MUGI: MULTI-TASK SONG GENERATION WITH JOINT AUTOREGRESSIVE-DIFFUSION TRAINING AND COVER-ORIENTED TOKENIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Song generation has advanced rapidly in recent years, but most existing studies still focus on text-conditioned generation, with limited support for tasks such as cover song generation. Meanwhile, recent systems increasingly adopt a two-stage paradigm in which an autoregressive (AR) language model predicts high-level representations and a diffusion model renders acoustic details. However, these two modules are typically trained separately and connected only through discrete tokens. In this paper, we present **Mugi**, a unified song generation model that supports text-to-song, cover song, and timbre-controllable song generation within a single framework. To support high-fidelity cover song generation, we introduce a **Cover-Tokenizer** that represents melodic information and linguistic content as two parallel token streams, enabling the generated song to faithfully preserve melodic, harmonic, and rhythmic characteristics while maintaining the phoneme-level temporal alignment of the lyrics. Furthermore, Mugi **jointly trains the AR and diffusion stages** by directly conditioning the diffusion model on the continuous hidden states of the AR model, optimizing the two stages together in a single end-to-end training process using both the AR language-modeling loss and the flow-matching objective. Extensive subjective and objective evaluations demonstrate strong performance across multiple song generation tasks, with objective results competitive with commercial systems[1].

1 INTRODUCTION

Song generation has become an increasingly important research topic in generative audio, aiming to synthesize complete songs with vocals and instrumental accompaniment from lyrics and textual descriptions. This task requires models to coordinate linguistic content, vocal performance, and musical structure over extended durations. Recent studies have made substantial progress in the quality and coherence of generated songs, demonstrating the potential of automated song creation (Liu et al., 2025; Lei et al., 2024; Yuan et al., 2025; Lei et al., 2026a; Yang et al., 2026b).

However, most existing studies still focus on text-to-song generation, where a song is created from scratch using textual conditions. Practical music creation often requires more flexible forms of control, such as reinterpreting an existing song or generating songs with a specified vocal timbre. Commercial systems such as Suno already support cover song generation and voice conditioning, extending their capabilities beyond text-to-song generation. These capabilities motivate a unified song generation model that supports text-to-song, cover song, and timbre-controllable generation within a single framework.

At the architectural level, previous studies have explored both diffusion-based models, such as DiffRhythm (Ning et al., 2025) and ACE-Step (Gong et al., 2025), and autoregressive (AR) language models, such as YuE (Yuan et al., 2025) and Khala (Liu et al., 2026). More recent systems increasingly combine the two approaches: an AR language model predicts audio tokens, and a Diffusion Transformer (DiT) renders acoustic details conditioned on these tokens (Lei et al., 2026a;b; Gong et al., 2026; Yang et al., 2026b). However, the two components are typically trained separately

[1] Audio samples are available at the anonymous link: `https://mugi-music.github.io/`

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

and connected only through discrete tokens. Consequently, the acoustic generation objective cannot directly update the AR model, and information passed to the diffusion model is restricted to the discrete token representation. A natural alternative is to connect the two components through continuous latent representations and train them jointly, allowing the AR model to learn representations that support both token prediction and acoustic generation. In image and speech generation (Zhou et al., 2025; Li et al., 2024b; Jia et al., 2025; Guan et al., 2025), joint training of autoregressive models and diffusion models has been explored, which inspires our exploration in song generation.

Beyond the generation architecture, cover song generation poses a distinct challenge in representing the reference song. Existing approaches commonly rely on vocal F0 (Li et al., 2026; Xu et al., 2026) or parallel MIDI and F0 streams (Dai et al., 2026). These features capture the vocal melody but do not explicitly represent the harmonic progression or rhythmic structure of the accompaniment, which also contribute to a song's musical identity. Moreover, preserving the correspondence between lyrics and melody requires information about phonetic content and timing. Recent work addresses this aspect through explicit phoneme-level alignment components (Lee et al., 2026), but a compact representation that captures vocal melody, accompaniment harmony, rhythm, and phoneme-level timing remains needed for cover song generation.

In this paper, we present **Mugi** (**M**ulti-task m**u**sic **g**enerative **i**ntelligence), a song generation model that supports text-to-song, cover song, and timbre-controllable song generation within a single framework. Mugi combines an autoregressive large language model (LLM) with a chunk-wise Diffusion Transformer (Chunk-DiT). The LLM predicts audio tokens, while Chunk-DiT generates acoustic representations conditioned directly on the continuous hidden states of the LLM. The two stages are trained jointly using both the AR language-modeling loss and the flow-matching objective. To support cover song generation, we further introduce a **Cover-Tokenizer** that converts a reference song into two parallel 25-Hz token streams. The melody stream encodes vocal F0, vocal MIDI, chord roots, and downbeats, capturing melodic, harmonic, and rhythmic information. The content stream captures phonetic content and its temporal alignment from the vocal track. Together, these designs enable a unified model to support multiple song generation tasks with flexible conditioning.

The main contributions of this paper can be summarized as follows [2]:

- **Joint AR-diffusion training via continuous connection.** We propose Mugi, a song generation model that jointly trains the autoregressive LLM and diffusion model by connecting them through continuous LLM hidden states rather than discrete audio tokens, allowing the two generation stages to be trained in a single end-to-end procedure.
- **Cover-Tokenizer for melody and content modeling.** We design a Cover-Tokenizer for cover song generation that represents melody and content as two parallel token streams. The melody tokenization module is trained with multiple melodic, harmonic, and rhythmic features, while the content tokenization module extracts phonetic information from vocals.
- **Unified multi-task song generation.** Mugi unifies text-to-song, cover song, and timbre-controllable song generation within a single model. Extensive experiments demonstrate strong performance on both subjective and objective evaluations, with objective results competitive with commercial systems such as Suno-v6-wild.

2 Related work

2.1 Music Generation

Music generation has advanced toward end-to-end audio generation. Early systems such as MusicLM(Agostinelli et al., 2023), MusicGen(Copet et al., 2023) and MusicLDM(Chen et al., 2024) primarily focused on instrumental music, while recent research has increasingly explored song generation with both vocals and accompaniment(Liu et al., 2025; Lei et al., 2024; Yang et al., 2025). Different modeling approaches have been investigated: DiffRhythm (Ning et al., 2025; Chen et al., 2025b; Jiang et al., 2025) and ACE-Step (Gong et al., 2025) use non-autoregressive diffusion models, whereas YuE (Yuan et al., 2025) and Khala (Liu et al., 2026) adopt autoregressive language

[2]To support reproducibility and future research, we will open source our code and model after the review process concludes or upon publication.

models to generate songs. The quality of generated songs is commonly assessed using a range of automatic metrics (Yao et al., 2025; Wu et al., 2026; Ma et al., 2026; Tjandra et al., 2025).

More recent systems combine AR modeling with diffusion acoustic generation. In this paradigm, an autoregressive model predicts audio tokens, and a Diffusion Transformer (DiT) (Peebles & Xie, 2023) reconstructs the audio conditioned on these tokens. Representative systems include LeVo (Lei et al., 2026a;b), ACE-Step 1.5 (Gong et al., 2026) and HeartMuLa (Yang et al., 2026b). These systems typically train the autoregressive and diffusion components separately. In contrast, Mugi connects the two components through continuous hidden states and jointly optimizes them.

In music generation, the closest work to ours is SongBloom (Yang et al., 2026a), which jointly trains autoregressive sketching and diffusion refinement with interleaved semantic and acoustic patches. Unlike SongBloom, Mugi does not interleave semantic and acoustic patches. Instead, it passes the LLM hidden states at each time step to DiT in parallel as continuous conditioning, without providing discrete tokens to DiT.

2.2 Cover Song Generation

Cover song generation aims to reinterpret a reference song's vocal performance and accompaniment while preserving its core melody. Early work conditions generation on explicit melodic cues: SongEcho (Li et al., 2026) and Qwen-Music (Xu et al., 2026) use vocal F0, while Dai et al. (2026) use parallel MIDI and F0 token streams to capture note-level and fine-grained pitch variations. Beyond pitch control, preserving the correspondence between lyrics and melody is also important. MPEcho (Lee et al., 2026) extends SongEcho with a phoneme encoder and a length regulator, improving lyric accuracy through explicit phoneme-level conditioning. Our proposed Cover-Tokenizer jointly represents melodic, harmonic, and rhythmic features, as well as phonetic content.

3 Method

3.1 System Overview

As illustrated in Figure 1, Mugi is a unified model for text-to-song, cover song, and timbre-controllable song generation. The model takes a text caption and lyrics as inputs, together with optional audio references for additional control. All conditions are provided to an autoregressive large language model (LLM), which works with a jointly trained chunk-wise Diffusion Transformer (Chunk-DiT) to generate the song.

The caption and lyrics are tokenized using the LLM's text tokenizer. For timbre control, the voice reference is encoded into a timbre embedding. For cover song generation, a reference song is processed by our proposed Cover-Tokenizer, a tokenizer specifically designed for this task that produces two parallel token streams representing melody and content. Conditioned on these inputs, the LLM generates audio tokens, while Chunk-DiT produces the corresponding acoustic representations. A VAE decoder converts these representations into the output waveform.

Together, these components enable Mugi to support all three tasks within a single framework.

3.2 Joint Training via Continuous Connection

Mugi combines a 3B-parameter autoregressive LLM with a lightweight 0.6B-parameter Chunk-DiT. Discrete audio tokens provide a compact prediction target for the LLM, while its continuous hidden states are passed to Chunk-DiT, so that gradients from the flow-matching objective propagate into the LLM. This differs from prior approaches (Yang et al., 2026b; Lei et al., 2026a; Gong et al., 2026; Dai et al., 2026) that train the two modules separately and connect them through discrete tokens.

Autoregressive modeling. Each training song is represented as a single stream of discrete audio tokens $\mathbf{a} = (a_1, \ldots, a_N)$ at 25 Hz. Given the conditions $\mathcal{C}$ and preceding tokens, the LLM produces a hidden state h_i and predicts the next token:

$$h_i = f_\theta(\mathcal{C}, \mathbf{a}_{<i}), \qquad p_\theta(a_i \mid \mathbf{a}_{<i}, \mathcal{C}) = \mathrm{Softmax}(W_{\mathrm{out}} h_i + b_{\mathrm{out}}), \tag{1}$$

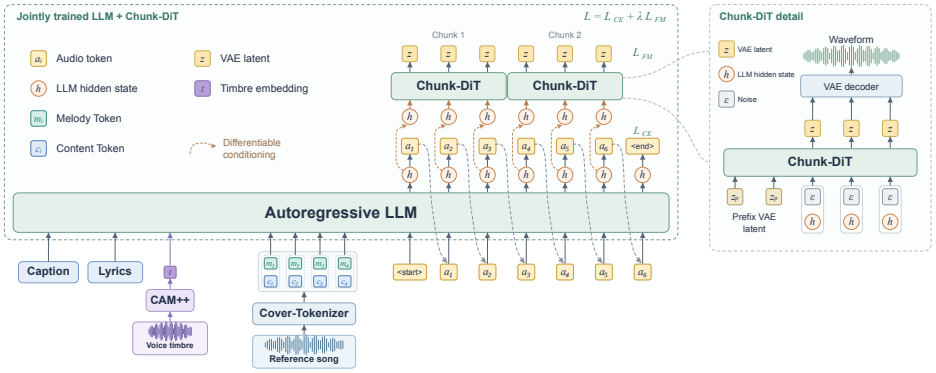


Figure 1: **Overview of Mugi**. Given a text caption, lyrics, an optional voice timbre audio, and an optional reference song, Mugi uses an autoregressive LLM to generate audio tokens, with a jointly trained Chunk-DiT that renders the LLM hidden state to VAE latent.

trained with the language-modeling objective $\mathcal{L}_{\text{CE}} = -\mathbb{E}_{x,\mathcal{C}}\left[\sum_{i=1}^{N} \log p_\theta(a_i \mid \mathbf{a}_{<i}, \mathcal{C})\right]$. To keep the connection differentiable, Chunk-DiT is conditioned on the hidden states rather than sampled tokens (Figure 1).

Chunk-wise acoustic generation. We divide the token sequence into chunks of K tokens ($K/25$ seconds). For chunk j, the conditioning and its acoustic prefix are

$$H^{(j)} = \left(h_{(j-1)K+1}, \ldots, h_{jK}\right), \qquad z^{(j)}_{\text{pre}} = \begin{cases} \varnothing, & j = 1, \\ z^{(j-1)}_{L-P+1:L}, & j > 1, \end{cases} \tag{2}$$

where $z^{(j)}$ denotes the corresponding segment of L VAE latent frames. The prefix consists of the final P frames of the preceding chunk, kept clean as cross-chunk context; during training it is taken from the target latents, and during inference from the previously generated chunk. K and P are specified in the experimental setup.

Following flow matching, Chunk-DiT predicts the velocity of the current chunk conditioned on the hidden states and the prefix:

$$\hat{v}^{(j)} = v_\phi\left(z^{(j)}_\tau, \tau \mid H^{(j)}, z^{(j)}_{\text{pre}}\right), \qquad z^{(j)}_\tau = (1-\tau)\epsilon^{(j)} + \tau z^{(j)}, \tag{3}$$

with the training objective

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{x,\mathcal{C},j,\epsilon,\tau}\left[\left\| v_\phi\left(z^{(j)}_\tau, \tau \mid H^{(j)}, z^{(j)}_{\text{pre}}\right) - \left(z^{(j)} - \epsilon^{(j)}\right)\right\|_2^2\right], \tag{4}$$

computed only over the newly generated frames.

Joint optimization. The two models are jointly trained with $\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda\mathcal{L}_{\text{FM}}$. Since $H^{(j)}$ is passed without detaching gradients, the flow-matching objective updates both models: the LLM learns from discrete token supervision and acoustic generation jointly, while the smaller Chunk-DiT focuses on acoustic rendering.

3.3 Cover-Tokenizer

Motivation. Prior approaches to cover song generation (Li et al., 2026; Dai et al., 2026; Xu et al., 2026) often use vocal melody (e.g., F0 or MIDI) as the control condition, with less explicit consideration of harmonic and rhythmic consistency with the reference. Moreover, vocal melody alone does not specify the correspondence between lyrics and notes: even with lyrics given, the timing and duration of individual phonemes remain unknown, which the phonetic content of the reference vocals can provide. We therefore propose Cover-Tokenizer, which produces two parallel token streams: melody tokens encoding melodic, harmonic, and rhythmic information, and content tokens capturing phonetic content and its timing.

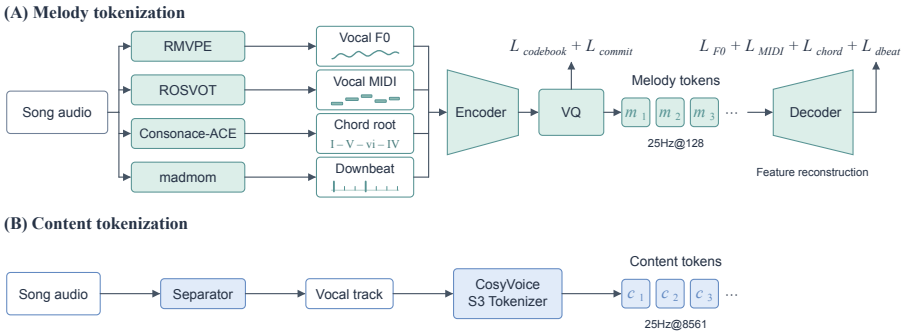


Figure 2: **Overview of the Cover-Tokenizer**, which produces two parallel token streams in 25 Hz.

Melody Tokenization. As illustrated in Figure 2, we extract four complementary features from the reference song and compress them into a single discrete token stream.

- **F0.** We use RMVPE (Wei et al., 2023) to extract the vocal F0 contour, which captures fine-grained pitch variations such as vibrato. Following Dai et al. (2026), we quantize F0 into 128 levels and reconstruct the frame-wise labels with a cross-entropy loss $\mathcal{L}_{\mathrm{F0}}$.
- **Vocal MIDI.** We use ROSVOT (Li et al., 2024a) to extract note-level vocal melody, which complements F0 with a more stable representation of note pitches and durations. The MIDI pitches are mapped to 128 levels and reconstructed with a cross-entropy loss $\mathcal{L}_{\mathrm{MIDI}}$.
- **Chord roots.** We extract chord features with Consonance-ACE (Poltronieri et al., 2025). Since chord qualities and extensions are often tied to particular styles (e.g., seventh chords in Jazz, power chords in Rock), we retain only the roots, preserving the root progression while allowing chord color to adapt to the target style. The reconstruction target has 13 classes (12 roots plus no-chord), with a cross-entropy loss $\mathcal{L}_{\mathrm{chord}}$.
- **Downbeats.** We use madmom (Böck et al., 2016) to extract downbeats, which provide coarse metrical structure while leaving flexibility in local rhythms and accompaniment. Downbeat positions are mapped to a 25-Hz grid as binary labels and reconstructed with a cross-entropy loss $\mathcal{L}_{\mathrm{dbeat}}$.

The four temporally aligned features are jointly encoded at 25 Hz as $\mathbf{e} = E_{\mathrm{mel}}(\mathbf{F})$ and quantized with a single codebook of 128 entries, yielding one melody token per time step. A decoder reconstructs all four feature sequences from the quantized representations $\hat{\mathbf{e}}$, with codebook and commitment losses $\mathcal{L}_{\mathrm{codebook}} = \frac{1}{M}\sum_{i=1}^{M} \|\mathrm{sg}[e_i] - \hat{e}_i\|_2^2$, $\mathcal{L}_{\mathrm{commit}} = \frac{1}{M}\sum_{i=1}^{M} \|e_i - \mathrm{sg}[\hat{e}_i]\|_2^2$, where $\mathrm{sg}[\cdot]$ denotes stop-gradient and M is the sequence length. The overall objective is

$$\mathcal{L}_{\mathrm{mel}} = \mathcal{L}_{\mathrm{F0}} + \mathcal{L}_{\mathrm{MIDI}} + \mathcal{L}_{\mathrm{chord}} + \mathcal{L}_{\mathrm{dbeat}} + \mathcal{L}_{\mathrm{codebook}} + \beta\mathcal{L}_{\mathrm{commit}}, \tag{5}$$

The resulting codebook indices form the melody token stream $\mathbf{m} = (m_1, \ldots, m_M)$ at 25 Hz.

Content Tokenization. We first apply a source separator to the reference song r to obtain the vocal track $v = \mathrm{Sep}(r)$, from which a pretrained CosyVoice S3 tokenizer (Du et al., 2025) extracts content tokens $\mathbf{c} = (c_1, \ldots, c_M)$ at 25 Hz with a vocabulary of 8,561. Following voice conversion studies (Liu, 2024; Chen et al., 2025a), we apply an OpenVoice timbre shifter (Qin et al., 2023) to the vocal track during training to reduce timbre leakage from the reference vocals.

The two 25-Hz streams provide complementary musical and phonetic conditions to the LLM. By default both are used for cover generation; for covers with modified lyrics, only melody tokens are used, so that the generated vocals follow the new lyrics without being constrained by the original phonetic content.

3.4 Voice Timbre Control

Motivated by its widespread use in singing and speech voice conversion (Liu, 2024; Chen et al., 2025a), we use CAM++ (Wang et al., 2023) to extract a voice timbre embedding as a conditioning

input to the LLM. Since CAM++ produces a single embedding without a temporal dimension, this also reduces the risk of leaking melodic information from the voice reference. During data preparation, we separate vocals from each song using BS-Roformer (Lu et al., 2024) and extract CAM++ embeddings from 10-second windows. Each embedding is ranked by its mean cosine similarity to all other embeddings within the same song, and during training we randomly sample one from the top 50% as the timbre condition. This selection reduces the influence of embeddings affected by sustained notes or noise from imperfect vocal separation.

4 Implementation details

4.1 Model configuration and Training Stages

The autoregressive LLM is initialized from Llama-3.2-3B (Grattafiori et al., 2024). Chunk-DiT follows a DiT architecture similar to TangoFlux (Hung et al., 2026) and contains approximately 0.6B parameters. For discrete audio token, we use the Mix Music Codec from LeVo (Lei et al., 2026a), which produces a single token stream at 25 Hz with a codebook size of 16,384. We adopt the VAE from Lei et al. (2026a), which follows the Stable Audio VAE architecture (Evans et al., 2025) and encodes audio at 25 Hz. Chunk-DiT uses a chunk duration of 5 seconds with 2 seconds of prefix context. We apply joint classifier-free guidance (CFG) over the lyrics, caption, timbre embedding, and Cover-Tokenizer tokens, treating the available conditions as a group for conditional and unconditional predictions.

Training consists of two stages: pretraining stage and supervised fine-tuning (SFT) stage. During pretraining, we train Mugi on a large-scale song dataset to learn general song generation capabilities. The LLM and Chunk-DiT are jointly optimized from the beginning of pretraining stage using both the language-modeling loss and the flow-matching objective. The SFT stage proceeds in two steps. First, we jointly fine-tune the LLM and Chunk-DiT on a smaller, curated dataset selected for high musicality. We then perform additional fine-tuning on a filtered set of high-fidelity music recordings to improve perceptual audio quality.

4.2 Dataset and Training Details

We use an in-house dataset, constructed with a preprocessing pipeline and filtering criteria similar to Dai et al. (2026), obtaining approximately 5 million full-length songs for pretraining and 1.5 million songs with high musical quality for SFT. Following a Hi-Res audio filtering procedure similar to that of Xu et al. (2026), we further select 0.2 million high-fidelity songs for the final phase of SFT. Mugi can generate full-length songs up to 5 minutes long. It focuses on English and Chinese but also supports Japanese, Korean, Spanish, and other languages.

We pretrain the model for 20,000 steps with a batch size of 256 and a learning rate of 2×10^{-4}. We use the AdamW optimizer and set the flow-matching loss weight λ to 1. For SFT, we use a batch size of 128 and a learning rate of 5×10^{-5}, training for one epoch. For each song, we sample 32 chunks, resulting in an effective batch size for Chunk-DiT that is 32 times that of the LLM. Training is conducted on 64 GPUs, with FlashAttention (Dao et al., 2022) and FSDP (Zhao et al., 2023) to improve computational and memory efficiency. Throughout training, we mix the three generation tasks and their combinations according to predefined sampling ratios.

5 Experiments Setup

5.1 Benchmark and Metrics

We build three 60-example test sets (balanced English/Chinese) for lyrics-to-song, cover, and timbre-controllable generation; all lyrics come from the MiniMax-Music Lyrics Generation API[3]. For covers, we take 60 source songs from SSLD-200 (Tan et al., 2025) with captions specifying a different target style. For timbre control, 3-15s vocal clips from MIR-1K (Hsu & Jang, 2009) and NUS-48E (Duan et al., 2013) serve as prompts. Primary metrics are SongBench (Wu et al., 2026), Audiobox Aesthetics (Tjandra et al., 2025), MuQ-MuLan (Zhu et al., 2025), and PER via

[3]https://platform.minimax.cn/docs/api-reference/lyrics-generation

Qwen-3-ASR (Shi et al., 2026). Covers are further evaluated on melodic consistency using DTW-aligned RPA/RCA with `mir_eval` (Raffel et al., 2014) plus MIDI pitch MAE; timbre control is measured by Resemblyzer (Wan et al., 2018) cosine similarity between generated vocals and the prompt. Detailed metric descriptions are provided in the appendix.

5.2 Baselines systems

We compare Mugi with open-source models, including ACE-Step 1.5(Gong et al., 2026), HeartMuLa-oss-3B (Yang et al., 2026b), LeVo 2 (Lei et al., 2026b), MiniMax-Music3 (MiniMax) and other models(Li et al., 2026), as well as commercial systems, including Suno-v6, Suno-v6-wild[4], and Mureka-V9[5]. We also conduct human evaluations to evaluate the musicality, audio quality and caption following capability of these systems. Detailed descriptions of the baseline systems and human evaluation protocol are provided in the appendix.

Table 1: Subjective evaluation results (MOS) of text-to-song generation, scored on a 5-point scale. **Params.** refers to the parameter count of the model's generation components (LLM and DiT).

Model	Params. (LLM+DiT)	Mean	Musicality	Audio Quality	Adherence
Suno-v6	–	**3.87**	3.87	**3.80**	3.93
Mureka-V9	–	3.86	**3.90**	3.70	**3.98**
Suno-v6-wild	–	3.78	3.73	3.71	3.90
MiniMax-Music3	11B (8.6B+2.4B)	3.55	3.42	3.54	3.70
Mugi (Ours)	**3.6B (3.0B+0.6B)**	3.50	3.39	3.49	3.61
ACE-Step-1.5 (Gong et al., 2026)	8.0B (4.0B+4.0B)	3.43	3.43	3.33	3.55
LeVo2 (Lei et al., 2026b)	4.6B (4.0B+0.6B)	3.40	3.51	3.42	3.25
HeartMuLa (Yang et al., 2026b)	4.8B (3.3B+1.5B)	3.21	3.28	3.32	3.04

6 Results

6.1 Main Results

6.1.1 Text to Song Generation

Table 2 presents the objective results for text-to-song generation. Mugi achieves the highest SongBench mean score (6.53) among open-source models, surpassing the strongest baseline, MiniMax-Music3, by 0.12 points, and leads in six of the seven dimensions, particularly in arrangement and instrumentation. It also attains the best MuQ-MuLan similarity among open-source models, indicating strong musical quality and text adherence. Compared with commercial systems, Mugi outperforms Suno-v6-wild on the SongBench mean and all four AudioBox Aesthetics metrics, reaching the highest CU, PC, and PQ scores overall. Against Suno-v6 and Mureka-V9, Mugi achieves better PER but still trails slightly on overall SongBench performance.

Table 1 summarizes human ratings averaged across 3 dimensions. Mugi ranks second among open-source models, approaching MiniMax-Music3 with only 3.6B parameters versus its 11B, showing competitive perceptual quality at a smaller size, though a gap to commercial systems remains.

6.1.2 Cover Song Generation

As shown in Table 3, Mugi achieves the lowest MIDI MAE and the highest F0 RPA and RCA among all evaluated models, demonstrating its ability to preserve the reference melody during cover generation. Mugi also outperforms the open-source baselines across all evaluated metrics. Compared with the commercial MiniMax-Music-Cover system, it matches the SongBench score and achieves better AudioBox Aesthetics and MuQ-MuLan scores, in addition to stronger melodic consistency. These results demonstrate that Mugi combines faithful melody preservation with high-quality, style-controlled cover generation.

[4]https://suno.com/

[5]https://mureka.ai/

Table 2: Song generation results. **Bold** indicates the best result across all models, while underline indicates the best result among open-source models.

Model	SongBench ↑								AudioBox ↑				MuQ ↑	ASR ↓
	Mean	Mel.	Arr.	Mus.	Voc.	Ins.	Mix.	Str.	CE	CU	PC	PQ	Sim.	PER
Commercial Systems (Closed-source)														
Suno-v6	6.74	6.82	6.93	5.96	7.03	6.98	6.83	6.65	7.55	7.82	6.49	8.14	0.449	0.076
Suno-v6-wild	6.45	6.47	6.63	5.63	6.80	6.74	6.44	6.42	7.63	7.83	6.25	8.27	**0.463**	0.043
Mureka-V9	**6.94**	**6.98**	**7.16**	**6.07**	**7.31**	**7.12**	**6.93**	**7.01**	7.53	7.73	6.75	8.08	0.365	0.167
Open-source Systems														
ACE-Step-1.5	6.09	6.25	6.29	5.29	6.37	6.28	6.07	6.05	7.59	7.80	6.66	8.23	0.381	**0.032**
HeartMuLa	6.28	6.35	6.33	5.55	6.80	6.46	6.33	6.12	**7.81**	7.92	6.55	8.31	0.352	0.109
LeVo2	6.30	6.38	6.50	5.47	6.60	6.63	6.35	6.18	7.72	7.92	6.54	8.38	0.280	0.130
MiniMax-Music3	6.41	6.43	6.58	5.53	6.87	6.68	6.48	6.31	7.63	7.90	6.53	8.31	0.357	0.054
Mugi(Ours)	6.53	6.55	6.89	5.75	6.62	6.89	6.62	6.42	7.72	**7.99**	**6.80**	**8.41**	0.391	0.064

Table 3: Cover song generation results. Identity denotes the original reference songs without modification. The results for MiniMax-Music-Cover were obtained through its API, as its cover generation model is not open-source.

Model	SongBench ↑	AudioBox ↑	MuQ ↑	ASR ↓	Melody		
	Mean	Mean	Sim.	PER	MIDI MAE ↓	F0 RPA ↑	F0 RCA ↑
Identity	6.27	7.32	0.138	0.228	—		
SongEcho	4.21	6.98	0.159	0.419	1.03	0.638	0.641
ACE-Step-1.5	4.55	6.82	0.165	0.338	2.28	0.318	0.322
MiniMax-Music-Cover	**6.14**	7.46	0.252	**0.258**	1.03	0.585	0.587
Mugi(Ours)	**6.14**	**7.54**	**0.263**	0.279	**0.62**	**0.724**	**0.726**

6.1.3 Timbre-controllable Song Generation

The results in Table 4 demonstrate Mugi's ability to preserve the target vocal timbre while generating songs. Mugi achieves the joint-highest speaker similarity with ACE-Step-1.5, indicating comparable fidelity to the reference voice. It also obtains the highest SongBench mean score among the evaluated systems, with competitive AudioBox Aesthetics performance. These results demonstrate effective timbre control alongside strong musical quality, although there remains a gap to ACE-Step-1.5 in caption alignment.

Table 4: Timbre-controllable song generation results.

Model	SongBench ↑	AudioBox ↑	MuQ ↑	ASR ↓	Timbre ↑
	Mean	Mean	Sim.	PER	SSIM
ACE-Step-1.5	6.02	7.51	**0.340**	**0.049**	**0.78**
LeVo2	6.02	**7.57**	0.263	0.095	0.74
Mugi(Ours)	**6.06**	7.54	0.295	0.072	**0.78**

6.2 Ablation Studies

6.2.1 Ablation on AR-diffusion Connection and Training

We ablate to examine the effects of continuous conditioning and joint training. We compare three configurations: (1) *discrete tokens*, where Chunk-DiT receives audio-token embeddings rather than LLM hidden states, and the two models are optimized separately; (2) *detached continuous latents*, where Chunk-DiT receives continuous LLM hidden states, but gradients are stopped at this connection, so the flow-matching loss does not update the LLM; and (3) *joint continuous latents*, our proposed method, which allows gradients to pass through the hidden states and jointly updates both models. All variants are trained for 15,000 steps on the same setup.

As shown in Table 5, both continuous-latent variants outperform discrete-token conditioning on SongBench and MuQ-MuLan, supporting the benefit of retaining continuous contextual information for acoustic generation. Compared with detached-latent conditioning, joint training further improves SongBench and substantially reduces PER, while AudioBox and MuQ-MuLan scores remain nearly unchanged. These results suggest that continuous conditioning and joint optimization

offer complementary benefits: the former improves the information passed to Chunk-DiT, while the latter allows acoustic supervision to refine the LLM representations.

Table 5: Ablation on AR-diffusion connection and training. Connection specifies the conditioning input to DiT. Joint Training indicates whether gradients from DiT propagate into the LLM.

Configuration		SongBench ↑	AudioBox ↑	MuQ-MuLan ↑	ASR ↓
Connection	**Joint Training**	**Mean**	**Mean**	**Sim.**	**PER**
Discrete tokens	No	6.27	**7.72**	0.317	0.098
Continuous latents	No	6.55	7.71	**0.400**	0.110
Continuous latents	Yes	**6.63**	**7.72**	0.398	**0.075**

6.2.2 Ablation on Cover-Tokenizer design

We ablate the Cover-Tokenizer to examine how different melody features and the addition of content tokens affect cover generation. To keep the study affordable, all ablation runs use a setup comparable to SFT: every variant starts from the same pretrained checkpoint without any cover-task pretraining, and is fine-tuned on the same data for cover generation with batch size 128 for 5,000 steps.

As Table 6 shows, adding MIDI, chord roots, and downbeats one by one consistently improves all three melody metrics. Incorporating content tokens further strengthens melodic fidelity and brings a clear drop in PER, indicating that explicit phonetic timing helps the model follow the reference more faithfully. At the same time, SongBench scores slip slightly as more control signals are added, revealing a trade-off that tighter reference adherence comes at some cost to musical aesthetics.

Table 6: Ablation study of Cover-Tokenizer for cover song generation. The Melody Token column lists the features used for melody tokenization. The Content Token column indicates whether content tokens are included (Yes) or excluded (No).

Token Configuration		SongBench ↑	ASR ↓	Melody		
Melody Token	**Content Token**	**Mean**	**PER**	**MIDI MAE ↓**	**F0 RPA ↑**	**F0 RCA ↑**
F0	No	**6.35**	0.383	0.99	0.626	0.628
F0+MIDI	No	6.33	0.346	0.80	0.661	0.663
F0+MIDI+Chord+Downbeat	No	6.32	0.325	0.79	0.673	0.676
F0+MIDI+Chord+Downbeat	Yes	6.29	**0.282**	**0.66**	**0.719**	**0.721**

7 Discussion

Our Cover-Tokenizer uses chord roots to preserve the reference song's harmonic progression in the accompaniment. While this encourages harmonic consistency, it also limits the freedom to reharmonize the song. Whether, and to what extent, a cover should retain the original harmony remains an open question: faithful reproduction and creative reinterpretation may call for different levels of harmonic control. Future work will explore more flexible harmonic conditioning, as well as incorporating accompaniment melody as additional conditions for cover song generation. Incorporating preference alignment methods such as direct preference optimization (DPO) and reinforcement learning also remains a promising direction for future work.

8 Conclusion

In this paper, we present Mugi, a unified model for text-to-song, cover song, and timbre-controllable song generation. Mugi jointly trains an autoregressive LLM and Chunk-DiT through continuous hidden-state conditioning, combining discrete audio-token supervision with a flow-matching objective. We also introduced Cover-Tokenizer, which represents melodic, harmonic, and rhythmic information alongside a parallel stream of phonetic content tokens. Experiments demonstrate strong performance across all three tasks, with results competitive with commercial systems on multiple objective metrics.

AI USE STATEMENT

In this work, we used AI for generating part of the training data: the lyric annotations and captions were produced with the Qwen-Omni API. We also used generative AI tools for assisting with the implementation of part of the experimental code, without any involvement in the model architecture or method design, and We have not used generative AI tools for proposing or refining research hypotheses, developing theoretical models or conceptual frameworks, designing the methodology or experiments, or interpreting the results, and the rest of the tasks with required disclosure (formulating or proving mathematical claims, assisting with translation, and supporting qualitative data analysis) are not applicable to this work. Additionally, we used generative AI tools for editing the manuscript to improve readability and English writing quality, and for creating and modifying scientific figures. We have reviewed all AI-assisted work. Specifically, all AI-edited text was manually checked by the authors to ensure that it faithfully preserves the intended technical meaning, all AI-assisted figures were verified against our model design for correctness, all AI-assisted code was reviewed and tested for correctness by the authors, and the API-generated lyric annotations and captions were inspected and filtered by the authors before use. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our study includes human listening tests to assess the quality of generated songs. All participants were informed of the study's purpose and evaluation procedures, provided informed consent, and received compensation for their participation.

Mugi is intended to support music creation and research. Its ability to generate cover songs and reproduce vocal timbres raises concerns about copyright, consent, and potential impersonation. Users should obtain appropriate authorization for reference songs and voice recordings, respect the rights of creators and performers, and clearly disclose AI-generated content when sharing it. The model should not be used to impersonate individuals without consent or falsely attribute generated music to real artists. We will consider these limitations and potential misuse when planning the release of code and model checkpoints.

REPRODUCIBILITY STATEMENT

We will open-source our code and model checkpoints after the review period to support reproducibility and future research. The Method section 3 describes Mugi's architecture, joint training objectives, and conditioning mechanisms. The Implementation Details section 4 reports the model configuration, data preparation, training stages, and hyperparameters.

REFERENCES

Andrea Agostinelli, Timo I Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, et al. Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325*, 2023.

Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. Madmom: A new python audio and music signal processing library. In *Proceedings of the 24th ACM international conference on Multimedia*, pp. 1174–1178, 2016.

Gongyu Chen, Xiaoyu Zhang, Zhenqiang Weng, Junjie Zheng, Da Shen, Chaofan Ding, Wei-Qiang Zhang, and Zihao Chen. Yingmusic-svc: Real-world robust zero-shot singing voice conversion with flow-grpo and singing-specific inductive biases. *arXiv preprint arXiv:2512.04793*, 2025a.

Huakang Chen, Yuepeng Jiang, Guobin Ma, Chunbo Hao, Shuai Wang, Jixun Yao, Ziqian Ning, Meng Meng, Jian Luan, and Lei Xie. Diffrhythm+: Controllable and flexible full-length song generation with preference optimization. In *2025 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 1–8. IEEE, 2025b.

Ke Chen, Yusong Wu, Haohe Liu, Marianna Nezhurina, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Musicldm: Enhancing novelty in text-to-music generation using beat-synchronous mixup

strategies. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1206–1210. IEEE, 2024.

Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation. *Advances in neural information processing systems*, 36:47704–47720, 2023.

Junyu Dai, Xinyue Fan, Weiqin Li, Xiangang Li, Yunjia Li, Bin Ma, Yukun Ma, Chongjia Ni, Yufei Shi, Biao Tian, et al. Pushing the frontier of full-song generation: Hierarchical autoregressive planning meets flow-matching rendering. *arXiv preprint arXiv:2607.20253*, 2026.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

P Kingma Diederik and Welling Max. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

Zhihao Du, Changfeng Gao, Yuxuan Wang, Fan Yu, Tianyu Zhao, Hao Wang, Xiang Lv, Hui Wang, Chongjia Ni, Xian Shi, et al. Cosyvoice 3: Towards in-the-wild speech generation via scaling-up and post-training. *arXiv preprint arXiv:2505.17589*, 2025.

Zhiyan Duan, Haotian Fang, Bo Li, Khe Chai Sim, and Ye Wang. The nus sung and spoken lyrics corpus: A quantitative comparison of singing and speech. In *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pp. 1–9. IEEE, 2013.

Zach Evans, Julian D Parker, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons. Stable audio open. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025.

Junmin Gong, Sean Zhao, Sen Wang, Shengyuan Xu, and Joe Guo. Ace-step: A step towards music generation foundation model. *arXiv preprint arXiv:2506.00045*, 2025.

Junmin Gong, Yulin Song, Wenxiao Zhao, Sen Wang, Shengyuan Xu, Jing Guo, and Xuerui Yang. Ace-step 1.5: Pushing the boundaries of open-source music generation. *arXiv preprint arXiv:2602.00744*, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Wenhao Guan, Zhikang Niu, Ziyue Jiang, Kaidi Wang, Peijie Chen, Qingyang Hong, Lin Li, and Xie Chen. Univoice: Unifying autoregressive asr and flow-matching based tts with large language models. *arXiv preprint arXiv:2510.04593*, 2025.

Yiwei Guo, Zhihan Li, Hankun Wang, Bohan Li, Chongtian Shao, Hanglei Zhang, Chenpeng Du, Xie Chen, Shujie Liu, and Kai Yu. Recent advances in discrete speech tokens: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Chao-Ling Hsu and Jyh-Shing Roger Jang. On the improvement of singing voice separation for monaural recordings using the mir-1k dataset. *IEEE transactions on audio, speech, and language processing*, 18(2):310–319, 2009.

Chia-Yu Hung, Navonil Majumder, Zhifeng Kong, Ambuj Mehrish, Amir Zadeh, Chuan Li, Rafael Valle, Bryan Catanzaro, and Soujanya Poria. Tangoflux: Super fast and faithful text to audio generation with flow matching and clap-ranked preference optimization. In *International Conference on Learning Representations*, volume 2026, pp. 150793–150816, 2026.

Dongya Jia, Zhuo Chen, Jiawei Chen, Chenpeng Du, Jian Wu, Jian Cong, Xiaobin Zhuang, Chumin Li, Zhen Wei, Yuping Wang, et al. Ditar: Diffusion transformer autoregressive modeling for speech generation. *arXiv preprint arXiv:2502.03930*, 2025.

Yuepeng Jiang, Huakang Chen, Ziqian Ning, Jixun Yao, Zerui Han, Di Wu, Meng Meng, Jian Luan, Zhonghua Fu, and Lei Xie. Diffrhythm 2: Efficient and high fidelity song generation via block flow matching. *arXiv preprint arXiv:2510.22950*, 2025.

Wei-Jaw Lee, Hsuan-Yu Yeh, Ting-Yi Hu, Chih-Pin Tan, Fang-Duo Tsai, and Yi-Hsuan Yang. Mpecho: A melody and phoneme-aware generative framework for controllable cover song generation. *arXiv preprint arXiv:2607.26698*, 2026.

Shun Lei, Yixuan Zhou, Boshi Tang, Max W Lam, Feng Liu, Hangyu Liu, Jingcheng Wu, Shiyin Kang, Zhiyong Wu, and Helen Meng. Songcreator: Lyrics-based universal song generation. *Advances in Neural Information Processing Systems*, 37:80107–80140, 2024.

Shun Lei, Yaoxun Xu, Huaicheng Zhang, Hangting Chen, Yixuan Zhang, Chenyu Yang, Haina Zhu, Shuai Wang, Zhiyong Wu, Dong Yu, et al. Levo: High-quality song generation with multi-preference alignment. *Advances in Neural Information Processing Systems*, 38:102448–102479, 2026a.

Shun Lei, Huaicheng Zhang, Dapeng Wu, Yaoxun Xu, Lishi Zuo, Wei Tan, Hangting Chen, Guangzheng Li, Jianwei Yu, Zhiyong Wu, et al. Levo 2: Stable and melodious song generation via hierarchical representation modeling and progressive post-training. *arXiv preprint arXiv:2606.30642*, 2026b.

Ruiqi Li, Yu Zhang, Yongqi Wang, Zhiqing Hong, Rongjie Huang, and Zhou Zhao. Robust singing voice transcription serves synthesis. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9751–9766, 2024a.

Sifei Li, Yang Li, Zizhou Wang, Yuxin Zhang, Fuzhang Wu, Oliver Deussen, Tong-Yee Lee, and Weiming Dong. Songecho: Towards cover song generation via instance-adaptive element-wise linear modulation. *arXiv preprint arXiv:2602.19976*, 2026.

Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems*, 37: 56424–56445, 2024b.

Jiafeng Liu, Yuanliang Dong, Hongjia Liu, Yuqing Cheng, Zhancheng Guo, Huijing Liang, Wenbo Zhan, Yuming Sun, Xiaobing Li, Feng Yu, et al. Khala: Scaling acoustic token language models toward high-fidelity music generation. *arXiv preprint arXiv:2605.01790*, 2026.

Songting Liu. Zero-shot voice conversion with diffusion transformers. *arXiv preprint arXiv:2411.09943*, 2024.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Zihan Liu, Shuangrui Ding, Zhixiong Zhang, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Dahua Lin, and Jiaqi Wang. Songgen: A single stage auto-regressive transformer for text-to-song generation. *arXiv preprint arXiv:2502.13128*, 2025.

Wei-Tsung Lu, Ju-Chiang Wang, Qiuqiang Kong, and Yun-Ning Hung. Music source separation with band-split rope transformer. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 481–485. IEEE, 2024.

Yinghao Ma, Haiwen Xia, Hewei Gao, Weixiong Chen, Yuxin Ye, Yuchen Yang, Sungkyun Chang, Mingshuo Ding, Yizhi Li, Ruibin Yuan, Simon Dixon, and Emmanouil Benetos. CMI-RewardBench: Evaluating music reward models with compositional multimodal instruction. *arXiv preprint arXiv:2603.00610*, 2026. URL https://arxiv.org/abs/2603.00610.

MiniMax. MiniMax Music 3. https://github.com/MiniMax-AI/MiniMax-Music3. Accessed: September 25, 2026.

Ziqian Ning, Huakang Chen, Yuepeng Jiang, Chunbo Hao, Guobin Ma, Shuai Wang, Jixun Yao, and Lei Xie. Diffrhythm: Blazingly fast and embarrassingly simple end-to-end full-length song generation with latent diffusion. *arXiv preprint arXiv:2503.01183*, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Andrea Poltronieri, Xavier Serra, and Martín Rocamora. From discord to harmony: Decomposed consonance-based training for improved audio chord estimation. *arXiv preprint arXiv:2509.01588*, 2025.

Zengyi Qin, Wenliang Zhao, Xumin Yu, and Xin Sun. Openvoice: Versatile instant voice cloning. *arXiv preprint arXiv:2312.01479*, 2023.

Colin Raffel, Brian McFee, Eric J Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel PW Ellis, and C Colin Raffel. Mir_eval: A transparent implementation of common mir metrics. In *ISMIR*, volume 10, pp. 2014, 2014.

Xian Shi, Xiong Wang, Zhifang Guo, Yongqi Wang, Pei Zhang, Xinyu Zhang, Zishan Guo, Hongkun Hao, Yu Xi, Baosong Yang, et al. Qwen3-asr technical report. *arXiv preprint arXiv:2601.21337*, 2026.

Wei Tan, Shun Lei, Huaicheng Zhang, Guangzheng Li, Yixuan Zhang, Hangting Chen, Jianwei Yu, Rongzhi Gu, and Dong Yu. Songprep: A preprocessing framework and end-to-end model for full-song structure parsing and lyrics transcription. *arXiv preprint arXiv:2509.17404*, 2025.

Andros Tjandra, Yi-Chiao Wu, Baishan Guo, John Hoffman, Brian Ellis, Apoorv Vyas, Bowen Shi, Sanyuan Chen, Matt Le, Nick Zacharov, Carleigh Wood, Ann Lee, and Wei-Ning Hsu. Meta audiobox aesthetics: Unified automatic quality assessment for speech, music, and sound. *arXiv preprint arXiv:2502.05139*, 2025.

Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno. Generalized end-to-end loss for speaker verification. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4879–4883. IEEE, 2018.

Hui Wang, Siqi Zheng, Yafeng Chen, Luyao Cheng, and Qian Chen. Cam++: A fast and efficient network for speaker verification using context-aware masking. *arXiv preprint arXiv:2303.00332*, 2023.

Haojie Wei, Xueke Cao, Tangpeng Dan, and Yueguo Chen. Rmvpe: A robust model for vocal pitch estimation in polyphonic music. *arXiv preprint arXiv:2306.15412*, 2023.

Dapeng Wu, Shun Lei, Wei Tan, Guangzheng Li, Yunzhe Wang, Huaicheng Zhang, Lishi Zuo, and Zhiyong Wu. SongBench: A fine-grained multi-aspect benchmark for song quality assessment. *arXiv preprint arXiv:2604.25937*, 2026.

Jin Xu, Kangdi Wang, Ruibin Yuan, Shun Lei, Xiong Wang, Xize Cheng, Xueyao Zhang, Yang Zhang, Yiheng Chen, Yongqi Wang, et al. Qwen-music technical report. *arXiv preprint arXiv:2607.11699*, 2026.

Chenyu Yang, Shuai Wang, Hangting Chen, Jianwei Yu, Wei Tan, Rongzhi Gu, Yaoxun Xu, Yizhi Zhou, Haina Zhu, and Haizhou Li. Songeditor: Adapting zero-shot song generation language model as a multi-task editor. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25597–25605, 2025.

Chenyu Yang, Shuai Wang, Hangting Chen, Wei Tan, Jianwei Yu, and Haizhou Li. Songbloom: Coherent song generation via interleaved autoregressive sketching and diffusion refinement. *Advances in Neural Information Processing Systems*, 38:34256–34277, 2026a.

Dongchao Yang, Yuxin Xie, Yuguo Yin, Zheyu Wang, Xiaoyu Yi, Gongxi Zhu, Xiaolong Weng, Zihan Xiong, Yingzhe Ma, Dading Cong, et al. Heartmula: A family of open sourced music foundation models. *arXiv preprint arXiv:2601.10547*, 2026b.

Jixun Yao, Guobin Ma, Huixin Xue, Huakang Chen, Chunbo Hao, Yuepeng Jiang, Haohe Liu, Ruibin Yuan, Jin Xu, Wei Xue, Hao Liu, and Lei Xie. SongEval: A benchmark dataset for song aesthetics evaluation. *arXiv preprint arXiv:2505.10793*, 2025.

Ruibin Yuan, Hanfeng Lin, Shuyue Guo, Ge Zhang, Jiahao Pan, Yongyi Zang, Haohe Liu, Yiming Liang, Wenye Ma, Xingjian Du, et al. Yue: Scaling open foundation models for long-form music generation. *arXiv preprint arXiv:2503.08638*, 2025.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model. In *International Conference on Learning Representations*, volume 2025, pp. 6446–6469, 2025.

Haina Zhu, Yizhi Zhou, Hangting Chen, Jianwei Yu, Ziyang Ma, Rongzhi Gu, Yi Luo, Wei Tan, and Xie Chen. Muq: Self-supervised music representation learning with mel residual vector quantization. *IEEE Transactions on Audio, Speech and Language Processing*, 2025.

A Appendix

A.1 Task Formulation and Preliminaries

A.1.1 Task Formulation

We consider conditional song generation: generating a song waveform x containing both vocals and accompaniment (Liu et al., 2025; Lei et al., 2026a), conditioned on a caption d describing musical attributes (genre, mood, instrumentation), lyrics ℓ, an optional reference song r, and an optional voice reference s. Denoting the conditions by $\mathcal{C} = (d, \ell, r, s)$, with $\varnothing$ for an unavailable reference, we model

$$p(x \mid \mathcal{C}) = p(x \mid d, \ell, r, s), \tag{6}$$

which accommodates three tasks: (1) **Text-to-song generation.** Given $\mathcal{C} = (d, \ell, \varnothing, \varnothing)$, the model generates a song that follows the caption and sings the lyrics. (3) **Cover song generation.** Given an additional reference song r, the model generates a new rendition preserving its melodic and lyrical content while allowing the caption to restyle the arrangement. (3) **Timbre-controllable song generation.** Given a voice reference s, the model generates vocals with the desired timbre.

A.1.2 Preliminaries

Autoregressive audio token modeling. An audio tokenizer maps a waveform x to discrete audio tokens $\mathbf{a} = (a_1, \ldots, a_N)$ (Guo et al., 2025). An autoregressive model factorizes their distribution as

$$p_\theta(\mathbf{a} \mid \mathcal{C}) = \prod_{i=1}^{N} p_\theta(a_i \mid \mathbf{a}_{<i}, \mathcal{C}), \qquad \mathcal{L}_{\mathrm{CE}} = -\mathbb{E}_{x,\mathcal{C}} \left[\log p_\theta(\mathbf{a} \mid \mathcal{C}) \right], \tag{7}$$

where $\mathcal{C}$ denotes conditioning, $\mathbf{a}_{<i}$ the preceding tokens, and the sequence is terminated by an end-of-sequence token. Training minimizes $\mathcal{L}_{\mathrm{CE}}$. We denote by h_i the hidden state used to predict a_i; it serves as the continuous representation of the audio context.

Flow matching diffusion model. Rectified flow-matching learns a velocity field that transports noise to data along a prescribed probability path (Liu et al., 2022). Let z denote the VAE representation (Diederik & Max, 2019) of a song waveform and u the conditioning features. Given noise $\epsilon \sim \mathcal{N}(0, I)$ and time $\tau \sim \mathcal{U}[0, 1]$, we construct $z_\tau = (1 - \tau)\epsilon + \tau z$. The velocity predictor v_ϕ is trained to match the target velocity $z - \epsilon$:

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{z,u,\epsilon,\tau} \left[\| v_\phi(z_\tau, \tau \mid u) - (z - \epsilon) \|_2^2 \right]. \tag{8}$$

At inference, we integrate the learned velocity field from Gaussian noise and decode the resulting representation into a waveform using the VAE decoder.

A.2 FEATURE COMPARISON

In Table 7, a feature comparison between existing commercial and open-source song generation models with Mugi is shown. Mugi unifies text-to-song, cover song, and timbre-controllable song generation, with timbre-specific control.

Table 7: Feature comparison between existing commercial and open-source song generation models with ours. △ indicates partial support: the model accepts an audio prompt as a reference input, but the reference conditions the overall style rather than providing timbre-specific control.

		Text to Song Generation	Cover Song Generation	Timbre Control
Commercial	Suno	✓	✓	✓
	Mureka	✓	✓	✓
	MiniMax-Music (API version)	✓	✓	✗
Open-source	MiniMax-Music (Open-source version)	✓	✗	✗
	SongBloom (Yang et al., 2026a)	✓	✗	△
	LeVo 2 (Lei et al., 2026b)	✓	✗	△
	HeartMuLa (Yang et al., 2026b)	✓	✗	△
	ACE-Step 1.5 (Gong et al., 2026)	✓	✓	✓
	Mugi (Ours)	✓	✓	✓

A.3 BASELINE SYSTEMS

We compare Mugi against the following open-source and commercial systems.

Open-source models.

- **ACE-Step 1.5** (Gong et al., 2026). An open-source music generation foundation model that combines a language model with a diffusion-based acoustic decoder. Version 1.5 supports more than 50 languages and is trained with intrinsic reinforcement learning objectives. We use its open-source weights and follow the official inference settings of `ACE-Step-1.5-xl-sft`.
- **HeartMuLa-oss-3B** (Yang et al., 2026b). An open-source song generation model with a 3B-parameter autoregressive language model over discrete codec tokens, trained in four stages including direct preference optimization. We evaluate the official 3B checkpoint.
- **LeVo 2** (Lei et al., 2026b). An open-source song generation system that models mixed and separated vocal and accompaniment tracks, with progressive direct preference optimization for lyric alignment, prompt consistency, and musicality.
- **MiniMax-Music3** (MiniMax). A closed-source commercial model accessed through its official API. We include it among the open-source comparisons only where its API supports the corresponding conditioning inputs.
- **SongEcho** (Li et al., 2026). An open-source cover song generation baseline that extracts melodic conditions from the reference song. We include it in the cover generation comparison.

Commercial systems.

- **Suno-v6** and **Suno-v6-wild**[6]. Commercial song generation systems accessed via the official website on September 17, 2026, where Suno-v6-wild is a sub-version of Suno-v6.
- **Mureka-V9**[7]. A commercial song generation system accessed via the official API on September 17, 2026.

[6] https://suno.com/
[7] https://mureka.ai/

For each task, we evaluate only the baselines that support the required inputs: lyrics-to-song generation is evaluated on all systems, cover song generation on systems accepting a reference song, and timbre-controllable generation on systems accepting a voice prompt. Commercial systems are used in their default settings with the same captions and lyrics as our model.

A.4 Melody Token in Cover-Tokenizer

The Cover-Tokenizer compresses four frame-level symbolic melody streams (25 Hz), vocal F0 (128 bins), vocal MIDI notes (128 bins), chord labels (13 classes including a "no-chord" token), and downbeat indicators (binary), into a single discrete sequence, i.e., the melody token.

Architecture. Each stream is mapped to a 64-dimensional embedding and concatenated (256 d) per frame. The encoder consists of a convolutional stem (a kernel-7 Conv1d followed by two residual blocks), learned absolute positional embeddings over a 750-frame (30 s) window, and a 4-layer pre-LN Transformer encoder (256 hidden units, 8 attention heads, $4\times$ FFN expansion, GELU activation, dropout 0.1). A 1×1 convolution projects the hidden states to the 64-dimensional code space, followed by a single VQ codebook of size $K{=}128$ trained with EMA updates (decay 0.99, commitment cost 0.25, dead-code restart). The decoder mirrors the encoder: a 1×1 projection, two residual blocks, and a kernel-7 Conv1d, after which four linear heads reconstruct the four input symbol streams with cross-entropy losses. The whole model contains only 5.3M parameters.

Training. The model is trained as a pure reconstruction autoencoder on 180,000 full songs, for 30,000 steps on 8 GPUs. The global batch contains 256 windows of 750 frames (8 records per GPU, 4 windows sampled per record). We use AdamW with a peak learning rate of 3×10^{-4}, 1,000 warmup steps, and cosine decay. Songs are split into non-overlapping 30 s windows with zero-padding at the tail; padded frames are excluded from the loss. As shown in 3, at convergence the Melody Token reconstructs the four streams with accuracies of 0.982 (F0), 0.983 (MIDI), 0.993 (chord), and 0.999 (downbeat) on a held-out set, i.e., the discrete VQ is nearly lossless for melody information.

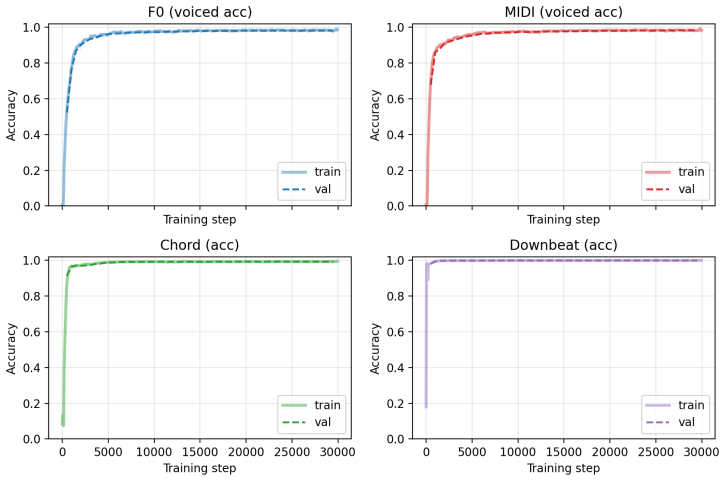


Figure 3: Visualization of accuracy curves for F0, MIDI, chord root, and downbeat during Melody Tokenizer training.

A.5 Data distribution

The pretrain dataset of Mugi is a large in-house corpus that contains approximately 5 million full-length songs. Tracks are filtered for audio quality and de-duplicated at the track level, which mitigates leakage between pre-training and downstream evaluation. We characterize its two most salient axes of diversity, genre and lyrics language, in Figures 4 and 5.

Genre composition. As shown in Figure 4, the corpus is broad but pop-leaning: Pop accounts for 44.7% of tracks, followed by Rock (10.3%), Hip-Hop/Rap (8.5%), Latin (8.1%), and Electronic/Dance (6.9%). A long tail spanning R&B/Soul, Country, Classical, Folk, World, Punk, Jazz, and Afrobeats covers the remaining ∼21%, ensuring that Mugi is exposed to a wide stylistic range rather than a single dominant idiom.

Language composition. Figure 5 reports the lyrics-language distribution over labeled tracks. English is the most prevalent (29.0%), yet the corpus is strongly multilingual: Spanish (16.2%), Japanese (16.2%), and Mandarin Chinese (14.6%) are nearly as common, and a further 8.2% of tracks mix multiple languages. Korean (5.5%), Cantonese (2.1%), German, French, and Portuguese each contribute additional mass, with the remainder spread over dozens of lower-resource languages. This balance lets Mugi learn language-agnostic musical structure while retaining strong coverage of the world's major vocal traditions.

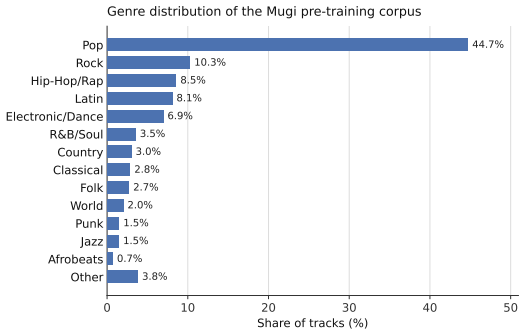


Figure 4: Genre distribution (top-13 genres plus *Other*). Pop dominates, but a broad long tail spans Rock, Hip-Hop/Rap, Latin, Electronic/Dance, and beyond.

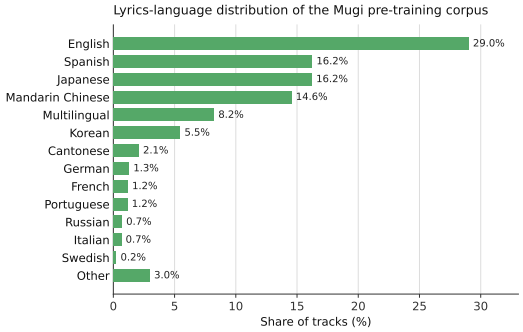


Figure 5: Lyrics-language distribution over labeled tracks. English leads, but Spanish, Japanese, and Mandarin Chinese are nearly as prevalent, and 8.2% of tracks are multilingual.

A.6 Task and Caption

A.6.1 Task sampling strategy

The training of Mugi optimizes one model over three generation behaviours: *plain* song generation from text, *cover* generation additionally conditioned on a reference melody, and generation conditioned on a *timbre embedding* that pins the vocal identity. Instead of separate heads or sequential phases, the behaviour is a latent per-sample configuration drawn at data-loading time, so all modes coexist in every batch and share one parameter set.

Each sample draws two independent Bernoulli variables. With probability $p_{\text{cover}} = 0.35$ it becomes a cover task and its melody prefix is loaded; otherwise it is plain song generation. Independently, with probability $p_{\text{timbre}} = 0.25$ a 192-dimensional CAM++ speaker embedding is injected into the

reference slot, while the remaining samples receive an all-zero vector that the model learns to read as "no timbre specified". The realized mixture is therefore the product of the two draws (Table 8).

Cover conditioning is inserted as a prefix block between the lyrics and the acoustic tokens. Timbre conditioning reuses the reference slot, online audio-prompt extraction being disabled; where a song yields several embeddings, one is sampled from the half of the chunks with highest centrality and re-drawn at every exposure, making the timbre reference a per-epoch augmentation.

Table 8: Joint task mixture in the pretrain stage of Mugi, i.e. the product of two independent per-sample Bernoulli draws ($p_{\text{cover}} = 0.35$, $p_{\text{timbre}} = 0.25$).

	Timbre embedding		
	absent	present	total
Plain generation	50%	15%	65%
Cover generation	25%	10%	35%
Total	75%	25%	100%

A.6.2 Caption

Table 9: The five caption formats used in the pretrain stage of Mugi, all rendered from the same song. Long examples are abridged with …; the full-attribute format concatenates short attribute sentences with the caption and the detailed caption in a randomly permuted order.

Format	Example
Full attributes	The genre is Funk / Soul. The moods are groovy, upbeat, romantic, passionate, energetic. The language is English.… This upbeat funk and soul song revolves around a tight, syncopated rhythm section anchored by slap bass and drum breaks, …
Caption only	A classic 1970s-style soul and funk track featuring a driving bassline, bright brass section, and passionate vocals about escaping the city for love.
Short keywords	Funk / Soul,brass section,backing vocals,electric guitar
Minimal genre	Funk / Soul
Detail caption only	This upbeat funk and soul song revolves around a tight, syncopated rhythm section anchored by slap bass and drum breaks, layered with wah-wah guitar and dynamic brass stabs. The soulful lead male vocal expresses a yearning to leave the busy city life …

Every caption in the pretrain corpus is produced by an audio-capable large language model API of the Qwen3.5-Omni class, which listens to the finished mix and emits a structured description: a short caption, a longer detailed caption, and attribute fields covering genre, mood, instrumentation, vocal characteristics, and usage context. From this single annotation we render five caption formats, exemplified in Table 9.

Text conditioning varies orthogonally to the task draw: each caption is rendered in one of six formats (full attributes 20%, caption only 25%, short 30%, minimal genre 10%, detail caption only 10%, empty 5%), and a further 10% of samples are flagged unconditional to supply the null branch for classifier-free guidance.

A.7 Evaluation Metrics

General quality. We use SongBench (Wu et al., 2026) and Audiobox Aesthetics (Tjandra et al., 2025) to assess overall musicality and audio quality. Prompt consistency is measured by the cosine similarity between MuQ-MuLan (Zhu et al., 2025) embeddings of the generated audio and the input caption. Intelligibility is measured by the phoneme error rate (PER): we transcribe the generated vocals with Qwen-3-ASR (Shi et al., 2026) and compute the phoneme-level edit distance against the input lyrics.

Cover-specific metrics. Melodic consistency with the reference song is evaluated on the vocal tracks of the generated and reference songs. We extract F0 sequences from both, align them with dynamic time warping, and compute raw pitch accuracy (RPA) and raw chroma accuracy (RCA) using `mir_eval` (Raffel et al., 2014). RCA measures pitch accuracy up to octave errors, while RPA additionally penalizes octave deviations. We further report the mean absolute error (MAE) between the transcribed MIDI pitches of the generated and reference vocals.

Timbre-specific metric. We extract speaker embeddings from the generated vocals and the voice prompt using Resemblyzer (Wan et al., 2018) and report their cosine similarity. Generated vocals are obtained by applying the same source separator used in our data pipeline, so that all systems are evaluated under identical conditions.

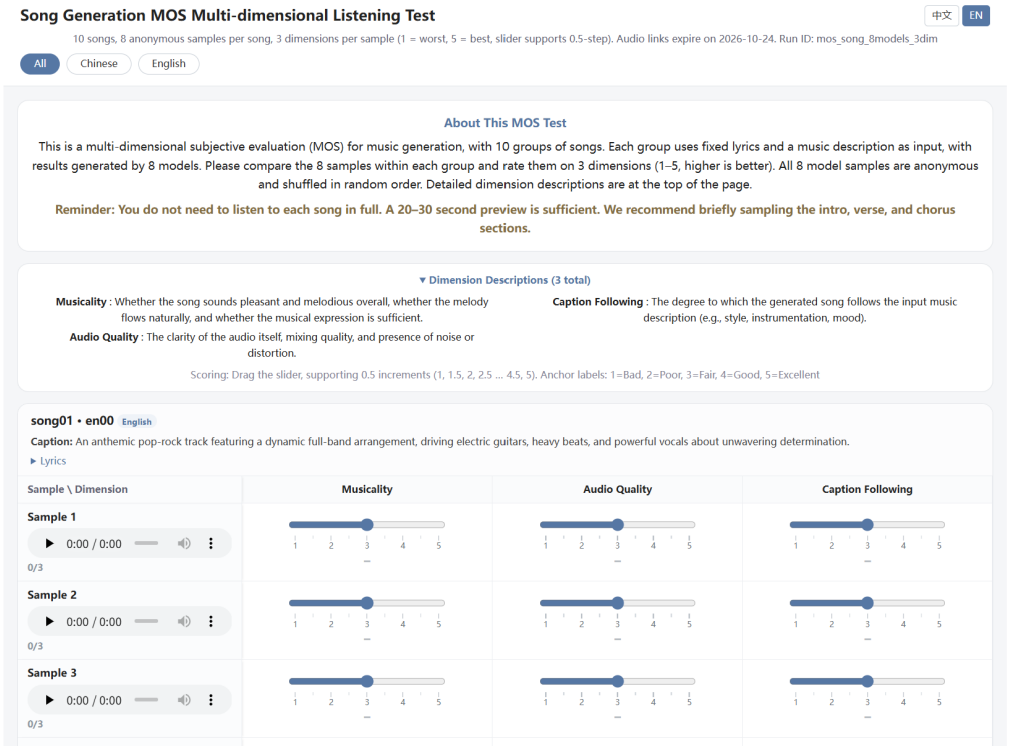


Figure 6: Screenshot of human subjective evaluation webpage.

A.8 Subjective Evaluation Protocol

We conduct the listening study on the generated songs of the compared systems. All samples are anonymized and presented in an order that is randomized independently for each participant. For each sample, participants rate three aspects on a 5-point scale: musicality, audio quality, and consistency with the input caption. Participants are instructed to listen each song of 10 groups. Each participant can submit the questionnaire only once. We collect ratings from 15 participants and report the mean scores per system. A screenshot of the subjective evaluation webpage is shown in Figure 6.