MuLaCover: Flexible Cover Song Generation via Symbolic Tonal Control

MuLa Labs, Vera Praxis Lab

contact@mulalabs.ai

https://github.com/HeartMuLa/MuLaCover

https://verapraxis.ai/mulalabs

Abstract

A cover song re-renders an existing piece by preserving its tonal content—melody and chord progression—while reshaping other musical attributes. This suggests a natural formulation for generative cover synthesis with pretrained music language models: explicitly control tonal content, while specifying lyrics and style through text. Existing approaches condition on frame-level features, tightly coupling generation to the reference's timing and structure, and limiting flexibility such as tempo variation, structural rearrangement, or partial conditioning. To this end, we present MuLaCover, a cover generation model that conditions a pretrained text-to-song foundation model on a symbolic lead sheet. Our design enables alignment-free control: generated outputs preserve the tonal signature of the source without frame-level alignment, allowing flexible timing, structure, and segment-level generation. We further introduce a training curriculum with partially mismatched audio–symbolic pairs, improving diversity and robustness. We evaluate MuLaCover with objective metrics, standard subjective ratings, and in-depth expert interviews that provide fine-grained diagnostic insights. Experiments show that MuLaCover achieves state-of-the-art performance among open-source systems and is competitive with leading commercial models such as Suno-v5.5.

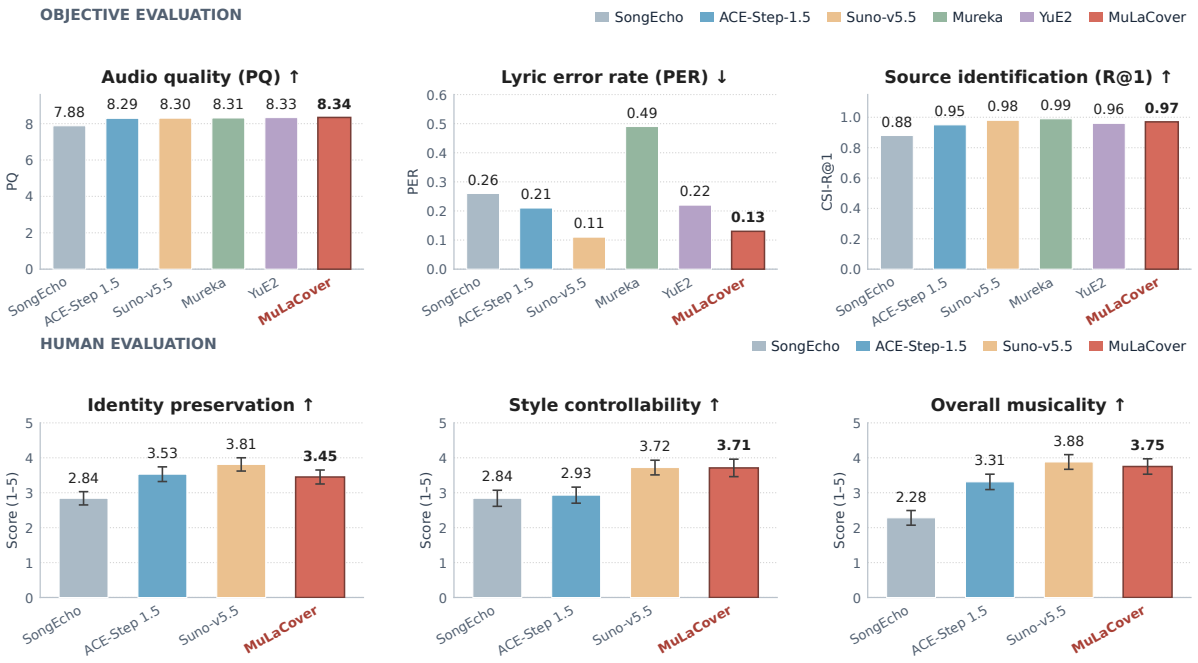

Figure 1 | Full-song cover generation: objective results for six systems (top; Mureka: 248 outputs, others: 256) and listening scores for four systems (bottom; mean ± 95% CI, 22 participants). Mureka and YuE2 were not included in the listening study. Data: Tables 1 and 3.

Contents

1. Introduction

A *cover song* is a re-rendering of an existing piece that preserves its musical identity while freely reshaping every other aspect of the recording. Decades of cover-song-identification research [1–7] converge on a clean characterization of this identity: across renditions, key, tempo, instrumentation, vocal timbre, song structure, and even lyrics may all change; what stays invariant is the *tonal content* — the melody and the underlying chord progression. This invariant/variant taxonomy suggests a natural recipe for cover *generation*: condition a pretrained text-to-song foundation model [8–10] on the tonal content of a source rendition, and let its lyrics and style prompts shape everything else.

The unresolved design question is how to encode the tonal-content signal. Existing audio-domain cover systems answer it with frame-level features extracted from the source recording itself, such as vocal F_0 contours [11] or quantized audio latents [12], sampled at the same rate as the target audio. This choice imposes a strict *frame-wise alignment* between source and cover: the cover cannot run at a different tempo, cannot rearrange structure, and cannot draw on a partial reference such as a single verse or chorus.

We argue that the right tonal-content representation is the *symbolic lead sheet* long used in music information retrieval [13–15]: a melody line paired with a chord progression, which by construction abstracts away surface acoustics and retains only the cover-invariant musical skeleton. Crucially, before feeding the lead sheet into the model, we time-stretch it so that all sequences share a canonical tempo. This BPM normalization is what frees the conditioning signal from any frame-wise tie to the source recording: it leaves the cover model free to set its own pace, and, via attention masking over the symbolic sequence, lets the model accept any sub-segment of the song as a partial reference. We inject this normalized symbolic signal into a pretrained text-to-song backbone through a gated cross-attention module, yielding our system **MuLaCover**.

Two issues remain after the architectural design. The first concerns data. Most text-to-song models [8, 16, 17, 9–11] are trained on recordings whose lyrics, style, and audio necessarily come from the same single rendition; the model has never been asked to render the tonal content of one rendition under the lyrics and style of a *different* one, yet precisely this mismatch is what cover generation demands. We close this supervision gap through *cross-cover finetuning* on a curated paired dataset that pairs the tonal content of one rendition with the lyrics, style, and audio of another. The second concerns evaluation. Cover quality is multi-faceted and idiomatic, but an aggregate Mean Opinion Score (MOS) listening study reveals *whether* a system is good without explaining *why*, and offers little guidance for system iteration. We therefore complement listening-study questionnaires with a structured *expert-interview* protocol that elicits diagnostic, comparative judgments from trained listeners. Together with objective metrics, these subjective signals yield a comprehensive evaluation.

We evaluate MuLaCover on two cover tasks. On the standard *full-song cover* task, MuLaCover achieves strong musical quality, lyric controllability, and cover identifiability, with quality and lyric metrics competitive with Suno-v5.5; the listening study and expert interviews corroborate these findings. As a natural consequence of its alignment-free conditioning, MuLaCover further tackles *partial-reference cover generation*, in which only a fragment of the reference is provided and the model produces a full-song cover that preserves the fragment's tonal content while freely re-creating the rest, a task on which MuLaCover achieves superior performance over existing open-sourced baselines across three fragment configurations in objective evaluation.

Our main contributions are:

- We formulate cover generation as conditioning a pretrained text-to-song foundation model on the BPM-normalized symbolic lead sheet of a source rendition, and realize it as MuLaCover, which injects the signal through a gated cross-attention module. The design removes the strict frame-wise alignment that has bound prior cover systems to the source's exact timing and structure, and supports partial-reference cover generation as a natural by-product.
- We identify cross-version pairing of content and style as the missing supervision signal for cover generation, and provide a curated paired dataset together with a cross-cover finetuning curriculum that turns a generic text-to-song model into a cover generator.
- We evaluate MuLaCover with objective metrics, listening-study questionnaires, and a structured expert-interview protocol that complements aggregate scores with fine-grained, comparative diagnostics. The combined evaluation establishes MuLaCover as state-of-the-art among open-source cover systems and broadly on par with the closed-source commercial system Suno-v5.5.

2. Related Work

2.1. Cover Song Identification

Cover song identification (CSI) is the music information retrieval task of retrieving cover versions of a query song from a database [1]. Across versions, key, tempo, structure, instrumentation, timbre, and lyrics may all change [1–7]; what remains invariant is the *tonal content* [3], namely the melody [18] and the underlying chord progression or chroma features [19, 2], which serves as the primary cross-version identifier. This invariant-vs-variant decomposition has shaped algorithmic design throughout the CSI literature. Classical rule-based systems [2, 3] extract tonality-oriented descriptors such as chroma features [20] or HPCP sequences [21], and pair them with similarity computations that explicitly account for the varying attributes: transposition handles key shifts, and dynamic time warping handles tempo differences [3]. Modern deep CSI systems [4, 22, 5–7] preserve the same inductive bias but realise it through learned representations, training networks on contrastive or classification objectives with pooling, matching, and augmentation strategies (e.g. frequency-axis rolling for pitch shifts, speed perturbation, and spectrogram masking) deliberately designed so that embeddings remain invariant to key, tempo, and structural variations between versions. Viewed through a generative lens, the variant/invariant taxonomy of CSI is exactly a *content/style* decomposition of a song: tonal content is the content, and everything else (instrumentation, timbre, tempo, structure, lyrics) is the style that a cover artist freely reshapes. Cover song generation is then the *style transfer* counterpart of CSI, and as such belongs to the broader space of controllable music generation reviewed next.

2.2. Controllable Music Generation

In the audio domain, controllable music generation has advanced rapidly through language-model and diffusion systems [23, 24, 16, 9, 25, 8, 17, 10]. Control here is predominantly textual—style descriptions and lyric prompts—with finer-grained content signals such as melody and chord progression only recently emerging as auxiliary conditions [26–28, 11]. Symbolic music generation, by contrast, has long offered interpretable, disentangled control at global (time signature, tempo, instrumentation [29–31]), local (chord progression, compositional texture [32–34]), and hierarchical [35] scales. For cover song generation, symbolic piano cover systems

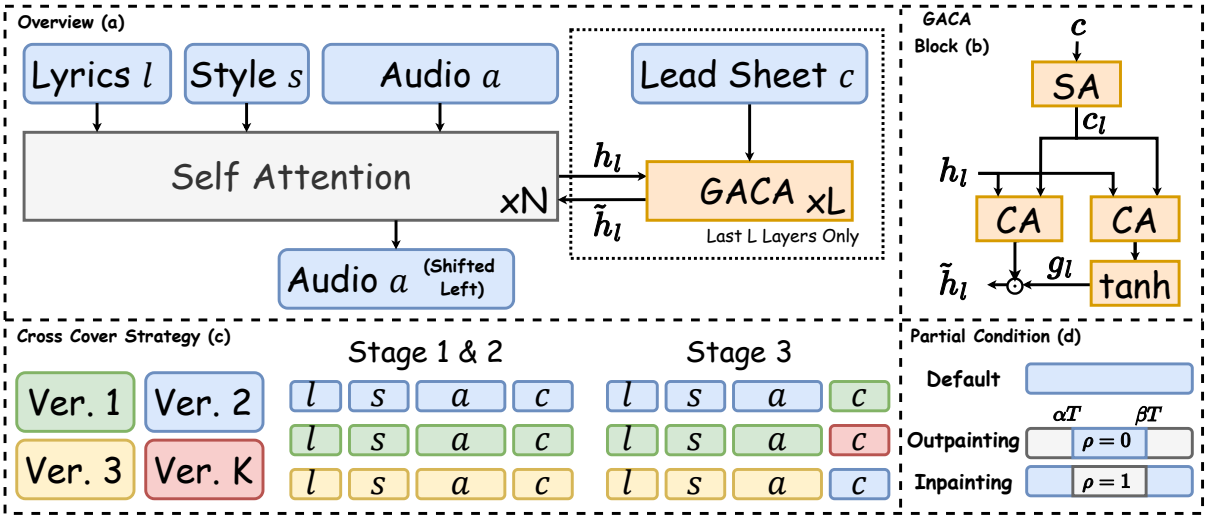

Figure 2 | Overview of MuLaCover. **(a)** The pretrained text-to-song backbone has N transformer layers consuming lyrics ℓ, style s, and audio tokens a; at its last L layers, the Gated Adaptive Cross-Attention (GACA) module injects the symbolic tonal-content condition c into the hidden states h_l. **(b)** Inside a GACA block, c is refined by a layer-specific self-attention (SA) into c_l, then merged into h_l through a cross-attention (CA) update modulated by a learned, tanh-bounded gate g_l. **(c)** Cross-cover strategy in Stage 3: colors denote different renditions of the same song. Stages 1–2 draw all four signals ℓ, s, a, c from the same rendition; Stage 3 draws c from a different rendition than ℓ, s, a. **(d)** Partial conditioning via attention masking on c: ρ=0 keeps the interval $[\alpha T, \beta T]$ (*outpainting*), ρ=1 keeps its complement (*inpainting*); the default (α, β, ρ)=$(0, 1, 0)$ recovers the full-song setting.

[36–38] remain narrow in scope. Audio-domain systems such as SongEcho [11] and ACE-Step 1.5 [12] condition on a temporal feature of the source sampled at the same rate as the target audio features, enforcing strict frame-wise alignment and leaving no degree of freedom in tempo or structure. MuLaCover removes this rigidity through flexible symbolic lead-sheet conditioning.

3. Method

MuLaCover equips a decoder-only text-to-song backbone with tonal-content control. Mainstream text-to-song foundation models [8, 10] already expose two controls — a lyrics prompt ℓ along with a style prompt s that conveys non-lyric attributes such as genre and instrumentation — and autoregressively model the audio token sequence a as their generation target. MuLaCover adds the tonal-content signal c on top of these existing controls; an overview of the resulting architecture is shown in Figure 2(a).

Abstracting away the specific text-to-song instantiation, the remaining design problem becomes one of sequence-to-sequence conditional injection. The backbone produces, at each of its N transformer layers, a sequence of hidden states $h_l \in \mathbb{R}^{S\times D}$ ($l = 1, \ldots, N$, with S the sequence length and D the feature dimension); the new control is itself a sequence $c \in \mathbb{R}^{T\times D}$ of length T. MuLaCover merges information from c into h_l at the last L layers via a gated adaptive cross-attention module, leaving the remaining $N-L$ layers and the original training loss intact.

Three components, each ablated independently in §4.5, complete the system: (i) a symbolic lead-sheet representation that defines the tonal-content sequence c (Sec. 3.1); (ii) the gated

adaptive cross-attention injection module (Sec. 3.2); and (iii) a three-stage curriculum that aligns the newly introduced parameters with the backbone and exposes the model to cross-version training pairs in which the tonal-content signal and the existing controls are drawn from different renditions of the same song (Sec. 3.3).

3.1. Symbolic Lead Sheet Representation

A lead sheet, comprising a melody line and a chord progression, captures the tonal content essential to the cover task and has been used in music generation [13, 37]; we adopt it as our conditioning signal c.

Given a reference recording, we extract the vocal melody and chord progression with pretrained transcription models [39, 40]. To prevent the lead sheet from inheriting the source's specific tempo, and thereby to avoid imposing any frame-wise alignment between c and the generated audio, we estimate the source's beats-per-minute (BPM) and time-stretch the lead sheet so that all sequences are normalized to a canonical BPM=120. We further quantize the symbolic timeline at the resolution of a sixteenth note and denote the resulting sequence length as T. This BPM normalization is essential to the system: removing it forces a strict frame-wise correspondence between c and the audio that exposes a sharp content–style tradeoff, as we quantify in Section 4.5 (w/o BPM-norm).

After BPM normalization, the melody is encoded as a binary pianoroll $M \in \{0,1\}^{T\times 256}$, whose $256 = 128 \times 2$ feature dimensions capture the onset and sustain status of each of the 128 MIDI pitches at every step; the chord progression is encoded as a binary chromaroll $H \in \{0,1\}^{T\times 12}$ over the twelve pitch classes. The two representations are projected into a shared embedding space and concatenated along the feature dimension to form the conditioning sequence consumed downstream:

$$c \;=\; \left[\, M\,W_M \;\|\; H\,W_H \,\right] \;\in\; \mathbb{R}^{T\times D}, \tag{1}$$

where $W_M \in \mathbb{R}^{256\times D/2}$ and $W_H \in \mathbb{R}^{12\times D/2}$ are learned linear projections, $\|$ denotes feature-wise concatenation, and D matches the feature dimension of the backbone. Conditioning on the melody alone, without the chord progression, is ablated in Section 4.5 (w/o Chord); we also compare the symbolic lead sheet against the frame-wise F_0 contour used by prior cover systems [11] in the same section (w/ F_0).

3.2. Gated Adaptive Cross-Attention Injection

MuLaCover injects the symbolic condition c only into the last L layers of the backbone — layers $N{-}L{+}1$ through N. The intuition is that adding tonal-content control should only need to reshape the predicted audio-token distribution at the final stages of generation; the lower $N{-}L$ layers, which carry the pretrained backbone's general acoustic representations, can therefore be left structurally unchanged. The internal structure of the resulting GACA block is illustrated in Figure 2(b). We compare this cross-attention-based design against the alternative prefix-tuning mechanism used in prior LM-based controllable music generation work [26, 28] in Section 4.5 (w/ Prefix tuning).

At each modified layer l, the symbolic condition is first refined by a layer-specific *self-attention* block, denoted $\mathrm{SA}_l(\cdot)$, which allows the conditioning to specialize to the level of abstraction expected by the corresponding decoder layer:

$$c_l \;=\; \mathrm{SA}_l(c) \;\in\; \mathbb{R}^{T\times D}. \tag{2}$$

Here and throughout, $\mathrm{SA}(\cdot)$ stands for a standard multi-head self-attention block [41]. The layer subscript indicates that each modified layer carries its own independent parameters.

The refined symbolic states c_l are then merged into h_l through a *cross-attention* block, which we denote $\mathrm{CA}_l(q, m)$: q is the query input and m is the source input, from which the keys and the values are derived; the output has the same length as q. Before c_l is fed in as the source input, we apply rotary positional embeddings (RoPE) [42] to it, so that the temporal positions of the symbolic sequence are carried through into the keys and values together with their content. The query input is the audio-side hidden states h_l.

A naive residual update of the form $\tilde{h}_l = h_l + \mathrm{CA}_l(h_l, c_l)$ admits the conditioning update at every position and feature dimension indiscriminately, leaving no room for the model to decide where the symbolic information should actually be incorporated; we observe in Section 4.5 (w/o Gate) that this substantially degrades cover identifiability. Inspired by [26], we therefore introduce a *learned gate* $g_l \in (-1, 1)^{S \times D}$ that lets the model decide adaptively, for each position and feature dimension, how much symbolic information to admit at layer l:

$$\tilde{h}_l = h_l + g_l \odot \mathrm{CA}_l\big(h_l, c_l\big), \tag{3}$$

$$g_l = \tanh\big(\mathrm{CA}_l^{\mathrm{gate}}\big(h_l, c_l\big)\big), \tag{4}$$

where $\odot$ denotes elementwise multiplication, tanh bounds the gate to $(-1, 1)$, and $\mathrm{CA}_l^{\mathrm{gate}}$ is a parameter-light cross-attention block that shares the same (q, m) convention as CA_l and likewise receives the RoPE-augmented c_l as its source input. We zero-initialize the output projection of $\mathrm{CA}_l^{\mathrm{gate}}$, so that $g_l = \tanh(0) = 0$ at the start of training and the augmented model exactly reproduces the pretrained backbone's behavior at initialization.

Partial conditioning via attention masking. The condition c is built from the full-song lead sheet, yet MuLaCover also supports covers of arbitrary song segments through attention masking, naturally instantiating the two canonical partial-generation modes. The user specifies a mask interval via two scalars $\alpha, \beta \in [0, 1]$ with $\alpha \le \beta$, together with a binary inversion flag $\rho \in \{0, 1\}$: ρ=0 keeps the interval as the only conditioned region (*outpainting*, in which the model is asked to freely generate the rest of the song around a given segment), while ρ=1 keeps its complement (*inpainting*, in which the model fills in the masked interval given the surrounding conditioning context). These parameters induce a set of *active positions*

$$\mathcal{A}(\alpha, \beta, \rho) = \begin{cases} \{\, t \in \{1, \dots, T\} : \alpha T < t \le \beta T \,\}, & \rho = 0 \quad \text{(outpainting)}, \\ \{1, \dots, T\} \setminus \{\, t \in \{1, \dots, T\} : \alpha T < t \le \beta T \,\}, & \rho = 1 \quad \text{(inpainting)}. \end{cases} \tag{5}$$

Within SA_l, self-attention is restricted to position pairs both lying in $\mathcal{A}$; within CA_l and $\mathrm{CA}_l^{\mathrm{gate}}$, queries from h_l may only attend to keys and values whose source position lies in $\mathcal{A}$. Positions outside $\mathcal{A}$ are thus ignored entirely by the injection pathway. The defaults $(\alpha, \beta, \rho) = (0, 1, 0)$ recover the full-song setting; the three masking modes are illustrated in Figure 2(d).

3.3. Curriculum Training Strategy

We train MuLaCover in three stages. Stages 1 and 2 establish the mapping from the lead sheet to the audio at increasing scale; Stage 3 exposes the model to cross-version pairings of content and style.

Stage 1: Short-clip warmup. We first train on 30-second segments. Short clips are cheap to optimize and let the newly introduced modules pick up the basic mapping from the lead sheet

to the audio quickly, before the model is asked to handle longer-range structure. Skipping this stage degrades both perceptual quality and cover identifiability, as we show in Section 4.5 (w/o Warmup).

Stage 2: Full-song training. We then continue training on full-length recordings, extending the mapping learned in Stage 1 to song-scale temporal context. The existing controls ℓ, s, the tonal-content signal c, and the audio target are all drawn from the same song in Stage 1 and 2.

Stage 3: Cross-cover finetuning. A cover song takes the tonal content of one rendition and re-renders it under the lyrics and style of another. Stages 1 and 2 do not expose the model to this cross-version pairing, since c and ℓ, s are always drawn from the same recording. To close this gap, we collect, for each song, a group of K available renditions $\{v_1, \ldots, v_K\}$, and at each training step sample two indices $i, j \in \{1, \ldots, K\}$: the tonal-content signal c is extracted from rendition v_i, while the remaining controls ℓ, s and the audio target are taken from rendition v_j. The model is thus explicitly trained to render the tonal content of v_i under the lyrics and style of v_j, which is precisely the cover-generation behavior we wish to evoke at inference. Figure 2(c) contrasts this Stage 3 cross-version pairing with the same-rendition setup of Stages 1 and 2. The contribution of this cross-cover pairing to style controllability is quantified in Section 4.5 (w/ SP).

4. Experiments

This section describes the training setup of MuLaCover and conducts evaluation on the full-song cover task and partial-reference cover task. Through objective and subjective evaluation, we show that MuLaCover surpasses representative open-source systems and is on par with the strongest closed-source model. We conduct ablation studies to show the necessity of each design choice.

4.1. Training Configurations

We select approximately 200k songs from our internal dataset. Following the data-processing protocol of recent work [11, 43], each track is passed through an automated annotation pipeline: vocals are isolated with Demucs [44] and transcribed with Whisper [45]; section-level structure is obtained by ensembling All-in-One [46] and SongFormer [47]; and free-form style tags produced by Qwen2-Audio [48] are formatted with DeepSeek [49] into the tag vocabulary used by HeartMuLa [10]. For Stage 3, we additionally curate a paired cross-cover set from our internal dataset. For each of 2,500 source songs, we select 7 stylistically distinct cover versions, yielding 2,500 clusters of 8 renditions each (the source plus 7 covers), for a total of 20,000 recordings.

We build MuLaCover on the 3B-parameter variant of HeartMuLa [10], an LM-based music generation model with $N = 28$ backbone transformer layers. Our Gated Adaptive Cross-Attention modules are inserted into the last $L = 8$ backbone layers. Stages 1 and 2 keep the backbone frozen and update only the newly introduced parameters; Stage 3 additionally applies LoRA [50] to the last $L = 8$ backbone layers and finetunes on the cross-cover pairs. All experiments run on 16×NVIDIA A100 (80 GB) GPUs; per-stage hyperparameters are listed in Appendix A.

4.2. Tasks and Baselines

We evaluate MuLaCover on two cover-generation tasks. Task 1 is full-song cover generation, where the model is asked to generate a cover conditioned on the reference music, target style,

and corresponding lyrics. Task 2 is partial-reference cover generation. Under this setting, only a fragment of the reference music is observed, along with the target style and lyrics for the whole song. The model is asked to produce a full-song cover, preserving the tonal content of the observed part while recreating the remainder. Instantiating the active-position mask $\mathcal{A}(\alpha, \beta, \rho)$ from Eq. 5, Task 2 admits three setups—continuation, outpainting, and inpainting—detailed in Appendix B.

We compare against three contemporary systems: SongEcho [11], a fine-tuning-based cover system; ACE-Step-1.5 [12], a large-scale pretrained music foundation model; and Suno-v5.5 [51], a leading closed-source commercial system. We additionally report Mureka [52] and YuE2 [8][1] in the Task 1 objective comparison; neither is included in the partial-reference or listening evaluations. See Appendix B for baseline configurations. Both Task 1 and Task 2 are evaluated objectively in Section 4.3; for Task 2, objective metrics are averaged across the three setups, with per-setup metrics reported in Appendix C. Task 1 additionally receives a subjective evaluation in Section 4.4. Suno-v5.5 does not expose an API entry point for the partial-reference setting and is therefore omitted from comparison in Task 2.

4.3. Objective Evaluation

We prompt DeepSeek [49] to generate 32 sets of lyrics and feed them to Suno-v5.5 [51] to obtain 32 reference songs, and prompt DeepSeek again to generate 8 distinct style tags. Each model under evaluation is conditioned on a reference song, its lyrics, and one style tag at a time, producing 8 stylistic variants per reference song for a target total of 256 covers per model. For the additional full-song baselines, YuE2 has all 256 outputs; Mureka has 248 outputs because reference song 19 is absent under all eight styles. Their results are averaged over available outputs within each style and then across the eight styles.

We assess perceptual quality with two complementary pretrained predictors. From AudioBox-Aesthetics [53] we report Production Quality (PQ), Production Complexity (PC), Content Enjoyment (CE), and Content Usefulness (CU). From SongEval [54] we report Coherence (Coh.), Memorability (Mem.), Naturalness (Nat.), Clarity (Cla.), and Musicality (Mus.). For lyric controllability, we isolate the vocal track of each cover with Demucs [44], transcribe it with Whisper [45], and report Phoneme Error Rate (PER) against the input lyrics. For style, we use the MuQ-MuLan music–text contrastive model [55]: MuQ-Sim is the cosine similarity between the cover and its target style tag in the joint embedding space, measuring fidelity to the requested style; MuQ-Dist is the cosine distance between the cover and the source recording, measuring stylistic deviation from the source. For cover identifiability, we utilize cover song identification model CoverHunter [7]: each cover serves as a query against the database of 32 source songs, and we report top-1 retrieval accuracy (CSI-R@1) and the mean retrieval rank of the true source (CSI-Rank).

[1] https://github.com/multimodal-art-projection/YuE

Table 1 | Objective evaluation on the full-song cover task (Task 1). ↑ / ↓ indicate higher / lower is better. **Bold** marks the best and underline the second-best result in each column. Rankings use unrounded scores. All CSI results use the fine-tuned CoverHunter evaluator.

AudioBox quality (↑) [53]

Method	PQ	PC	CE	CU
SongEcho [11]	7.88	6.11	7.26	7.54
ACE-Step-1.5 [12]	8.29	6.30	7.68	7.86
Suno-v5.5 [51]	8.30	**6.44**	7.66	7.92
Mureka [52]	8.31	6.39	7.61	7.82
YuE2 [8]	8.33	6.30	**7.71**	7.91
MuLaCover	**8.34**	6.34	7.70	**7.93**

Table 1 (continued).

SongEval aesthetics (↑) [54]

Method	Coh.	Mem.	Nat.	Cla.	Mus.
SongEcho [11]	3.70	3.57	3.37	3.47	3.46
ACE-Step-1.5 [12]	4.48	4.46	4.26	4.35	4.36
Suno-v5.5 [51]	**4.61**	**4.60**	**4.42**	**4.52**	**4.51**
Mureka [52]	4.28	4.24	4.05	4.12	4.11
YuE2 [8]	4.48	4.46	4.24	4.35	4.36
MuLaCover	4.59	4.59	4.34	4.44	4.47

Table 1 (continued).

Lyrics, style, and cover identification

Method	PER (↓)	MuQ-Sim (↑)	MuQ-Dist (↑)	CSI-R@1 (↑)	CSI-Rank (↓)
SongEcho [11]	0.26	0.38	0.39	0.88	1.55
ACE-Step-1.5 [12]	0.21	0.30	0.40	0.95	1.10
Suno-v5.5 [51]	**0.11**	**0.50**	0.33	0.98	1.11
Mureka [52]	0.49	0.48	0.40	**0.99**	**1.02**
YuE2 [8]	0.22	0.45	0.32	0.96	1.06
MuLaCover	0.13	0.41	**0.48**	0.97	1.05

Table 2 | Objective evaluation on the partial-reference cover task (Task 2), averaged over the three masking configurations (continuation, inpainting, outpainting). ↑ / ↓ indicate higher / lower is better. **Bold** marks the best.

AudioBox quality (↑) [53]

Method	PQ	PC	CE	CU
SongEcho [11]	7.58	5.59	6.80	7.44
ACE-Step-1.5 [12]	8.26	5.91	7.66	7.84
MuLaCover	**8.33**	**6.31**	**7.69**	**7.92**

Table 2 (continued).

SongEval aesthetics (↑) [54]

Method	Coh.	Mem.	Nat.	Cla.	Mus.
SongEcho [11]	3.14	3.04	2.85	2.87	2.92
ACE-Step-1.5 [12]	4.22	4.20	3.95	4.08	4.08
MuLaCover	**4.57**	**4.57**	**4.32**	**4.42**	**4.45**

Table 2 (continued).

Lyrics, style, and cover identification

Method	PER (↓)	MuQ-Sim (↑)	MuQ-Dist (↑)	CSI-R@1 (↑)	CSI-Rank (↓)
SongEcho [11]	0.55	0.25	**0.63**	0.41	5.94
ACE-Step-1.5 [12]	0.36	0.34	0.50	0.52	4.92
MuLaCover	**0.12**	**0.42**	0.50	**0.70**	**2.60**

Table 1 reports objective results on Task 1. MuLaCover achieves the highest Production Quality and Content Usefulness and ranks second to Suno-v5.5 on all five SongEval dimensions and PER. YuE2 obtains the highest Content Enjoyment, while Mureka and YuE2 both exceed MuLaCover on target-style similarity. Mureka obtains the highest CSI-R@1 and lowest CSI-Rank on its 248 available outputs; MuLaCover ranks second on CSI-Rank. Table 2 reports objective results on Task 2 averaged over the three masking configurations (continuation, inpainting, outpainting); per-configuration breakdowns are deferred to Appendix C. When only a fragment of the reference music is exposed, the baselines suffer a sharp drop in cover identifiability, accompanied by varying degrees of degradation in lyric controllability and audio quality. MuLaCover, in contrast, retains the best performance across all metrics, further demonstrating the flexibility of our control scheme.

4.4. Subjective Evaluation

Listening study. We complement the objective evaluation with a blind listening study. From the test set introduced in Section 4.3 we randomly select 8 source songs, each paired with a target style tag and the corresponding lyrics. For each song we generate a cover with all four systems (MuLaCover, ACE-Step-1.5, SongEcho, and Suno-v5.5), shuffle their order, and ask each rater to score every cover on four 5-point Likert scales: *Identity Preserveness*, *Style Controllability*, *Creativeness*, and *Overall Musicality*. Each participant was assigned a random subset of 5 of the 8 songs. We received 34 saved submissions. After excluding submissions with no complete case and dropping incomplete case blocks, the analysis retained 22 participants self-reporting their musical background as amateur (11), intermediate (6), or professional (5), yielding 95 complete participant–case blocks and 1,520 ratings in total. Partial completion resulted in 4.32 complete cases per retained participant on average.

Table 3 reports the per-metric mean and 95% confidence interval across all (participant, song) pairs. Under paired block-level Wilcoxon tests, MuLaCover significantly outperforming open-sourced baselines on Style Controllability, Creativeness, and Overall Musicality and is on par with Suno-v5.5. On Identity Preserveness, MuLaCover significantly outperforms SongEcho and shows no statistically significant difference from ACE-Step-1.5, while Suno-v5.5 remains significantly higher. The full questionnaire instrument, scoring rubrics, and detailed instructions are detailed in Appendix D.

Table 3 | Listening study: mean ± 95% confidence interval on each metric (5-point Likert; higher is better). **Bold** marks the best and underline the second-best result in each column.

Method	Identity Preserveness (↑)	Style Controllability (↑)	Creativeness (↑)	Overall Musicality (↑)
SongEcho [11]	2.84 ± 0.19	2.84 ± 0.23	2.56 ± 0.21	2.28 ± 0.21
ACE-Step-1.5 [12]	3.53 ± 0.21	2.93 ± 0.23	3.03 ± 0.21	3.31 ± 0.22
Suno-v5.5 [51]	**3.81 ± 0.19**	**3.72 ± 0.21**	3.51 ± 0.20	**3.88 ± 0.21**
MuLaCover	3.45 ± 0.20	3.71 ± 0.25	**3.65 ± 0.22**	3.75 ± 0.22

Expert interview. The Likert questionnaire reveals *how good* each system is but not *why*; to probe the latter, we run a structured expert-interview protocol and report an R&B case study. We first sample one genre uniformly at random from a catalogue of common popular-music styles and, without playing any audio, ask the expert to enumerate from prior experience the salient attributes a faithful instance of that genre should exhibit, each with a short justification. We then sample one source song from the test set, present the four blinded covers, and ask the expert to rate and comment on each version against the elicited attributes. Finally, the commentary is fed to a language model to produce a concise summary verdict. The full protocol, the sampled genre, the elicited attributes, the organized expert record, and the summarization prompt are reproduced in Appendix E, where Versions A, B, C, D in the blinded record correspond to ACE-Step-1.5, SongEcho, MuLaCover, and Suno-v5.5, respectively. The summary verdict reads as follows:

> *Under the expert's two pre-specified R&B criteria—rhythm/groove and vocal melodic development—the four blinded versions rank D > C > B > A, with A deviating most toward electronic pop, B and C progressively closer to R&B's cyclic vocal phrasing, and D judged most canonical thanks to its natural groove and pop-R&B vocal timbre, melismas, and phrase endings.*

4.5. Ablation Studies

Table 4 | Ablation studies on the full-song cover task. The first row is the full MuLaCover; rows below ablate one component at a time, grouped by *training schedule*, *condition signal*, and *architecture*. ↑ / ↓ indicate higher / lower is better.

AudioBox quality (↑) [53]

Variant	PQ	PC	CE	CU
MuLaCover-small	8.28	6.36	7.67	7.89
Training schedule				
w/o Warmup	8.07	6.13	7.44	7.72
w/ SP	8.24	6.43	7.63	7.87
Condition signal				
w/o BPM-norm	8.21	6.36	7.61	7.84
w/o BPM-norm (SP.)	8.19	6.40	7.59	7.82
w/ F_0	8.10	6.46	7.55	7.74
w/ F_0 (SP.)	8.12	6.45	7.56	7.76
w/o Chord	7.97	6.30	7.39	7.65
Architecture				
w/o Gate	8.26	6.38	7.62	7.88
w/ Prefix tuning	7.97	5.81	7.37	7.67

Table 4 (continued).

SongEval aesthetics (↑) [54]

Variant	Coh.	Mem.	Nat.	Cla.	Mus.
MuLaCover-small	4.58	4.56	4.33	4.42	4.45
Training schedule					
w/o Warmup	4.41	4.40	4.13	4.24	4.23
w/ SP	4.57	4.55	4.33	4.41	4.44
Condition signal					
w/o BPM-norm	4.56	4.55	4.33	4.40	4.43
w/o BPM-norm (SP.)	4.48	4.45	4.23	4.30	4.35
w/ F_0	4.45	4.41	4.19	4.24	4.41
w/ F_0 (SP.)	4.44	4.38	4.16	4.23	4.39
w/o Chord	4.35	4.32	4.07	4.16	4.16
Architecture					
w/o Gate	4.59	4.58	4.34	4.44	4.46
w/ Prefix tuning	4.34	4.35	4.08	4.16	4.18

Table 4 (continued).

Lyrics, style, and cover identification

Variant	PER (↓)	MuQ-Sim (↑)	MuQ-Dist (↑)	CSI-R@1 (↑)	CSI-Rank (↓)
MuLaCover-small	0.14	0.38	0.49	0.98	1.04
Training schedule					
w/o Warmup	0.16	0.40	0.54	0.11	9.75
w/ SP	0.16	0.35	0.45	0.98	1.04
Condition signal					
w/o BPM-norm	0.14	0.36	0.51	0.73	2.70
w/o BPM-norm (SP.)	0.18	0.34	0.47	0.99	1.04
w/ F_0	0.20	0.39	0.52	0.33	5.20
w/ F_0 (SP.)	0.20	0.38	0.51	0.40	5.55
w/o Chord	0.20	0.38	0.50	0.46	4.86
Architecture					
w/o Gate	0.14	0.38	0.52	0.86	1.74
w/ Prefix tuning	0.43	0.36	0.58	0.09	11.58

To isolate the contribution of each design choice in MuLaCover, we ablate three axes of the system in turn: (i) the *training schedule*, (ii) the *condition signal*, and (iii) the *architecture* used to inject that signal into the backbone. All ablations follow the protocol of Section 4.3 and are reported in Table 4. Due to limited compute, we retrain on a 70k-song subset (denoted MuLaCover-small) and use the same data scale for every variant.

Training schedule. We first ablate the short-clip warmup (Stage 1). The variant w/o Warmup skips Stage 1 and starts training directly from Stage 2, with one extra epoch in Stage 2 to match the total number of training steps. Without the warmup, the newly introduced modules fail to align with the backbone and c is not effectively injected. The output is no longer identifiable as a cover, and the broken injection collaterally drags down perceptual quality. We next ablate the cross-cover pairing in Stage 3, where c is drawn from a different rendition than the one supplying ℓ, s, a. The variant w/ SP (self-pair) reverts Stage 3 to the same-rendition setup of Stages 1 and 2, with extra epochs compensating for the reduced number of (c, ℓ, s, a) tuples. Style controllability deteriorates while everything else is preserved, confirming that the cross-cover stage is what unlocks the model's stylistic flexibility.

Condition signal. Our symbolic tonal-content signal is BPM-normalized to a canonical BPM=120, with the intent of relaxing the otherwise strict frame-wise correspondence between c and the audio so as to enhance style controllability without compromising content control. To verify this, we replace c with a non-BPM-normalized symbolic sequence whose length and frame rate match the audio token stream, and train two variants: w/o BPM-norm (the full three-stage schedule including the cross-cover Stage 3) and w/o BPM-norm (SP.) (Stage 3 restricted to self-pairs). The two together expose a content/style tradeoff that is structural rather than incidental. Once c is strictly frame-wise aligned with the audio, the self-pair regime is fully consistent with the alignment and yields strong cover identifiability at the cost of style flexibility, whereas the cross-cover regime pairs the condition of one rendition with the audio of another, and a frame-wise signal becomes too brittle to small tempo or duration mismatches, so cover identifiability collapses instead. BPM normalization loosens the temporal alignment just enough to keep the content needed for cover identifiability while leaving the slack required by style control.

We note that SongEcho [11] uses a frame-wise F_0 sequence as its content control. Substituting our symbolic signal with the same F_0 representation gives the analogous pair of variants w/ F_0 and w/ F_0 (SP.), and the same content/style tradeoff between the two reappears. Beyond the tradeoff, F_0 based injection is uniformly weaker than the symbolic counterpart, which we attribute to two intrinsic properties of F_0: being continuous rather than discretized to per-pitch values, it is harder to internalize after embedding and thus less amenable to injection; and carrying only the vocal melody, it is a sparser and less informative signal in the first place.

The variant w/o Chord drops the chord progression and keeps only the melody, and ineffective injection emerges. In popular music the chord progression carries the harmonic *skeleton* of the song and is far more informative than the vocal melody alone, and we believe removing it leaves c too sparse for the model to reliably exploit.

Taken together, these results justify our choice of a BPM-normalized symbolic lead sheet (melody plus chord progression) as the tonal-content signal: among the design alternatives we considered, it best balances content control (cover identifiability) and style control.

Architecture. We first remove the learned gate g_l in Equation 3, reducing the injection to a plain residual cross-attention $\tilde{h}_l = h_l + \mathrm{CA}_l(h_l, c_l)$; we denote this variant w/o Gate. Cover identifiability suffers, consistent with our hypothesis that the adaptive gate is precisely the mechanism through which the model selects which positions and feature dimensions of the conditioning update should be admitted into h_l. Without it, the symbolic information is broadcast indiscriminately and the model loses the freedom to bind the condition where it is actually useful. We next replace cross-attention-based injection with a prefix-tuning mechanism, previously shown to be effective for adapting LM-based music generators to new controls [26, 28]; we denote this variant w/ Prefix tuning. Here condition injection essentially fails, and prefix tuning even disrupts the lyrics-to-music mapping already established by the foundation model, as reflected in the PER blow-up. We attribute this to the fact that our backbone already consumes lyrics and style as prefix prompts, and forcing an additional newly introduced control interferes with the autoregressive mapping the model previously learned.

Overall, GACA is a sound and novel choice for condition injection on top of an LM-based music foundation model, compared with the alternatives we examined.

5. Conclusion

We present MuLaCover, a cover song generation system that conditions a pretrained text-to-song foundation model on a BPM-normalized symbolic lead sheet via a gated cross-attention module. MuLaCover is alignment-free, supporting flexible tempo and partial-reference generation in a single unified model. Objective metrics, listening-study questionnaires, and a structured expert-interview protocol jointly show that MuLaCover outperforms open-source cover baselines and is broadly on par with the leading closed-source commercial system Suno-v5.5. We discuss the limitations in Appendix F.

Appendix

A. Per-Stage Training Hyperparameters

Table 5 reports the per-stage training hyperparameters.

Table 5 | Per-stage training hyperparameters. GACA: gated adaptive cross-attention. Entries marked $\star$ follow the dynamic mask schedule described below.

Hyperparameter	Stage 1	Stage 2 (w/o mask)	Stage 2 (w/ mask)	Stage 3
Learning rate	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-5}
Epochs	3	2	2	2
Avg. bs/GPU	9	1	1	1
(α, β, ρ)	$(0.0, 1.0, 0.0)$	$(0.0, 1.0, 0.0)$	$\star$	$\star$
Trainable modules	GACA	GACA	GACA	GACA & LoRA
Trainable params	~300M	~300M	~300M	~400M
Time per epoch	0.5 h	~5 h	~5 h	~2 h
Optimizer	AdamW			
LR scheduler	warmup followed by cosine annealing			
Gradient accum.	4			
GPUs	16			

Mask configuration (α, β, ρ)**.** α, β, and ρ denote the start ratio, end ratio, and invert ratio of the masked window. For Stage 1 and Stage 2 (w/o mask), the configuration $(0.0, 1.0, 0.0)$ corresponds to the default full-song setting. For entries marked $\star$ (Stage 2 w/ mask and Stage 3), we fix $\rho = 0.5$ and sample α, β per call: a window-length ratio r is either fixed or drawn from a curriculum range $r \sim \mathcal{U}(r_{\min}, r_{\max})$; then $\alpha \sim \mathcal{U}(0, 1 - r)$ and $\beta = \alpha + r$.

B. Task Setup and Baseline Configurations

The partial-reference cover task (Task 2; Section 4.2) exposes only a fragment of the reference recording to the model. For each of the 32 reference pieces $i = 1, 2, \ldots, 32$, we manually label the start and end α_i, β_i of the first chorus. The three setups—continuation, outpainting, and inpainting—are then specified by instantiating the active-position mask $\mathcal{A}(\alpha, \beta, \rho)$ from Eq. 5 as follows. *(i)* **Continuation** $[\mathcal{A}(0, \alpha_i, 0)]$: the prefix from the song's start through the

section preceding the first chorus is observed and the remainder is generated. *(ii)* **Outpainting** $[\mathcal{A}(\alpha_i, \beta_i, 0)]$: only the first chorus is observed and the rest of the song (preceding and following) is generated outward from it. *(iii)* **Inpainting** $[\mathcal{A}(\alpha_i, \beta_i, 1)]$: everything except the first chorus is observed, and the model is asked to fill in the missing chorus.

For each configuration, the observed fragment supplies the tonal-content condition for the corresponding region, while the unobserved region is generated freely under the target lyrics and style tag.

For SongEcho [11], we refer to its official implementation[2]. For inference on Task 1, we extract the f0 sequence using its provided RMVPE model and follow its official inference Python script. We use its released checkpoint with all other parameters set to default. For Task 2, the SongEcho paper claims support for inpainting and outpainting, also showcased on their demo page[3]; however, the relevant instructions and inference script for this setting are not found in their open-sourced code. We therefore refer to its description in the original paper, which we quote here: "Our method supports inpainting and outpainting via a simple masking strategy" (Sec. 5.3.2 of the SongEcho paper), and implement a masking strategy based on our understanding: for a given reference music, the observed part is kept as the condition, and the rest is masked out as zero silence.

For ACE-Step 1.5 [12], we refer to its official implementation[4]. We use its turbo version. We refer to its official documentation for cover song generation[5]. Besides lyrics and style prompts, the influence of the reference music is controlled by the parameter `audio_cover_strength`. When this parameter is set too high (≥ 0.4), we empirically find that the model copies the original song; when set too low (≤ 0.2), it cannot preserve the tonal content and therefore cannot be identified as a cover given the reference. To ensure a fair comparison and a genuine baseline performance, we set `audio_cover_strength` to 0.3. We additionally report ACE-Step 1.5's objective metrics in Table 6 while iterating through `audio_cover_strength` λ from 0.1 to 0.8. All the metrics converge as $\lambda \geq 0.4$, with 100 percent accuracy for cover song identification model. It is exactly caused by the model copying the original song and producing identical inference results. While as $\lambda = 0.1$ or 0.2, the accuracy of cover song identification is low, indicating that the model can not preserve the necessary tonal content that a cover version should keep invariant. For Task 2, ACE-Step 1.5 does not natively support our setting; however, this can be achieved by cascading its cover and repaint features. Cascading approach: for a reference music, split it into two parts—the partial-condition part and the remaining part. For the partial-condition part, we separately use the cover feature to generate a cover, and we substitute the original portion with the generated part. We then feed the whole song (partial-condition part replaced by the cover result while the remaining stays the same) into the model and ask it to repaint the rest.

[2] https://github.com/lsfhuihuiff/SongEcho_ICLR2026
[3] https://vvanonymousvv.github.io/SongEcho_updated/
[4] https://github.com/ace-step/ACE-Step-1.5
[5] https://github.com/ace-step/ACE-Step-1.5/blob/main/docs/en/INFERENCE.md#2-cover

Table 6 | Objective evaluation results under different audio cover control strengths λ for ACE-Step 1.5.

AudioBox quality (↑) [53]

Method	PQ	PC	CE	CU
ACE-Step-1.5 $\lambda = 0.1$	8.25	5.58	7.64	7.80
ACE-Step-1.5 $\lambda = 0.2$	8.23	5.87	7.64	7.78
ACE-Step-1.5 $\lambda = 0.3$	8.28	6.30	7.67	7.86
ACE-Step-1.5 $\lambda = 0.4$	8.28	6.40	7.64	7.87
ACE-Step-1.5 $\lambda = 0.5$	8.27	6.43	7.61	7.86
ACE-Step-1.5 $\lambda = 0.6$	8.27	6.42	7.61	7.85
ACE-Step-1.5 $\lambda = 0.7$	8.25	6.43	7.60	7.84
ACE-Step-1.5 $\lambda = 0.8$	8.25	6.44	7.59	7.83

Table 6 (continued).

SongEval aesthetics (↑) [54]

Method	Coh.	Mem.	Nat.	Cla.	Mus.
ACE-Step-1.5 $\lambda = 0.1$	4.21	4.17	3.93	4.08	4.07
ACE-Step-1.5 $\lambda = 0.2$	4.28	4.25	4.03	4.14	4.17
ACE-Step-1.5 $\lambda = 0.3$	4.48	4.46	4.25	4.34	4.36
ACE-Step-1.5 $\lambda = 0.4$	4.51	4.49	4.29	4.37	4.38
ACE-Step-1.5 $\lambda = 0.5$	4.51	4.50	4.29	4.38	4.38
ACE-Step-1.5 $\lambda = 0.6$	4.51	4.49	4.28	4.38	4.38
ACE-Step-1.5 $\lambda = 0.7$	4.51	4.49	4.28	4.38	4.38
ACE-Step-1.5 $\lambda = 0.8$	4.51	4.50	4.27	4.38	4.37

Table 6 (continued).

Lyrics, style, and cover identification

Method	PER (↓)	MuQ-Sim (↑)	MuQ-Dist (↑)	CSI-R@1 (↑)	CSI-Rank (↓)
ACE-Step-1.5 $\lambda = 0.1$	0.37	0.39	0.56	0.08	12.40
ACE-Step-1.5 $\lambda = 0.2$	0.29	0.37	0.50	0.25	8.03
ACE-Step-1.5 $\lambda = 0.3$	0.21	0.30	0.40	0.95	1.10
ACE-Step-1.5 $\lambda = 0.4$	0.13	0.27	0.36	1.00	1.00
ACE-Step-1.5 $\lambda = 0.5$	0.11	0.27	0.34	1.00	1.00
ACE-Step-1.5 $\lambda = 0.6$	0.12	0.27	0.34	1.00	1.00
ACE-Step-1.5 $\lambda = 0.7$	0.13	0.27	0.33	1.00	1.00
ACE-Step-1.5 $\lambda = 0.8$	0.12	0.27	0.33	1.00	1.00

C. Partial-Reference Cover: Per-Configuration Results

Table 2 in the main text reports objective scores averaged over the three masking configurations. Tables 7–9 report the per-configuration breakdown. Among each configuration, MuLaCover achieves superior results over baselines.

Table 7 | Partial-reference cover, *continuation* configuration. Same metrics as Table 1.

AudioBox quality (↑) [53]

Method	PQ	PC	CE	CU
SongEcho [11]	7.49	5.44	6.63	7.43
ACE-Step-1.5 [12]	8.26	5.86	7.65	7.83
MuLaCover	8.33	6.33	7.71	7.93

Table 7 (continued).

SongEval aesthetics (↑) [54]

Method	Coh.	Mem.	Nat.	Cla.	Mus.
SongEcho [11]	3.06	2.98	2.76	2.77	2.83
ACE-Step-1.5 [12]	4.19	4.17	3.91	4.05	4.04
MuLaCover	4.59	4.59	4.33	4.44	4.47

Table 7 (continued).

Lyrics, style, and cover identification

Method	PER (↓)	MuQ-Sim (↑)	MuQ-Dist (↑)	CSI-R@1 (↑)	CSI-Rank (↓)
SongEcho [11]	0.64	0.19	0.70	0.25	7.59
ACE-Step-1.5 [12]	0.36	0.35	0.51	0.38	6.18
MuLaCover	0.11	0.42	0.51	0.63	3.15

Table 8 | Partial-reference cover, *outpainting* configuration. Same metrics as Table 1.

AudioBox quality (↑) [53]

Method	PQ	PC	CE	CU
SongEcho [11]	7.41	5.43	6.56	7.32
ACE-Step-1.5 [12]	8.23	5.71	7.62	7.82
MuLaCover	8.31	6.27	7.66	7.91

Table 8 (continued).

SongEval aesthetics (↑) [54]

Method	Coh.	Mem.	Nat.	Cla.	Mus.
SongEcho [11]	2.87	2.76	2.63	2.58	2.67
ACE-Step-1.5 [12]	4.17	4.13	3.89	4.02	4.01
MuLaCover	4.54	4.54	4.29	4.39	4.43

Table 8 (continued).

Lyrics, style, and cover identification

Method	PER (↓)	MuQ-Sim (↑)	MuQ-Dist (↑)	CSI-R@1 (↑)	CSI-Rank (↓)
SongEcho [11]	0.64	0.20	0.72	0.25	7.61
ACE-Step-1.5 [12]	0.40	0.36	0.53	0.38	6.89
MuLaCover	0.12	0.42	0.51	0.55	3.36

Table 9 | Partial-reference cover, *inpainting* configuration. Same metrics as Table 1.

AudioBox quality (↑) [53]

Method	PQ	PC	CE	CU
SongEcho [11]	7.83	5.90	7.20	7.56
ACE-Step-1.5 [12]	8.30	6.16	7.70	7.87
MuLaCover	8.34	6.33	7.70	7.93

Table 9 (continued).

SongEval aesthetics (↑) [54]

Method	Coh.	Mem.	Nat.	Cla.	Mus.
SongEcho [11]	3.50	3.37	3.17	3.27	3.24
ACE-Step-1.5 [12]	4.31	4.30	4.06	4.16	4.19
MuLaCover	4.58	4.58	4.33	4.43	4.46

Table 9 (continued).

Lyrics, style, and cover identification

Method	PER (↓)	MuQ-Sim (↑)	MuQ-Dist (↑)	CSI-R@1 (↑)	CSI-Rank (↓)
SongEcho [11]	0.38	0.35	0.46	0.73	2.62
ACE-Step-1.5 [12]	0.32	0.32	0.46	0.82	1.68
MuLaCover	0.13	0.41	0.49	0.93	1.28

D. Subjective Evaluation: Listening Study

Protocol. On each rating page the participant is presented with the reference music, the target style tag, and the target lyrics, and is asked to score the cover on four 5-point Likert scales: *Identity Preserveness*, *Style Controllability*, *Creativeness*, and *Overall Musicality*. The verbatim wording of each metric and the anchor description for each of the scores 1–5 are shown on the case page in Figure 3. Before the rating phase the participant reports age and self-assessed musical background (amateur, intermediate, or professional). Participants could stop at any point, with completed responses auto-saved, so as to mitigate listening fatigue. All participants in both the listening study and the expert interview were unpaid volunteers and gave informed consent on the welcome page before starting the questionnaire.

Participants. We received 34 saved submissions. After excluding 12 submissions with no complete case and dropping incomplete case blocks, the analysis retained 22 participants self-reporting their musical background as amateur (11), intermediate (6), or professional (5). Among participants with valid age responses (n=18), age was 25.5 ± 4.1 (range 21–33). Each retained participant completed 4.32 cases on average, yielding 95 complete (participant, case) blocks and 1,520 ratings in total. A complete block is a participant–case tuple in which all four systems were rated on all four metrics. Table 10 breaks the average rating down by self-reported musical background, pooled across systems.

Omnibus test across systems. Because every (participant, case) block receives one rating from each of the four systems on each metric, the rating data is within-subject and paired.

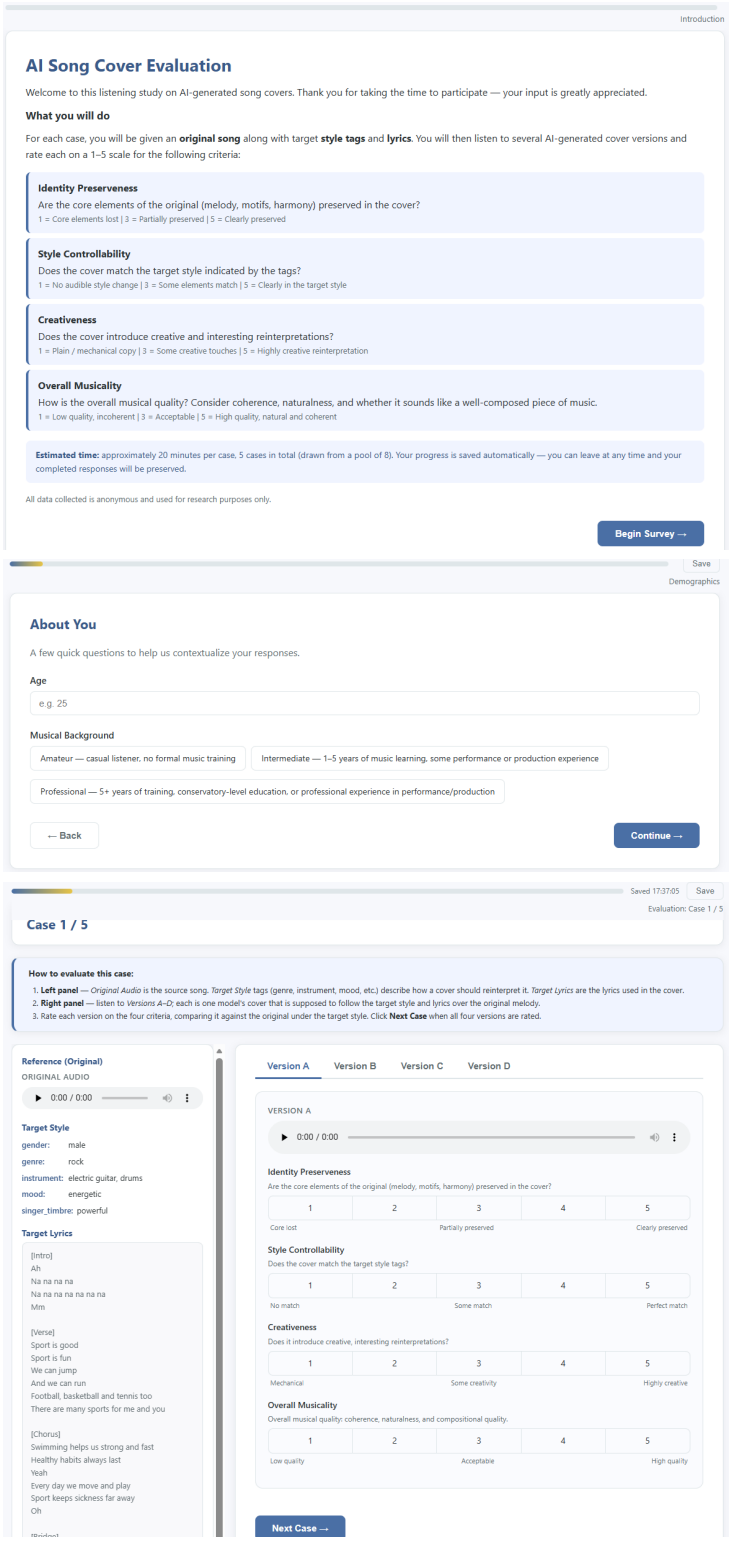

Figure 3 | Listening-study questionnaire interface. Top: welcome page. Middle: demographics page. Bottom: per-case rating page, listing the four metrics together with the textual anchors for scores 1–5.

Table 10 | Mean rating per metric by self-reported musical background, pooled across systems, with cluster-bootstrap 95% CI in brackets (resampling at the participant level, 1,000 replicates). n is the number of (participant, case, system) ratings contributing to each row, per metric.

Background	n	Identity Preserveness	Style Controllability	Creativeness	Overall Musicality
Amateur	200	3.54 [3.27, 3.79]	3.38 [3.12, 3.62]	3.21 [3.00, 3.40]	3.50 [3.35, 3.67]
Intermediate	100	3.25 [2.54, 3.80]	3.08 [2.37, 3.70]	3.16 [2.69, 3.58]	3.26 [2.82, 3.64]
Professional	80	3.26 [2.82, 3.88]	3.36 [2.67, 4.05]	3.17 [2.67, 3.51]	2.89 [2.28, 3.34]

We therefore use the complete participant–case tuple as the paired analysis block: for each metric, the k=4 system ratings are ranked within each of the n_{blocks}=95 blocks and tested for system-level rank differences with a Friedman rank test. The 95 complete blocks come from 112 evaluated cases after dropping 17 incomplete cases, and each complete block contributes 4×4=16 ratings. Table 11 reports the test statistic, p-value, and Kendall's coefficient of concordance W as effect size. Significant system-level differences are observed on every one of the four metrics, with W ranging from 0.19 (Creativeness) to 0.31 (Overall Musicality), motivating the pairwise comparisons below. We report these p-values as paired block-level evidence of within-block system differences, rather than as evidence from 95 fully independent subject-level observations.

Table 11 | Friedman omnibus test across the four systems, per metric. n_{blocks} is the number of (participant, case) blocks within which the k=4 system ratings are ranked. Kendall's W is the coefficient-of-concordance effect size.

Metric	n_{blocks}	χ^2	p-value	Kendall's W
Identity Preserveness	95	57.19	2.3×10^{-12}	0.20
Style Controllability	95	55.69	4.9×10^{-12}	0.20
Creativeness	95	52.90	1.9×10^{-11}	0.19
Overall Musicality	95	88.08	5.7×10^{-19}	0.31

Pairwise post-hoc tests. Conditional on the omnibus rejecting on every metric, we run pairwise Wilcoxon signed-rank tests on the within-block (participant, case) score differences for each of the $\binom{4}{2}$=6 system pairs, and apply a Bonferroni correction over the 6 comparisons within each metric. Table 12 reports the corrected p-values; bold entries are significant at the 0.05 level. Three observations follow. *(i)* On Style Controllability, Creativeness, and Overall Musicality, MuLaCover shows no significant difference from Suno-v5.5 (p_{bonf}=1.0 on all three) while significantly outperforming both ACE-Step-1.5 ($p_{\text{bonf}}\leq$0.017) and SongEcho ($p_{\text{bonf}}<10^{-4}$ on all three). *(ii)* On Identity Preserveness, MuLaCover significantly outperforms SongEcho (p_{bonf}=0.0001) and shows no significant difference from ACE-Step-1.5 (p_{bonf}=1.0), while Suno-v5.5 remains significantly higher than MuLaCover (p_{bonf}=0.006). *(iii)* SongEcho is significantly lower than MuLaCover and Suno-v5.5 on all four metrics; compared with ACE-Step-1.5, SongEcho is lower on Identity Preserveness, Creativeness, and Overall Musicality, but the Style Controllability difference is not significant.

Table 12 | Bonferroni-corrected p-values from pairwise Wilcoxon signed-rank tests across the four systems, per metric. Each test pairs the two systems on the within-block (participant, case) score differences over the n_{blocks}=95 complete blocks; the Bonferroni correction is applied over the 6 pairs within each metric. Bold marks $p < 0.05$.

Pair	Identity	Style	Creativeness	Overall
ACE-Step-1.5 vs. MuLaCover	1.0000	$\mathbf{< 10^{-4}}$	**0.0004**	**0.0161**
ACE-Step-1.5 vs. SongEcho	$\mathbf{< 10^{-4}}$	1.0000	**0.0169**	$\mathbf{< 10^{-4}}$
ACE-Step-1.5 vs. Suno-v5.5	**0.0475**	$\mathbf{< 10^{-4}}$	**0.0007**	$\mathbf{< 10^{-4}}$
MuLaCover vs. SongEcho	**0.0001**	$\mathbf{< 10^{-4}}$	$\mathbf{< 10^{-4}}$	$\mathbf{< 10^{-4}}$
MuLaCover vs. Suno-v5.5	**0.0060**	1.0000	1.0000	1.0000
SongEcho vs. Suno-v5.5	$\mathbf{< 10^{-4}}$	$\mathbf{< 10^{-4}}$	$\mathbf{< 10^{-4}}$	$\mathbf{< 10^{-4}}$

E. Subjective Evaluation: Expert Interviews

Protocol. The expert interview is designed to explain the Likert results without letting the evaluator choose criteria after hearing the generated audio. In each session, we first sample a target genre from a catalogue of common popular-music styles. Before any audio is played, the expert is asked to list the most salient musical attributes of that genre and briefly explain how each attribute should be judged. Only after these attributes are fixed do we present the blinded covers from the four systems. The expert then scores and comments on each version using the pre-elicited attributes as the evaluation criteria. Finally, we feed the organized expert record to a language model and ask it to produce a concise, neutral summary. The respondent identifier and date are anonymized in the record below.

R&B case study. The sampled genre in the recorded session is R&B. The four blinded versions were later decoded as Version A = ACE-Step-1.5, Version B = SongEcho, Version C = MuLaCover, and Version D = Suno-v5.5. The tables and notes below are translated from the original Chinese notes and lightly edited for readability while preserving the expert's recorded ratings and judgments.

Table 13 | Primary R&B attributes elicited from the expert before any generated audio was played.

Attribute	Expert definition / judging criterion
Rhythm and groove	The most salient property of rhythm-and-blues: drums and bass jointly form the groove, and faithful R&B should exhibit typical rhythmic patterns rather than a generic pop or electronic feel.
Vocal melody, repetition, and variation	In R&B, melodic development often happens through modular repetition and varied repetition over a cyclic rhythmic/harmonic frame; phrase-end ornaments, melismas, and vocal timbre are important cues.

The expert also listed harmonic looping and instrumentation as auxiliary references, but described them as less diagnostic than rhythm/groove and vocal melodic treatment. We therefore report only the two primary criteria in Table 14. In the original notes, the auxiliary comments are sparse: ACE-Step-1.5 receives a harmony score of 3 without an additional written

Table 14 | Organized expert record for the R&B case study. Each attribute follows the pre-listening criteria in Table 13. The table focuses on the two primary attributes for which the expert gave complete scores.

Version / system	Rhythm and groove	Vocal melody / timbre	Overall comment
A / ACE-Step-1.5	2: Electric drums, 808, and synthetic bass make the version sound closer to electronic pop.	2: The vocal line is closer to a normal pop-song melody than to R&B melodic development.	The first two attributes deviate substantially from typical R&B; the version leans toward electronic pop rather than canonical R&B.
B / SongEcho	3: The drum timbre and groove contain pop-R&B characteristics.	3: The vocal development is closer to R&B: it uses a four-phrase loop rather than the linear emotional buildup typical of pop ballads.	Overall, this version is relatively better than Version A.
C / MuLaCover	4: The expert rated the rhythm and groove higher than Versions A and B.	4: The expert rated the vocal melodic treatment higher than Versions A and B; the verse also has a vocal timbre that sounds very R&B-like.	The expert judged it close to Version B in broad style but with a more R&B-like singer timbre; it is still not the most canonical R&B rendition.
D / Suno-v5.5	4.5: The groove is slightly better than the preceding versions.	4.5: The vocal line is much more natural and close to pop R&B, especially in phrase endings, melismas, and timbre.	The expert also noted that the auxiliary harmonic and instrumentation-related attributes were relatively well matched.

rationale, and Suno-v5.5 is described in the overall comment as relatively well matched on the harmonic and instrumentation-related cues.

Prompt used for the language-model summary. For step (iv), we used the following prompt template to convert the organized expert record into a concise verdict. To avoid first-person or system-ownership bias, the prompt asks the language model to reason over the blinded version labels only. The version labels were mapped back to system names after the summary was produced.

Input: A structured expert-interview record for an AI song-cover evaluation. The record contains: (1) the sampled target genre; (2) the musical attributes elicited from the expert before any generated audio was played; and (3) blinded versions A–D with per-attribute scores and expert comments.
Task: Produce a concise 4–6 sentence neutral research summary contrasting Versions A–D. Base the verdict only on the expert's stated attributes, scores, and comments. Mention the genre-specific criteria that drive the ranking. Do not add claims that are not supported by the expert record. Do not use first-person labels such as "our model" or infer that any version should be preferred from its label.
Desired output: A short neutral paragraph contrasting Version A, Version B, Version C, and Version D.

F. Limitations and Broader Impacts

Limitations. MuLaCover inherits the linguistic and stylistic coverage of its text-to-song backbone (HeartMuLa) and of the internal cross-cover dataset, so genres, languages, and vocal styles under-represented in those sources are not guaranteed to transfer faithfully. The symbolic conditioning relies on off-the-shelf melody and chord transcription models, whose errors propagate into the generated cover. Finally, evaluation is conducted on a single 32-song test set with an LLM-curated style catalogue; results may shift on broader in-the-wild repertoires.

Broader impacts. Cover generation can democratize music creation and assist musicians in prototyping reinterpretations. The same capability, however, raises concerns around copyright (covers built from copyrighted source recordings), attribution (potential confusion with original artists), and voice or style imitation. We encourage downstream users to obtain rights to source recordings, to clearly disclose machine-generated covers, and to avoid imitating the identifiable voice of specific living artists without consent. We will release the model under a license that requires adherence to these responsible-use guidelines.

References

[1] Joan Serrà, Emilia Gómez, and Perfecto Herrera. *Audio Cover Song Identification and Similarity: Background, Approaches, Evaluation, and Beyond*, pages 307–332. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-11674-2. doi: 10.1007/978-3-642-11674-2_14. URL https://doi.org/10.1007/978-3-642-11674-2_14.

[2] Daniel P.W. Ellis and Graham E. Poliner. Identifying 'cover songs' with chroma features and dynamic programming beat tracking. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, volume 4, pages IV–1429–IV–1432, 2007. doi: 10.1109/ICASSP.2007.367348.

[3] Joan Serra, Emilia Gomez, Perfecto Herrera, and Xavier Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(6):1138–1151, 2008. doi: 10.1109/TASL.2008.924595.

[4] Zhesong Yu, Xiaoshuo Xu, Xiaoou Chen, and Deshun Yang. Learning a representation for cover song identification using convolutional neural network. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 541–545, 2020. doi: 10.1109/ICASSP40776.2020.9053839.

[5] Xingjian Du, Zhesong Yu, Bilei Zhu, Xiaoou Chen, and Zejun Ma. Bytecover: Cover song identification via multi-loss training. In *ICASSP 2021 - 2021 IEEE International Conference on*

Acoustics, Speech and Signal Processing (ICASSP), pages 551–555, 2021. doi: 10.1109/ICASSP39728.2021.9414128.

[6] Xingjian Du, Zijie Wang, Xia Liang, Huidong Liang, Bilei Zhu, and Zejun Ma. Bytecover3: Accurate cover song identification on short queries. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10095389.

[7] Feng Liu, Deyi Tuo, Yinan Xu, and Xintong Han. CoverHunter: Cover Song Identification with Refined Attention and Alignments . In *2023 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1080–1085, Los Alamitos, CA, USA, July 2023. IEEE Computer Society. doi: 10.1109/ICME55011.2023.00189. URL https://doi.ieeecomputersociety.org/10.1109/ICME55011.2023.00189.

[8] 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.

[9] 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.

[10] 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*, 2026.

[11] 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. In *The Fourteenth International Conference on Learning Representations*.

[12] 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.

[13] Ye Bai, Haonan Chen, Jitong Chen, Zhuo Chen, Yi Deng, Xiaohong Dong, Lamtharn Hantrakul, Weituo Hao, Qingqing Huang, Zhongyi Huang, et al. Seed-music: A unified framework for high quality and controlled music generation. *arXiv preprint arXiv:2409.09214*, 2024.

[14] Alexandre Papadopoulos, Pierre Roy, and François Pachet. Assisted lead sheet composition using flowcomposer. In Michel Rueher, editor, *Principles and Practice of Constraint Programming*, pages 769–785, Cham, 2016. Springer International Publishing. ISBN 978-3-319-44953-1.

[15] Hao-Min Liu and Yi-Hsuan Yang. Lead sheet generation and arrangement by conditional generative adversarial network. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 722–727, 2018. doi: 10.1109/ICMLA.2018.00114.

[16] 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.

[17] 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. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[18] Wei-Ho Tsai, Hung-Ming Yu, Hsin-Min Wang, et al. Query-by-example technique for retrieving cover versions of popular songs with similar melodies. In *ISMIR*, volume 5, pages 183–190, 2005.

[19] Özgür İzmirli. Tonal similarity from audio using a template based attractor model. In *ISMIR*, pages 540–545, 2005.

[20] Mark A Bartsch and Gregory H Wakefield. To catch a chorus: Using chroma-based representations for audio thumbnailing. In *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No. 01TH8575)*, pages 15–18. IEEE, 2001.

[21] Emilia Gómez. Tonal description of music audio signals. *Department of Information and Communication Technologies*, 2006.

[22] Furkan Yesiler, Joan Serrà, and Emilia Gómez. Accurate and scalable version identification using musically-motivated embeddings. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 21–25, 2020. doi: 10.1109/ICASSP40776.2020.9053793.

[23] 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.

[24] 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.

[25] 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.

[26] Liwei Lin, Gus Xia, Junyan Jiang, and Yixiao Zhang. Content-based controls for music large language modeling. *arXiv preprint arXiv:2310.17162*, 2023.

[27] Fang-Duo Tsai, Shih-Lun Wu, Weijaw Lee, Sheng-Ping Yang, Bo-Rui Chen, Hao-Chung Cheng, and Yi-Hsuan Yang. Musecontrollite: Multifunctional music generation with lightweight conditioners. In *Forty-second International Conference on Machine Learning*.

[28] Yun-Han Lan, Wen-Yi Hsiao, Hao-Chung Cheng, and Yi-Hsuan Yang. Musicongen: Rhythm and chord control for transformer-based text-to-music generation. *arXiv preprint arXiv:2407.15060*, 2024.

[29] Peiling Lu, Xin Xu, Chenfei Kang, Botao Yu, Chengyi Xing, Xu Tan, and Jiang Bian. Musecoco: Generating symbolic music from text. *CoRR*, abs/2306.00110, 2023. doi: 10.48550/ARXIV.2306.00110. URL https://doi.org/10.48550/arXiv.2306.00110.

[30] Hao-Wen Dong, Ke Chen, Shlomo Dubnov, Julian J. McAuley, and Taylor Berg-Kirkpatrick. Multitrack music transformer. In *ICASSP 2023, Rhodes Island, Greece, June 4-10, 2023*, pages 1–5. IEEE, 2023. doi: 10.1109/ICASSP49357.2023.10094628. URL https://doi.org/10.1109/ICASSP49357.2023.10094628.

[31] Dimitri von Rütte, Luca Biggio, Yannic Kilcher, and Thomas Hofmann. FIGARO: controllable music generation using learned and expert features. In *ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=NyR8OZFHw6i.

[32] Ziyu Wang, Dingsu Wang, Yixiao Zhang, and Gus Xia. Learning interpretable representation for controllable polyphonic music generation. In *ISMIR 2020, Montreal, Canada, October 11-16, 2020*, pages 662–669, 2020. URL http://archives.ismir.net/ismir2020/paper/000094.pdf.

[33] Lejun Min, Junyan Jiang, Gus Xia, and Jingwei Zhao. Polyffusion: A diffusion model for polyphonic score generation with internal and external controls. In *ISMIR 2023, Milan, Italy, November 5-9, 2023*, pages 231–238, 2023. doi: 10.5281/ZENODO.10265265. URL

https://doi.org/10.5281/zenodo.10265265.

[34] Xiaoyu Yi, Qi He, Gus Xia, and Ziyu Wang. Vitex: Visual texture control for multi-track symbolic music generation via discrete diffusion models. In *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 14797–14801. IEEE, 2026.

[35] Ziyu Wang, Lejun Min, and Gus Xia. Whole-song hierarchical generation of symbolic music using cascaded diffusion models. In *ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=sn7CYWyavh.

[36] Ziyu Wang, Dejing Xu, Gus Xia, and Ying Shan. Audio-to-symbolic arrangement via cross-modal music representation learning. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 181–185. IEEE, 2022.

[37] Jingwei Zhao, Ziyu Wang, Gus Xia, and Ye Wang. Bossa: Learning music style through cross-modal bootstrapping. In *AI for Music Workshop*.

[38] Chih-Pin Tan, Shuen-Huei Guan, and Yi-Hsuan Yang. Picogen: Generate piano covers with a two-stage approach. In *Proceedings of the 2024 International Conference on Multimedia Retrieval*, pages 1180–1184, 2024.

[39] Sungkyun Chang, Emmanouil Benetos, Holger Kirchhoff, and Simon Dixon. Yourmt3+: Multi-instrument music transcription with enhanced transformer architectures and cross-dataset stem augmentation. In *2024 IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2024.

[40] Junyan Jiang, Ke Chen, Wei Li, and Gus Xia. Large-vocabulary chord transcription via chord structure decomposition. In *ISMIR*, pages 644–651, 2019.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[42] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[43] Changhao Jiang, Jiahao Chen, Zhenghao Xiang, Zhixiong Yang, Hanchen Wang, Jiabao Zhuang, Xinmeng Che, Jiajun Sun, Hui Li, Yifei Cao, et al. Muse: Towards reproducible long-form song generation with fine-grained style control. *arXiv preprint arXiv:2601.03973*, 2026.

[44] Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis Bach. Demucs: Deep extractor for music sources with extra unlabeled data remixed. *arXiv preprint arXiv:1909.01174*, 2019.

[45] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pages 28492–28518. PMLR, 2023.

[46] Taejun Kim and Juhan Nam. All-in-one metrical and functional structure analysis with neighborhood attentions on demixed audio. In *2023 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 1–5. IEEE, 2023.

[47] Chunbo Hao, Ruibin Yuan, Jixun Yao, Qixin Deng, Xinyi Bai, Wei Xue, and Lei Xie. Songformer: Scaling music structure analysis with heterogeneous supervision. *arXiv preprint arXiv:2510.02797*, 2025.

[48] Yunfei Chu, Jin Xu, Qian Yang, Haojie Wei, Xipin Wei, Zhifang Guo, Yichong Leng, Yuanjun Lv, Jinzheng He, Junyang Lin, et al. Qwen2-audio technical report. *arXiv preprint arXiv:2407.10759*, 2024.

[49] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao,

Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[50] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

[51] Mar 2026. URL https://suno.com/blog/v5-5.

[52] Mureka. Mureka: AI music generator. https://www.mureka.ai/, n.d. Accessed: 2026-09-15.

[53] Andros Tjandra, Yi-Chiao Wu, Baishan Guo, John Hoffman, Brian Ellis, Apoorv Vyas, Bowen Shi, Sanyuan Chen, Matt Le, Nick Zacharov, et al. Meta audiobox aesthetics: Unified automatic quality assessment for speech, music, and sound. *arXiv preprint arXiv:2502.05139*, 2025.

[54] Jixun Yao, Guobin Ma, Huixin Xue, Huakang Chen, Chunbo Hao, Yuepeng Jiang, Haohe Liu, Ruibin Yuan, Jin Xu, Wei Xue, et al. Songeval: A benchmark dataset for song aesthetics evaluation. *arXiv preprint arXiv:2505.10793*, 2025.

[55] 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.