COMPOSERFLOW: STEP-BY-STEP COMPOSITIONAL SONG GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Open-source end-to-end song generators have emerged in the wake of commercial systems like Suno, offering one-shot music with lyrics but at the cost of large datasets and substantial compute. We propose ComposerFlow, a hierarchical pipeline that composes songs by chaining four specialized components—lyrics-to-melody, melody harmonization, singing-voice synthesis (SVS), and singing accompaniment generation (SAG). This modular design is resource-efficient (trained on a single RTX 3090; 6k hours of data) and editable: users can revise intermediate results (e.g., melody, chords, vocal) and regenerate corresponding downstream results. We evaluate our pipeline against representative end-to-end baselines and observe perceptual quality comparable to open-source systems, while requiring significantly less training. We release audio demos at https://composerflow.github.io/web/. Our results suggest that modular, controllable pipelines are a practical alternative to end-to-end song models, enabling rapid iteration and reliable production on accessible hardware.

1 INTRODUCTION

Traditional song production is a staged, collaborative workflow that progresses from songwriting (lyrics, melody, form) to arrangement and pre-production, then multitrack recording, editing, mixing, and mastering. Each stage depends on specialized expertise—producers, topliners, instrumentalists, and engineers—and tight iterative feedback in a DAW, where decisions about key, tempo, harmony, vocal delivery, and sound design are refined across multiple passes. While this process is flexible and quality-driven, it is time- and resource-intensive; revisions at any stage often cascade into downstream rework. Producers preserve editability by saving stems and session states, yet precisely reproducing changes remains laborious. This conventional pipeline sets a high bar for control and fidelity—highlighting an opportunity for computational systems that retain its editability while reducing cost and turnaround.

With recent advances in generative models, commercial systems such as Suno (2025) and Udio, Inc. (2024) demonstrate high-quality, convenient song generation, followed by emerging open-source models (Yuan et al., 2025; Liu et al., 2025b; Ning et al., 2025; Gong et al., 2025; Yang et al., 2025; Lei et al., 2025; Liu et al., 2025a). In contrast to the conventional studio workflow, these models accept lyrics and textual descriptions and produce full songs within seconds. However, end-to-end generation poses several challenges: (i) when the final output is unsatisfying, intermediate edits are challenging; (ii) directly learning a mapping from lyrics/descriptions to audio is data- and compute-hungry; and (iii) vocals often misalign with the given lyrics and unnatural timbre.

We address these limitations with **ComposerFlow**, a step-by-step pipeline that leverages three small modules and a text-to-music model, trained only with a single RTX 3090. ComposerFlow also begins from lyrics but progressively generates melody, chords, singing voice, and backing track (Figure 1). For lyric-conditioned melody, we use CSL-L2M Chai & Wang (2025); for harmony, we apply AccoMontage2 Yi et al. (2022) to retrieve chord progressions that fit the generated melody. We then synthesize vocals with an SVS model fine-tuned from FastSpeech (Ren et al., 2020). Finally, we perform singing-accompaniment generation using Stable-Audio-Open (Evans et al., 2025) fine-tuned with MuseControlLite (Tsai et al., 2025). This modular design preserves editability: users may adjust melody, chords, or singer identity and deterministically regenerate downstream audio. It also leverages each component's strengths—requiring training only for SVS and MuseC-

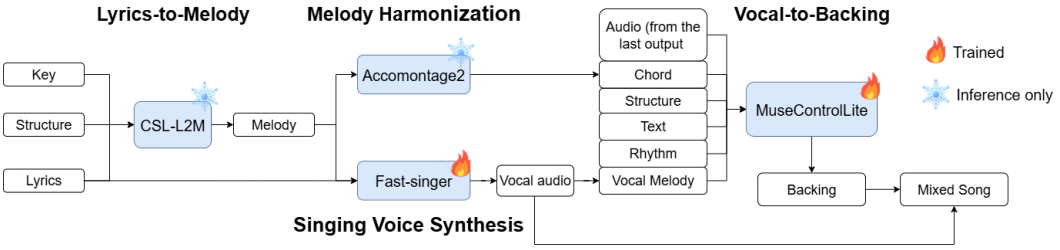

Figure 1: Overview of ComposerFlow. The pipeline begins with lyrics and proceeds through **lyrics-to-melody**, **melody harmonization**, **SVS**, and **singing accompaniment generation**. ComposerFlow is a flexible paradigm rather than a fixed system; any component can be replaced, provided it adheres to the same input and output specifications.

ontrolLite (Tsai et al., 2025). To mitigate copyright concerns, we trained the SVS on two licensed singers. For accompaniment, we condition Stable-Audio-Open on time-varying controls (chords, vocal melody, rhythm, reference audio, key, and structure) to ensure the backing track follows the vocal. All training runs on a single RTX 3090 GPU, substantially lowering training costs while improving lyric-vocal alignment and timbral quality. We present the first open-source pipeline that decomposes lyrics-to-song generation into editable stages, allowing low-cost fine-tuning and mid-pipeline modification. In particular, we adapt a text-to-music model for singing accompaniment generation, and evaluate our approach against state-of-the-art end-to-end systems using both objective and subjective metrics. Results show comparable perceptual quality to open-source models while requiring substantially less training data and compute.

System	Training data (hrs)	Compute resources
Yue (Yuan et al., 2025)	650K	16× H800
DiffRhythm (Liu et al., 2022)	60K	8× Huawei Ascend 910B
Acestep (Gong et al., 2025)	100K	120× A100
SongBloom (Yang et al., 2025)	100K	16× A100
JAM (Liu et al., 2025a)	54K	8× H100
Levo (Lei et al., 2025)	110K	8× A100
Ours	**6K**	**1 × 3090**

Table 1: Comparison of training data and compute resources across Song Generation systems. Our proposed **ComposerFlow** used significantly less training data and resources, serving as an eco-friendly song generation pipeline.

2 Related Works

2.1 Text-to-music

Generating music purely from text is appealing: it bypasses the complexity of the traditional production pipeline. Building on the success of Stable Diffusion (Rombach et al., 2022) in text-to-image, Riffusion (Forsgren & Martiros, 2022), AudioLDM (Borsos et al., 2023), AudioLDM2 Liu et al. (2024), and MusicLDM (Chen et al., 2024b) adopt latent-diffusion models that generate mel-spectrograms, which are then converted to audio with a vocoder (Kong et al., 2020). Stable Audio (Evans et al., 2024a;b; 2025) accelerates inference and replaces the vocoder with a fully convolutional variational autoencoder. Apart from diffusion, MusicGen Copet et al. (2024) uses a transformer over Encodec tokens Défossez et al. (2022) and can accept audio inputs for melody conditioning. To avoid training from scratch, Music ControlNet Wu et al. (2024a); Hou et al. (2025) fine-tunes ControlNet adapters to equip text-to-music systems with time-varying controls. MuseControlLite Tsai et al. (2025) further reduces the number of trainable parameters while broadening the

set of supported controls. In ComposerFlow, we utilize MuseControlLite with time-varying controls that we consider helpful for singing accompaniment generation.

2.2 SONG GENERATION

Song generation is inherently harder than purely instrumental generation because it composes several sub-tasks—**lyrics-to-melody** (Yu et al., 2021; Sheng et al., 2021; Ju et al., 2021; Chai & Wang, 2025), **singing voice synthesis** (Lu et al., 2020; Chen et al., 2020; Liu et al., 2022), and **singing-accompaniment generation** Donahue et al. (2023); Chen et al. (2024a); Trinh et al. (2024). Recent systems often tackle the lyrics-to-song problem with large end-to-end models trained from scratch that render a full track directly from lyrics. LLM-based approaches such as Yue Yuan et al. (2025) and Levo Lei et al. (2025) produce full-length songs but exhibit slow inference. DiffRhythm Ning et al. (2025) and AceStep Gong et al. (2025) employ diffusion transformers to markedly speed up decoding, while SongBloom Yang et al. (2025) uses an autoregressive diffusion design to pair diffusion-level fidelity with sequence modeling. However, these end-to-end paradigms typically require massive datasets and substantial compute, and they offer limited opportunities to intervene or edit intermediate representations. In contrast, our step-by-step pipeline leverages three lightweight specialist models alongside a text-to-music generator, thereby significantly reducing training costs and data requirements.

3 METHOD

The initial input is lyrics (with structure tags) provided by the users, just like other song generation models, but it also offers optional input for emotion and global key. The intermediate output of every model can be edited and passed to the downstream.

3.1 LYRICS-TO-MELODY

Lyrics-to-Melody. Lyrics-to-melody is challenging because lyrics and melody are only weakly correlated. We adopt the state-of-the-art CSL-L2M model Chai & Wang (2025) to map input lyrics to melody. CSL-L2M is a Transformer decoder with an in-attention mechanism Wu & Yang (2023) that supports fine-grained lyric–melody controls. We condition on the global key[1] and emotion[2], as well as sentence-level structure[3] and statistical musical attributes[4] that strengthen lyric–melody coupling. The statistical attributes are extracted from a melody track. Users may specify emotion, key, and structure (the latter carried by the lyrics), and optionally provide a reference MIDI from which we derive the statistical attributes. The model performs best when the reference melody and input lyrics have similar section structure, line count, and per-line word counts.

To enable use without a user-provided MIDI, we curate a bank of 1,000 reference attribute sets. Given new lyrics, ComposerFlow scores each candidate using a weighted sum

$$P \;=\; 0.4\,P_{\text{sent}} \;+\; 0.4\,P_{\text{prof}} \;+\; 0.2\,P_{\text{struct}},$$

where lower is better. Here, P_{sent} penalizes differences in total line count (optionally rejecting candidates with fewer lines than the target); P_{prof} is the mean absolute difference between per-line token counts—treating each visible Chinese character as one token—after padding the shorter sequence with its median and scaling by the maximum observed token count; and P_{struct} compares section tags mapped to integers, counting position-wise mismatches and adding a penalty for extra sections, normalized by the longer sequence length.

[1] 12 major: C, D♭, D, E♭, E, F, F♯, G, A♭, A, B♭, B; 12 minor: c, c♯, d, d♯, e, f, f♯, g, g♯, a, b♭, b.

[2] Three emotions: Neutral, Positive, Negative.

[3] Five sections: Verse, Chorus, Insertion, Bridge, Outro.

[4] 12 attributes: pitch mean (PM), pitch variance (PV), pitch range (PR), direction of melodic motion (DMM), amount of arpeggiation (AA), chromatic motion (CM), duration mean (DM), duration variance (DV), duration range (DR), prevalence of most common note duration (MCD), note density (ND), fraction of lyric syllables to melody notes (Align).

3.2 SINGING VOICE SYNTHESIS (SVS)

Previous song-generation systems typically synthesize vocals and accompaniment jointly, often yielding unnatural vocal timbre and weak lyric–phoneme alignment. In contrast, given user-provided lyrics and a melody (Section 3.1), we employ a lightweight SVS model to render realistic, controllable singing voices—something end-to-end models struggle to achieve. Accordingly, we decouple the pipeline: the SVS module focuses on high-quality vocal rendering, while accompaniment is generated separately. Specifically, we train FastSpeech (Ren et al., 2020) from scratch on licensed data from two singers (male/female), and add a MIDI-conditioning embedding that aligns each phoneme to its corresponding MIDI note. Before synthesis, we compare the register of the melody to typical vocal ranges—male (lower) vs. female (higher)—and test octave shifts $\Delta \in \{-12, 0, +12\}$ to align the melody with each profile's comfortable tessitura. We then choose the (singer, Δ) that keeps the most notes inside the target range while minimizing $|\Delta|$. By focusing solely on vocal generation, our approach produces more natural, intelligible singing with improved lyric alignment, while remaining model-agnostic—any high-quality SVS model can be plugged into *ComposerFlow*.

3.3 SINGING ACCOMPANIMENT GENERATION (SAG)

In prior Singing Accompaniment Generation (SAG) systems (Donahue et al., 2023; Chen et al., 2024a; Trinh et al., 2024), models typically condition on separated vocals. However, SingSong (Donahue et al., 2023) exhibits slow inference and FastSAG (Chen et al., 2024a) is limited to 10-second outputs. More importantly, conditioning on raw vocals may fail when the vocal track is silent (eg, intro, outro, solo). We therefore replace raw-vocal conditioning with structured, time-varying controls extracted from the data, reducing irrelevant information and enabling more reliable instrumental generation, and use MuseControlLite Tsai et al. (2025) to generate backing tracks to follow the extracted time-varying conditions.

Rhythm. To align backing with vocals, we enforce shared beats and downbeats. We use All-In-One (Kim & Nam, 2023) to obtain beat and downbeat timestamps from training audio and convert them into binary indicator sequences of shape $(T, 1)$, where T is the number of time frames (1 if an event occurs at a frame, else 0). We then apply a Gaussian filter to yield smooth "rhythm activation" curves. At inference, when All-In-One cannot operate on a cappella input, we derive beat/downbeat timing from the quantized MIDI produced by our lyrics-to-melody model (CSL-L2M (Chai & Wang, 2025)), which outputs melodies in 4/4.

Structure. Structure organizes a song's energy and narrative—shaping tension and release—so listeners can follow, feel, and remember the hook. We extract segment tags and timestamps with All-In-One (Kim & Nam, 2023)[5]. Timestamps slice audio at segment boundaries; segment-level captions are produced by AudioFlamingo3 (Goel et al., 2025), and the structure tags themselves are used as a time-varying conditioning signal.

Vocal Melody Vocal melody constrains plausible harmony. We first separate vocals and accompaniment with Mel-Band RoFormer (Wang et al., 2023), then select prominent pitches via the top-4 constant-Q-transform (CQT) method of Hou et al. (2025). In inference, we extract melody from the SVS-generated singing.

Chord. Chord progressions underpin harmonic flow, shape emotion, and support melody. We initially omitted chord conditioning in MuseControlLite (Tsai et al., 2025) and observed unstable harmony and weak progressions. To remedy this, we apply a chord detector (Park et al., 2019) to the separated backing track and encode the results as 12-bin chromagrams (pitch-class membership over time). At inference, chord sequences are provided by AccoMontage2 as described in Section 3.4.

Key. While chords convey strong local tonality, we include a segment-level key condition to capture broader tonal context. We apply key-CNN (Schreiber & Müller, 2019) per structure segment, reflecting that key modulations often occur at section boundaries.

[5]We discard truncated start/end fragments and retain intro, outro, break, bridge, inst, solo, verse, chorus.

Table 2: Evaluating Rhythm and key alignment to the vocal audio

Model / Setting	Rhythm F1	Key acc.
MuseControlLite w/ vocal audio	0.64	0.55
MuseControlLite w/ all condition	0.81	0.90

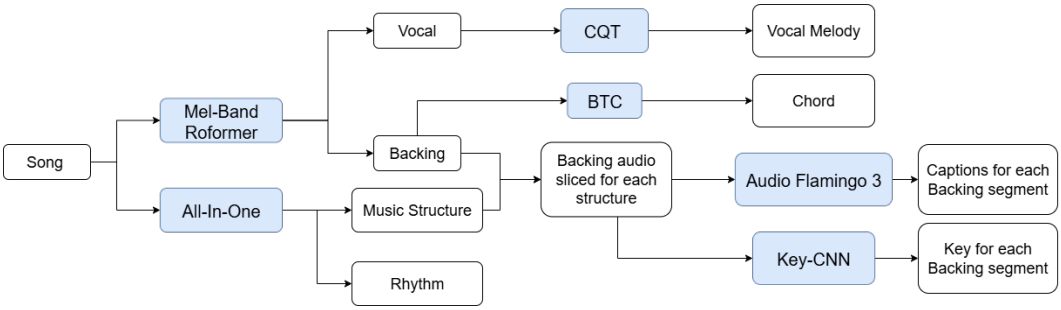

Figure 2: The data-preprocessing pipeline

The full data-processing pipeline is shown in Figure 2. We then train MuseControlLite (Tsai et al., 2025) to follow these controls. To quantify the benefit of explicit controls, we compare against a baseline conditioned only on the raw vocal signal (encoded with the Stable-Audio-Open VAE). As summarized in Table 3.3, the raw-vocal baseline fails to reliably track tempo and key, whereas our control-conditioned model adheres to both—suggesting that direct raw-vocal conditioning is not enough for MuseControlLite (Tsai et al., 2025).

Finally, to overcome the 47 s context limit of Stable-Audio-Open (the MuseControlLite backbone), we adapt and extend the audio-continuation strategy of Tsai et al. (2025) to produce seamless long-form outputs. During training, audio is sliced on structural boundaries: each slice starts at S_i^{start}, where $i \in \{\text{intro}, \text{verse}, \text{chorus}, \text{solo}, \text{inst}, \text{bridge}, \text{break}, \text{outro}\}$, and ends at $S_i^{\text{start}} + 47$. A 47-second window anchored at S_i^{start} may span multiple structural segments. We use the first segment's audio as a reference and train the model to improvise the forward continuation over subsequent segments. We also train *backward* continuation: because intros often lack vocal melody and degrade conditioning, when the target slice begins with an *intro*, we replace the reference with the *chorus* with probability 50%. The overall training setup is illustrated in Figure 3.

3.4 Melody Harmonization

Because chord progressions are used as time-varying controls during training, inference must supply compatible chord sequences. We harmonize the melody produced by CSL-L2M (Chai & Wang, 2025) using AccoMontage2 (Yi et al., 2022). To provide an instrumental lead-in, we prepend a 4-bar intro without melody and reuse (duplicate) the chord sequence from the first 4 melodic bars to harmonize this intro. Melody harmonization supplies chord progression, enabling the singing-accompaniment generator to produce coherent, musically appropriate harmony. If the automatically generated chords are unsatisfactory, users may instead provide their own progression.

4 Experiment Setup

4.1 Training Dataset and Implementation Details

Trainable vs. frozen components. We keep CSL-L2M Chai & Wang (2025) and AccoMontage2 (Yi et al., 2022) fixed, and train only the SVS stack and MuseControlLite. Within MuseControlLite, beyond the adapter, we unfreeze the self-attention blocks of the Stable-Audio-Open backbone (Evans et al., 2025) to improve continuity across structure boundaries.

SVS data and training. The SVS model is trained from scratch on an internal corpus totaling 10 hours from two licensed singers, with aligned MIDI notes, phoneme labels, and ground-truth mel-

Table 3: Per-module inference latency for generating a 90–120 seconds *intro–verse–chorus–outro* song. For the singing-accompaniment stage, MuseControlLite (Tsai et al., 2025) is configured with 50 denoising steps.

Module	**Time (s)**
Lyrics-to-Melody	10
Melody Harmonization	0.2
Singing Voice Synthesis	3
singing accompaniment generation	40

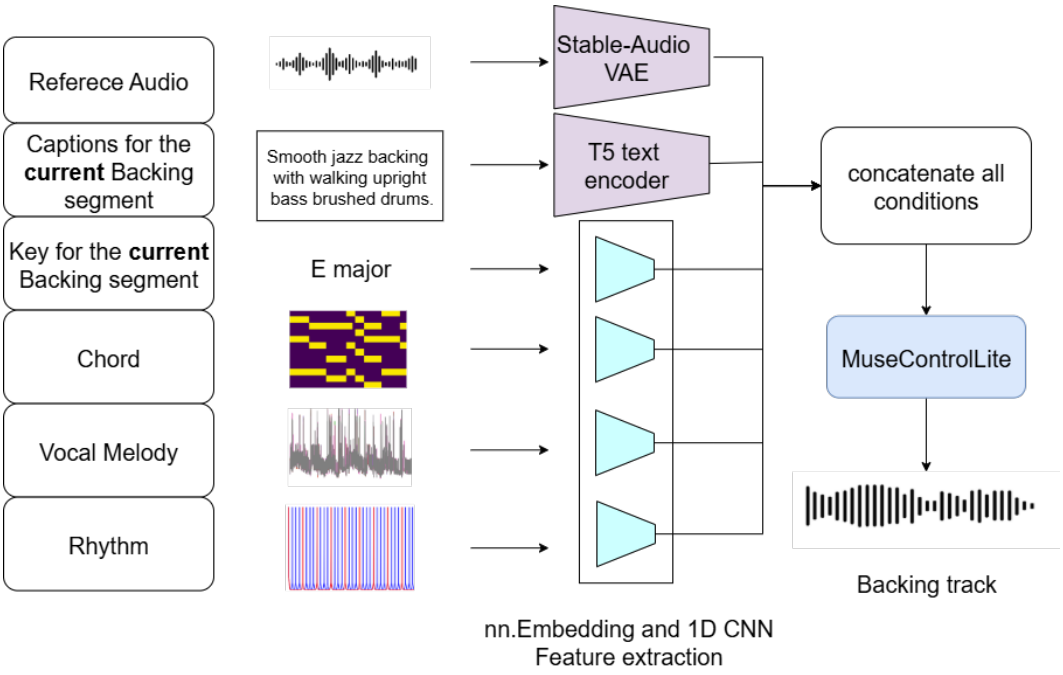

Figure 3: Equipping Stable-Audio Open with singing accompaniment generation and audio continuation abilities via MuseControlLite.

spectrograms. Training uses a single NVIDIA RTX 3090 for 24 hours. We then fine-tune a Parallel WaveGAN vocoder Yamamoto et al. (2020) for one week on a single RTX 3090 to reconstruct waveform audio.

SAG training data processing. For the singing accompaniment generation (SAG) task, we curate 6,000 hours of Mandarin pop. The pipeline (Figure 2 and 3.3) separates vocal and backing track, captions the audio, extracts time-varying controls—chords, rhythm, vocal melody, structure tags, and local key—which serve as conditioning signals.

MuseControlLite training strategy. Following Tsai et al. (2025), all time-varying conditions are temporally interpolated to a common sequence length and concatenated along the cross-attention feature dimension (not along time). This yields stable conditioning while allowing the partially unfrozen backbone to learn smoother transitions at section boundaries. MuseControlLite is trained using a single NVIDIA RTX 3090 with equivalent batch size of 108 for 9 days.

ComposerFlow inference process. As shown in Figure 1, users provide lyrics with structure tags, plus optional conditions (emotion and key). As described in Section 3.1, the system selects suitable statistical musical attributes from our collection and, together with the structure and lyrics, conditions the melody generator. The resulting melody and lyrics are then passed to the SVS model to synthesize a singing voice. Because the pitch contour of a quantized MIDI melody can deviate from

the realized vocal pitch, we extract the melody from the synthesized singing and use this extracted contour as the vocal melody condition.

In parallel, AccoMontage2 (Yi et al., 2022) performs melody harmonization to produce a chord progression (one chord per bar). With the assembled controls—vocal melody, chords, rhythm (from MIDI timestamps), key (user-specified or inferred from the generated MIDI), and structure—MuseControlLite generates the backing track. Although MuseControlLite generates at most 47 s per pass, we extend duration via the audio-continuation procedure in Section 3.3: we first generate the *chorus* without any audio condition, then generate the *intro*. Subsequent windows condition on the previously generated audio (anchored at the chorus) and proceed segment by segment until the *outro*.

Finally, we mix the completed backing track with the synthesized singing voice by summation and apply peak normalization to avoid clipping.

4.2 Baselines

Recent work has rapidly advanced song generation. For comparison, we select several representative baselines: Suno v4.5 (Suno, 2025), DiffRhythm 1.2-base (Ning et al., 2025), AceStep (Gong et al., 2025), and Levo (Lei et al., 2025). We exclude the following models from evaluation for fairness and comparability: SongGen (Liu et al., 2025b), whose outputs are limited to 30 s; SongBloom (Yang et al., 2025), which requires a 10 s reference audio as a style prompt (incompatible with our reference-free setting); and Jam (Liu et al., 2025a), which leverages phoneme-level timing supervision not available to other systems considered here.

4.3 Evaluation dataset

All baselines and our model are evaluated on a 200-sample test set curated for this study. Each sample is automatically constructed via ChatGPT-5 system prompts that specify both lyrics and text prompts; the lyrics include one verse, one chorus, and one outro. We target 90–120 s *intro–verse–chorus–outro* songs, avoiding full-length pieces to mitigate listener fatigue. To accommodate heterogeneous model inputs, we also provide per-sample metadata (timestamps, timbre, genre, instruments) as separate annotations. For example, some systems (e.g., LeVo (Lei et al., 2025)) prefer tag-style textual descriptions, while others (e.g., DiffRhythm (Ning et al., 2025)) expect lyrics annotated with timestamps. We reformat inputs to match each model's specification while keeping the semantic content consistent across systems.

4.4 Objective metrics

To comprehensively evaluate the generated songs, we employ a set of objective metrics that capture different aspects of alignment and fidelity. These metrics quantify how well the outputs follow the intended textual and lyrical conditions. In addition, to better approximate human preferences, we further adopt automatic evaluation tools designed to reflect human-like judgments of aesthetics and production quality.

Lyrics Alignment: We employed Whisper ASR (Radford et al., 2022) to transcribe the generated vocals and compared the transcriptions with the ground-truth lyrics. We compute alignment quality by analyzing word-level confidence scores and measuring the consistency between the predicted transcription and the reference text.

Style Alignment: We utilized CLAP (Wu et al., 2024b) to compute the cosine similarity between audio embeddings of the generated music and embeddings of the text prompts, quantifying adherence to the intended global style.

Aesthetics Evaluation: We used Audiobox-Aesthetics (Tjandra et al., 2025), to provide an automatic aesthetics qualification of the generated songs, capturing aspects such as clarity, richness, and technical fidelity. Specifically, Audiobox reports four sub-scores: Coherence (CE), Cultural Understanding (CU), Production Complexity (PC), and Production Quality (PQ).

SongEval(Yao et al., 2025) A recently released evaluation tool specifically designed for songs, used to measure structural clarity, memorability, musical coherence, and overall musicality.

Table 4: Objective evaluation for Lyrics Alignment, Style Alignment, and inference speed. The inference speed is tested on RTX 3090.

Model	Lyrics Alignment ↑	CLAP ↑	Inference Time (s / song) ↓
Suno v4.5	0.634	**0.210**	–
Acestep	0.663	0.184	13.0
DiffRhythm	0.624	0.187	11.5
Levo	0.612	0.081	101.2
Ours	**0.702**	0.184	55.5

4.5 Subjective metrics

We conducted a mean opinion score (MOS)-style listening test with 33 participants, including both music experts and non-experts. Each participant listened to 10 generated samples (5 models $\times$ 2 pairs of prompt –lyrics) and rated them on a scale of 1–5 across five dimensions:

Overall Preference: Overall liking of the song.

Lyrics Adherence: Whether the vocal content matches the given lyrics.

Musicality: Perceived musical quality, including melody and harmony.

Voice Naturalness: Naturalness of the generated singing voice.

Song Structure Clarity: Whether the song structure clearly follows the provided lyrical format.

This evaluation design allows us to simultaneously assess text–music alignment, lyric–vocal alignment, and overall musical quality across systems.

4.6 Results

Objective results For lyric–audio alignment, ComposerFlow outperforms all baselines, largely because it leverages an SVS model that yields clear, well-timed vocals. We also observe that when lyrics are short, Suno v4.5 (Suno, 2025) tends to repeat lines, reducing its alignment score. For text–audio alignment, Suno v4.5 leads, while DiffRhythm (Ning et al., 2025), Acestep (Gong et al., 2025), and our system achieve comparable results. On *Audiobox-Aesthetics*, DiffRhythm scores highest overall, with other models clustered closely behind. In contrast, SongEval (Yao et al., 2025) indicates that Suno v4.5 is strongest across perspectives, consistent with the subjective evaluation in Section 4.6, and the SongEval metrics are highly correlated to each other. Our model is slightly weaker than DiffRhythm but surpasses Acestep and Levo (Lei et al., 2025).

Subjective results Consistent with the SongEval objective metrics (Yao et al., 2025), Suno v4.5 (Suno, 2025) outperforms all systems, highlighting a sizable gap between closed- and open-source models. Among open-source baselines, our system ranks above DiffRhythm (Ning et al., 2025) and Levo (Lei et al., 2025) in *Overall Preference*, but trails Acestep (Gong et al., 2025). For *Lyrics Adherence*, our scores are on par with Acestep and exceed other open-source models; this partially diverges from the objective *Alignment* metric. We hypothesize that participants' adherence judgments are influenced by other perceptual factors, even though an SVS front-end should strongly align vocals to the provided lyrics. In *Musicality* and *Song Structure Clarity*, our model surpasses Levo but falls short of the remaining systems. Notably, we obtain the best *Voice Naturalness* among open-source methods, underscoring the benefit of using an SVS module within the song-generation pipeline.

5 Limitations

Although our approach offers a promising alternative for song generation, several limitations remain. First, melodies produced by CSL-L2M (Chai & Wang, 2025) tend to degrade beyond $\sim$120 s, likely

Table 5: Objective evaluation (Audiobox-Aesthetics (Tjandra et al., 2025) and SongEval (Yao et al., 2025); *higher is better*).

Model	Audiobox				SongEval				
	CE	CU	PC	PQ	Coherence	Musicality	Memorability	Clarity	Naturalness
Suno v4.5	7.339	7.766	5.333	8.036	**4.198**	**4.011**	**4.174**	**4.034**	**3.939**
Acestep	7.209	7.642	5.820	7.948	3.449	3.214	3.203	3.216	3.162
DiffRhythm	7.530	**7.791**	**6.336**	8.189	3.740	3.419	3.595	3.512	3.354
Levo	7.565	7.674	4.993	**8.295**	3.392	3.272	3.198	3.265	3.155
Ours	**7.589**	7.715	6.287	8.231	3.648	3.396	3.436	3.434	3.250

Table 6: Subjective evaluation results across models (higher is better).

Model	Overall Preference	Lyrics Adherence	Musicality	Voice Naturalness	Song Structure Clarity
Suno v4.5	**4.091**	**4.076**	**4.091**	**3.909**	**3.970**
ACE-Step	2.803	3.455	3.045	2.561	3.045
DiffRhythm	2.409	2.788	2.758	2.561	2.530
LeVo	2.212	2.045	2.5	2.288	2.394
Ours	2.530	3.348	2.561	2.924	2.409

because most training examples follow an intro–verse–chorus–outro form and are only ~ 90–$120\,\mathrm{s}$ long, complicating full-length generation. Second, our MuseControlLite-based SAG model often yields weaker intros than end-to-end baselines, plausibly because conditioning relies on vocals while intros frequently lack singing. We attempted a reverse-generation strategy (generate the chorus first, then the intro), but there remains room for improvement.

6 Conclusion

In this paper, we proposed ComposerFlow, demonstrating that combining three lightweight modules (CSL-L2M (Chai & Wang, 2025), SVS, and Accomontage2 (Yi et al., 2022)) with a fine-tuned text-to-music model can effectively enable song generation while requiring significantly less training data and computational resources. Moreover, ComposerFlow allows users to freely edit intermediate results—such as melodies and chords in MIDI format—and provides separate vocal and backing tracks for greater flexibility. Importantly, the modular design means that components can be replaced with newer models (e.g., more advanced SVS systems), as long as they satisfy the pipeline's input and output requirements. Thus, ComposerFlow represents **a flexible paradigm rather than a fixed system**. We evaluated ComposerFlow through both objective and subjective experiments and observed comparable results to state-of-the-art open-source systems. These findings suggest that generating high-quality songs does not necessarily require massive datasets, excessive computational power, or high energy consumption.

7 Reproducibility Statement

We will release the full training and inference code for ComposerFlow, including the data-processing pipeline and configuration files. The base models are open sourced; we provide adapted implementations to meet our specific requirements. Implementation details are provided in Section 3.

8 Ethics Statement

This work develops a modular song-generation pipeline intended for research and creative assistance rather than full automation of musical labor. We mitigate risks as follows: (1) **Data and Copyright**: the SVS module is trained solely on licensed recordings from two professional singers; accompaniment modeling relies on time-varying symbolic controls (e.g., chords, rhythm, melody) rather than reproducing copyrighted waveforms, but was still trained to learn the underlying data distribution(2) **Environmental Impact**: training and fine-tuning are conducted on a single NVIDIA RTX 3090 GPU, substantially reducing compute and energy use relative to typical end-to-end sys-

tems. (3) **Responsible Use**: the system does not include identity cloning or voice-matching features and is intended for research, education, and human-in-the-loop creation. (4) **Transparency**: we acknowledge the use of large language models (LLMs) to polish writing and assist with code where appropriate. We will release code and configurations to support reproducibility and community scrutiny.

REFERENCES

Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, et al. Audiolm: a language modeling approach to audio generation. *IEEE/ACM transactions on audio, speech, and language processing*, 31:2523–2533, 2023.

Li Chai and Donglin Wang. Csl-l2m: Controllable song-level lyric-to-melody generation based on conditional transformer with fine-grained lyric and musical controls. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 23541–23549, 2025.

Jianyi Chen, Wei Xue, Xu Tan, Zhen Ye, Qifeng Liu, and Yike Guo. Fastsag: towards fast non-autoregressive singing accompaniment generation. *arXiv preprint arXiv:2405.07682*, 2024a.

Jiawei Chen, Xu Tan, Jian Luan, Tao Qin, and Tie-Yan Liu. Hifisinger: Towards high-fidelity neural singing voice synthesis. *arXiv preprint arXiv:2009.01776*, 2020.

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, 2024b.

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, 2024.

Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438*, 2022.

Chris Donahue, Antoine Caillon, Adam Roberts, Ethan Manilow, Philippe Esling, Andrea Agostinelli, Mauro Verzetti, Ian Simon, Olivier Pietquin, Neil Zeghidour, et al. Singsong: Generating musical accompaniments from singing. *arXiv preprint arXiv:2301.12662*, 2023.

Zach Evans, CJ Carr, Josiah Taylor, Scott H Hawley, and Jordi Pons. Fast timing-conditioned latent audio diffusion. In *Forty-first International Conference on Machine Learning*, 2024a.

Zach Evans, Julian D Parker, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons. Long-form music generation with latent diffusion. *arXiv preprint arXiv:2404.10301*, 2024b.

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.

Seth* Forsgren and Hayk* Martiros. Riffusion - Stable diffusion for real-time music generation. 2022. URL `https://riffusion.com/about`.

Arushi Goel, Sreyan Ghosh, Jaehyeon Kim, Sonal Kumar, Zhifeng Kong, Sang-gil Lee, Chao-Han Huck Yang, Ramani Duraiswami, Dinesh Manocha, Rafael Valle, et al. Audio flamingo 3: Advancing audio intelligence with fully open large audio language models. *arXiv preprint arXiv:2507.08128*, 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.

Siyuan Hou, Shansong Liu, Ruibin Yuan, Wei Xue, Ying Shan, Mangsuo Zhao, and Chao Zhang. Editing music with melody and text: Using controlnet for diffusion transformer. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025.

Zeqian Ju, Peiling Lu, Xu Tan, Rui Wang, Chen Zhang, Songruoyao Wu, Kejun Zhang, Xiangyang Li, Tao Qin, and Tie-Yan Liu. Telemelody: Lyric-to-melody generation with a template-based two-stage method. *arXiv preprint arXiv:2109.09617*, 2021.

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)*, pp. 1–5. IEEE, 2023.

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in neural information processing systems*, 33:17022–17033, 2020.

Shun Lei, Yaoxun Xu, Zhiwei Lin, Huaicheng Zhang, Wei Tan, Hangting Chen, Jianwei Yu, Yixuan Zhang, Chenyu Yang, Haina Zhu, et al. Levo: High-quality song generation with multi-preference alignment. *arXiv preprint arXiv:2506.07520*, 2025.

Haohe Liu, Yi Yuan, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Qiao Tian, Yuping Wang, Wenwu Wang, Yuxuan Wang, and Mark D Plumbley. Audioldm 2: Learning holistic audio generation with self-supervised pretraining. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:2871–2883, 2024.

Jinglin Liu, Chengxi Li, Yi Ren, Feiyang Chen, and Zhou Zhao. Diffsinger: Singing voice synthesis via shallow diffusion mechanism. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 11020–11028, 2022.

Renhang Liu, Chia-Yu Hung, Navonil Majumder, Taylor Gautreaux, Amir Ali Bagherzadeh, Chuan Li, Dorien Herremans, and Soujanya Poria. Jam: A tiny flow-based song generator with fine-grained controllability and aesthetic alignment. *arXiv preprint arXiv:2507.20880*, 2025a.

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*, 2025b.

Peiling Lu, Jie Wu, Jian Luan, Xu Tan, and Li Zhou. Xiaoicesing: A high-quality and integrated singing voice synthesis system. *arXiv preprint arXiv:2006.06261*, 2020.

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.

Jonggwon Park, Kyoyun Choi, Sungwook Jeon, Dokyun Kim, and Jonghun Park. A bi-directional transformer for musical chord recognition. In Arthur Flexer, Geoffroy Peeters, Julián Urbano, and Anja Volk (eds.), *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019*, pp. 620–627, 2019. URL `http://archives.ismir.net/ismir2019/paper/000075.pdf`.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022. URL `https://arxiv.org/abs/2212.04356`.

Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech 2: Fast and high-quality end-to-end text to speech. *arXiv preprint arXiv:2006.04558*, 2020.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Hendrik Schreiber and Meinard Müller. Musical tempo and key estimation using convolutional neural networks with directional filters. In *Proceedings of the Sound and Music Computing Conference (SMC)*, pp. 47–54, Málaga, Spain, 2019.

Zhonghao Sheng, Kaitao Song, Xu Tan, Yi Ren, Wei Ye, Shikun Zhang, and Tao Qin. Songmass: Automatic song writing with pre-training and alignment constraint. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 13798–13805, 2021.

Inc. Suno. Introducing v4.5. https://suno.com/blog/introducing-v4-5, may 2025. Accessed: 2025-09-24.

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, 2025. URL https://arxiv.org/abs/2502.05139.

Quoc-Huy Trinh, Minh-Van Nguyen, Trong-Hieu Nguyen Mau, Khoa Tran, and Thanh Do. Sing-on-your-beat: Simple text-controllable accompaniment generations. *arXiv preprint arXiv:2411.01661*, 2024.

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. *arXiv preprint arXiv:2506.18729*, 2025.

Udio, Inc. Introducing v1.5. https://www.udio.com/blog/introducing-v1-5, 2024. Accessed: 2025-09-20.

Ju-Chiang Wang, Wei-Tsung Lu, and Minz Won. Mel-band roformer for music source separation. *arXiv preprint arXiv:2310.01809*, 2023.

Shih-Lun Wu and Yi-Hsuan Yang. Musemorphose: Full-song and fine-grained piano music style transfer with one transformer vae. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:1953–1967, 2023.

Shih-Lun Wu, Chris Donahue, Shinji Watanabe, and Nicholas J Bryan. Music controlnet: Multiple time-varying controls for music generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:2692–2703, 2024a.

Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Marianna Nezhurina, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation, 2024b. URL https://arxiv.org/abs/2211.06687.

Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6199–6203. IEEE, 2020.

Chenyu Yang, Shuai Wang, Hangting Chen, Wei Tan, Jianwei Yu, and Haizhou Li. Songbloom: Coherent song generation via interleaved autoregressive sketching and diffusion refinement. *arXiv preprint arXiv:2506.07634*, 2025.

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, 2025. URL https://arxiv.org/abs/2505.10793.

Li Yi, Haochen Hu, Jingwei Zhao, and Gus Xia. Accomontage2: A complete harmonization and accompaniment arrangement system. *arXiv preprint arXiv:2209.00353*, 2022.

Yi Yu, Abhishek Srivastava, and Simon Canales. Conditional lstm-gan for melody generation from lyrics. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 17(1):1–20, 2021.

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.

A USE OF LLMS

We use LLM to polish our paper, and parts of the code are collaborated with LLMs.