Huang, W. Xue, X. Tan, and Y. Guo, "Yue: Scaling open foundation models for long-form music generation," 2025. [Online]. Available: https://arxiv.org/abs/2503.08638

[5] C. Yang, S. Wang, H. Chen, J. Yu, W. Tan, R. Gu, Y. Xu, Y. Zhou, H. Zhu, and H. Li, "Songeditor: Adapting zero-shot song generation language model as a multi-task editor," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, 2025, pp. 25 597–25 605.

[6] T. Li, Y. Tian, H. Li, M. Deng, and K. He, "Autoregressive image generation without vector quantization," *NeurIPS*, vol. 37, pp. 56 424–56 445, 2024.

[7] D. Jia, Z. Chen, J. Chen, C. Du, J. Wu, J. Cong, X. Zhuang, C. Li, Z. Wei, Y. Wang *et al.*, "Ditar: Diffusion transformer autoregressive modeling for speech generation," *arXiv preprint arXiv:2502.03930*, 2025.

[8] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *NeurIPS*, vol. 35, pp. 24 824–24 837, 2022.

[9] Z. Liu, S. Ding, Z. Zhang, X. Dong, P. Zhang, Y. Zang, Y. Cao, D. Lin, and J. Wang, "Songgen: A single stage auto-regressive transformer for text-to-song generation," *arXiv preprint arXiv:2502.13128*, 2025.

[10] Z. Hong, R. Huang, X. Cheng, Y. Wang, R. Li, F. You, Z. Zhao, and Z. Zhang, "Text-to-song: Towards controllable music generation incorporating vocal and accompaniment," in *ACL*, 2024, pp. 6248–6261.

[11] R. Li, Z. Hong, Y. Wang, L. Zhang, R. Huang, S. Zheng, and Z. Zhao, "Accompanied singing voice synthesis with fully text-controlled melody," *arXiv preprint arXiv:2407.02049*, 2024.

[12] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[13] S. Ding, Z. Liu, X. Dong, P. Zhang, R. Qian, C. He, D. Lin, and J. Wang, "Songcomposer: A large language model for lyric and melody composition in song generation," *arXiv preprint arXiv:2402.17645*, 2024.

[14] A. Vasuki and P. Vanathi, "A review of vector quantization techniques," *IEEE Potentials*, vol. 25, no. 4, pp. 39–47, 2006.

[15] R. Gray, "Vector quantization," *IEEE Assp Magazine*, vol. 1, no. 2, pp. 4–29, 1984.

[16] C. F. Barnes, S. A. Rizvi, and N. M. Nasrabadi, "Advances in residual vector quantization: A review," *IEEE transactions on image processing*, vol. 5, no. 2, pp. 226–262, 1996.

[17] T. Wu, Z. Fan, X. Liu, H.-T. Zheng, Y. Gong, J. Jiao, J. Li, J. Guo, N. Duan, W. Chen *et al.*, "Ar-diffusion: Auto-regressive diffusion model for text generation," *NeurIPS*, vol. 36, pp. 39 957–39 974, 2023.

[18] R. Benita, M. Elad, and J. Keshet, "Diffar: Denoising diffusion autoregressive model for raw speech waveform generation," in *ICLR*, 2024.

[19] Z. Liu, S. Wang, S. Inoue, Q. Bai, and H. Li, "Autoregressive diffusion transformer for text-to-speech synthesis," *arXiv preprint arXiv:2406.05551*, 2024.

[20] C. Zhou, L. Yu, A. Babu, K. Tirumala, M. Yasunaga, L. Shamis, J. Kahn, X. Ma, L. Zettlemoyer, and O. Levy, "Transfusion: Predict the next token and diffuse images with one multi-modal model," *arXiv preprint arXiv:2408.11039*, 2024.

[21] Y. Sun, H. Bao, W. Wang, Z. Peng, L. Dong, S. Huang, J. Wang, and F. Wei, "Multimodal latent language modeling with next-token diffusion," *arXiv preprint arXiv:2412.08635*, 2024.

[22] X. Liu, C. Gong *et al.*, "Flow straight and fast: Learning to generate and transfer data with rectified flow," in *ICLR*, 2023.

[23] A. Agostinelli, T. I. Denk, Z. Borsos, J. Engel, M. Verzetti, A. Caillon, Q. Huang, A. Jansen, A. Roberts, M. Tagliasacchi *et al.*, "Musiclm: Generating music from text," *arXiv preprint arXiv:2301.11325*, 2023.

[24] Y. Li, R. Yuan, G. Zhang, Y. Ma, X. Chen, H. Yin, C. Xiao, C. Lin, A. Ragni, E. Benetos *et al.*, "Mert: Acoustic music understanding model with large-scale self-supervised training," *ICLR*, 2024.

[25] M. W. Y. Lam, Q. Tian, T. Li, Z. Yin, S. Feng, M. Tu, Y. Ji, R. Xia, M. Ma, X. Song, J. Chen, W. Yuping, and Y. Wang, "Efficient neural music generation," in *NeurIPS*, vol. 36, 2023, pp. 17 450–17 463.

[26] Y. Xu, H. Chen, J. Yu, W. Tan, R. Gu, S. Lei, Z. Lin, and Z. Wu, "Mucodec: Ultra low-bitrate music codec," *arXiv preprint arXiv:2409.13216*, 2024.

[27] H. Zhu, Y. Zhou, H. Chen, J. Yu, Z. Ma, R. Gu, Y. Luo, W. Tan, and X. Chen, "Muq: Self-supervised music representation learning with mel residual vector quantization," *arXiv preprint arXiv:2501.01108*, 2025.

[28] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "Soundstream: An end-to-end neural audio codec," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 495–507, 2021.

[29] C. Wang, S. Chen, Y. Wu, Z. Zhang, L. Zhou, S. Liu, Z. Chen, Y. Liu, H. Wang, J. Li *et al.*, "Neural codec language models are zero-shot text to speech synthesizers," *arXiv preprint arXiv:2301.02111*, 2023.

[30] T. Dao, "FlashAttention-2: Faster attention with better parallelism and work partitioning," in *ICLR*, 2024.

[31] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4195–4205.

[32] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel *et al.*, "Scaling rectified flow transformers for high-resolution image synthesis," in *ICML*, 2024.

[33] S. Rouard, F. Massa, and A. Défossez, "Hybrid transformers for music source separation," in *ICASSP*, 2023, pp. 1–5.

[34] M. Bain, J. Huh, T. Han, and A. Zisserman, "Whisperx: Time-accurate speech transcription of long-form audio," *INTERSPEECH*, 2023.

[35] T. Kim and J. Nam, "All-in-one metrical and functional structure analysis with neighborhood attentions on demixed audio," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2023.

[36] Z. Evans, J. D. Parker, C. Carr, Z. Zukowski, J. Taylor, and J. Pons, "Stable audio open," in *ICASSP*, 2025, pp. 1–5.

[37] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[38] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, p. 127063, 2024.

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

[40] ——, "Sgdr: Stochastic gradient descent with warm restarts," in *ICLR*, 2022.

[41] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "Zero: Memory optimizations toward training trillion parameter models," in *International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020, pp. 1–16.

[42] Q. Huang, A. Jansen, J. Lee, R. Ganti, J. Y. Li, and D. P. Ellis, "Mulan: A joint embedding of music audio and natural language," *ISMIR*, 2022.

[43] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms," in *Interspeech*, 2019, pp. 2350–2354.

[44] P. Senin, "Dynamic time warping algorithm review," *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, vol. 855, no. 1-23, p. 40, 2008.

[45] A. Tjandra, Y.-C. Wu, B. Guo, J. Hoffman, B. Ellis, A. Vyas, B. Shi, S. Chen, M. Le, N. Zacharov *et al.*, "Meta audiobox aesthetics: Unified automatic quality assessment for speech, music, and sound," *arXiv preprint arXiv:2502.05139*, 2025.

[46] J. Gong, S. Zhao, S. Wang, S. Xu, and J. Guo, "Ace-step: A step towards music generation foundation model," *arXiv preprint arXiv:2506.00045*, 2025.

[47] 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," *NeurIPS*, vol. 36, pp. 53 728–53 741, 2023.

[48] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We describe our method in the abstract and conclude the contributions in the introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations at the end of this paper.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: This paper does not include many theoretical results that need proof. For the task formulation, we have explained the reason.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: In both methodology and experiment actions, we disclose the information of our experiments as much as possible.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: Some training data is sensitive and cannot be made public. We are collecting other desensitized data to train an open-source version.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See the experimental setup section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For subjective evaluation results, we have given the variance of scores.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide it in the experimental setup section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We read the NeurIPS Code of Ethics carefully and strictly adhere to it.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss it at the end of the paper.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: There will be a potential misuse if we release the model weights and training data. However, currently we do not do that.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We describe the source of code and the websites of commercial platforms used in our experiments.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer:[Yes]

    Justification: We provide the translated version of the scoring criteria in the appendix. All raters have been paid.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: There are no potential risks during scoring.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: LLM is only used in very few text processing steps.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Ablation Study on Different Sketch Choices

Table 5 demonstrates our early efforts at decomposing the semantic information of sketches. We compare different sketch representations, including pitch, pitch + chromagram, and semantic embeddings. As shown in the table, simple pitch-based sketches offer limited alignment capability, resulting in high PER and suboptimal performance in other metrics. The inclusion of chroma features slightly improves alignment, but still falls short of fully capturing the rich semantics needed for coherent vocal generation. These results validate the importance of incorporating abstract, semantically meaningful information into the sketch stage, and lay the groundwork for future exploration of interpretable yet powerful sketch formats.

Type	PER(%)↓	FAD↓	CE↑	CU↑	PC↑	PQ↑
no sketch	109.76	8.86	7.06	7.40	5.92	7.71
pitch	103.22	5.43	7.34	7.58	6.03	8.04
pitch + chroma	68.47	5.47	7.37	7.57	5.98	8.06
SSL embedding	9.44	5.60	7.55	7.66	6.00	8.25

Table 5: Impact of sketch types on SongBloom's performance.

B Time Complexity of SongBloom Inference

We analyze the inference time complexity of SongBloom compared to a decoupled two-stage model, assuming both have the same number of layers.

Let L_1 denote the number of layers in the language model stage, and L_2 the number of layers in the diffusion stage. Let T be the total number of frames for both semantic and acoustic sequences (eg. 30 s $\times$ 25 fps $= 600$), P the patch size, $N = T/P$ the number of patches, and S the number of diffusion steps.

Assuming key-value caching is used during inference, we analyze the leading-order time complexity:

Decoupled 2-stage model:

$$O_0 = L_1 \cdot \frac{(1+T)T}{2} + L_2 \cdot (2T)^2 \cdot S$$

SongBloom:

$$O_1 = L_1 \cdot \frac{(1+T+N)(T+N)}{2} + L_2 \cdot (2P)^2 \cdot N \cdot S$$

Substituting $N = T/P$, we compute the difference:

$$O_1 - O_0 = \left(\frac{L_1}{2P^2} + \frac{L_1}{P} - 4L_2S\right)T^2 + \left(\frac{L_1}{2P} + 4L_2PS\right)T$$

When T is sufficiently large, the T^2 term dominates. In most practical cases, where:

$$L_1 < 4SP \cdot \frac{2P}{2P+1} \cdot L_2$$

the coefficient of T^2 is negative. Therefore, we conclude that SongBloom is asymptotically more efficient than the decoupled two-stage models, owing to its patch-wise diffusion mechanism and reduced per-step input length during inference.

C Criteria of the Subjective Listening Test

1. **Musicality of vocal**: (1–5 points) Does the main melody of the generated vocal match the subjective expectation?

- **5 points**: The melody is pleasant and emotionally expressive, with strong musical phrasing. It aligns well with expectations.
- **4 points**: The melody generally meets expectations and conveys the song's theme and emotion, but lacks standout features.
- **3 points**: The melody mostly aligns with expectations and conveys the theme and emotion, though some notes feel abrupt.
- **2 points**: Only parts of the melody are coherent; most notes are scattered, and the theme and emotion are vaguely presented.
- **1 point**: The melody significantly deviates from expectations, lacks coherent musical phrasing, and fails to convey the song's theme and emotion.

2. **Musicality of accompaniment**: (1–5 points) Does the accompaniment of the generated song sound harmonious?
   - **5 points**: The accompaniment is richly colored and features diverse instrumentation. The melody is beautiful and complements the main melody harmoniously.
   - **4 points**: The accompaniment supports the main melody, but uses limited instrumentation or has a generally average melodic performance.
   - **3 points**: The accompaniment mostly supports the main melody, with only minor discord. However, it sometimes clashes with the main melody and lacks variety and color in instrumentation.
   - **2 points**: Some segments show disorganized instrumentation and monotonous melody, barely supporting the main melody.
   - **1 point**: The instrumentation is chaotic and the melody is discordant. There is a clear conflict with the main melody, failing to provide support.
3. **Quality of vocal**: (1–5 points) Is the vocal in the generated music clear and bright, with a full high-frequency range? Are there any noises or distortions present?
   - **5 points**: The vocal quality is rich and clear, with no noise, approaching studio-recording quality.
   - **4 points**: The vocal quality is relatively clear, with slight noise that is either imperceptible or barely noticeable.
   - **3 points**: The vocal quality contains some noise or distortion, but it does not significantly affect the listening experience.
   - **2 points**: The vocal quality is unclear and unstable, resulting in a poor listening experience. Noticeable noise or distortion is present.
   - **1 point**: The vocal quality is extremely poor, with an unpleasant listening experience, and the vocal characteristics are barely recognizable.
4. **Quality of accompaniment**: (1–5 points) Is the high-frequency range of the generated music's accompaniment full? Are there any noises or instrumental distortions?
   - **5 points**: The accompaniment has a full and clear sound quality with no flaws. The characteristics and melodies of different instruments are clearly distinguishable.
   - **4 points**: The accompaniment has good sound quality with slight noise. Only a few instruments in certain segments are hard to distinguish or slightly distorted, but this does not affect the overall listening experience.
   - **3 points**: The accompaniment has average sound quality. Some instruments are unclear or unidentifiable in certain segments. There is noticeable noise, distortion, or a lack of clarity.
   - **2 points**: The accompaniment has poor sound quality. In most parts of the piece, most instruments are unrecognizable. There is clear noise, distortion, or lack of clarity.
   - **1 point**: The accompaniment has extremely poor sound quality, with severe distortion, making it nearly impossible to identify any instrumental characteristics.
5. **Correctness of lyrics**: (1–4 points) Does the song content match the lyrics? Are there any errors such as extra words, missing words, or mechanical repetition?
   - **4 points**: The song content fully matches the lyrics, with no missing or extra words, and no mechanical repetition of musical segments.

- **3 points**: The generated song contains a small number (within 5 words) of unclear, repeated, or missing lyrics.
- **2 points**: The generated song contains multiple segments with unclear, repeated, or missing lyrics.
- **1 point**: The generated song does not match the lyrics at all.

6. **Consistency of prompt**: (1–5 points) Does the musical style of the generated song match the style of the reference audio prompt?
    - **5 points**: The musical style of the generated song fully matches the style specified in the prompt.
    - **4 points**: The musical style of the generated song is similar to the specified prompt, with only slight differences in some segments.
    - **3 points**: The musical style is somewhat similar to the specified style, but only vaguely reflects its characteristics.
    - **2 points**: The musical style does not resemble the specified style, with only faint traces of the intended musical elements.
    - **1 point**: The musical style has no relation to the specified style, making it difficult to connect the prompt with the resulting music.