DON'T DESCRIBE, DEMONSTRATE: LEARNING TRANSFERABLE MUSIC EDITS FROM BEFORE–AFTER EXAMPLES

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing music editing methods mainly rely on text instructions, which are often insufficient for fine-grained or difficult-to-describe transformations. We introduce *editing by demonstration*, where a before–after music pair specifies an edit to be transferred to unseen inputs. This setting poses two key challenges: (1) translating a demonstrated transformation into an executable editing condition, and (2) isolating the transferable edit from reference-specific musical content. To address them, we propose *Edit Condition Synthesis*, which recovers a soft edit condition in the native conditioning space of a frozen flow-based editor. We further introduce a transition-aware semantic constraint that captures source-to-target changes and improves the transferability of the synthesized condition. Experiments on instrument and style editing show improved edit transfer and source preservation over existing text-guided and demonstration-based methods. An online demo with audio examples is available here.

1 INTRODUCTION

Most music editing systems describe an edit with text. This works well for clear and named changes, such as replacing an instrument, changing the genre, or adjusting the mood Li et al. (2026); Yuan et al. (2026); Doh et al. (2026); Lan et al. (2026). Yet many musical edits are much easier to show than to describe. A producer may know exactly how a groove should change, how a texture should move, or how a sound should evolve, while finding it difficult to express the same change in words. In such cases, a before–after example can be a more direct form of instruction Skousen (1989) (As illustrated in Fig. 1). This motivates *editing by demonstration*: instead of telling an editor what to do, we show it how one piece was changed and ask it to apply the same kind of change to another piece Bar et al. (2022).

Once an edit is specified by a demonstration, the central objective is to isolate the transferable change encoded by the example Nguyen et al. (2023). A before–after pair contains both the desired transformation and musical details tied to the reference itself. Directly fitting the demonstrated target can therefore entangle the recovered edit with reference-specific timbre, structure, or instrumentation Lu et al. (2019). The source–target relation provides a more useful signal for identifying *what changed* in the example. A successful demonstration-based editor should capture this change and reproduce it on new music while allowing unrelated content to remain input-dependent.

Recent demonstration-based methods have shown that a before–after pair can provide a useful specification of an edit, either by optimizing a continuous condition or by modeling the change between paired examples Nguyen et al. (2023); Srivastava et al. (2025). However, music editing presents distinct challenges. **(1) Executable edit conditioning.** Music editors are largely controlled through textual conditions Evans et al. (2025), whereas an audio demonstration only shows the desired outcome. The key is to turn the observed change into a condition that can guide the editing process without relying on an incomplete text description. **(2) Transferable edit representation.** A music target contains not only the intended change but also reference-specific attributes, such as timbre, instrumentation, rhythm, and structure. The inferred condition should capture the transformation that remains consistent across different inputs while avoiding attributes specific to the demonstrated tar-

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

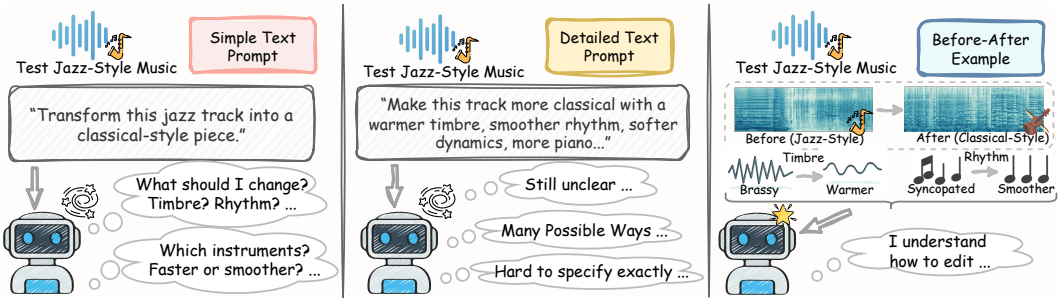


Figure 1: Illustration of music editing by demonstration. A source–target exemplar directly specifies the desired transformation, offering a more intuitive and expressive instruction.

get. We therefore seek an edit representation that is both *executable* and *transferable* across different music inputs.

To address these challenges, we propose Edit Condition Synthesis, which synthesizes a soft edit condition directly in the native conditioning space of a frozen flow-based music editor. Given a before–after demonstration, we construct paired latent flow states with shared noise, using their relative velocity as a source-to-target transition signal. We optimize a continuous condition to reproduce this transition, yielding an executable edit representation without retraining the generator. To improve transferability, we further adapt MuQ to capture source-to-target semantic changes and use its transition representation as an auxiliary constraint during condition synthesis. Together, these signals encourage the recovered condition to capture the demonstrated edit rather than reference-specific content. To support systematic evaluation, we construct instrument- and style-editing benchmarks from existing music datasets. Experiments on both benchmarks demonstrate effective edit transfer and source preservation.

Our main contributions are as follows:

- We introduce *editing by demonstration*, a new music editing setting that transfers the transformation specified by a before–after example to unseen music.
- We propose Edit Condition Synthesis, which extracts the demonstrated transition from paired flow dynamics and optimizes an executable soft condition in the native conditioning space of a frozen music editor.
- We introduce a transition-aware MuQ representation that captures source-to-target semantic changes and provides an auxiliary constraint for learning transferable, rather than reference-specific, edits.
- We construct paired instrument- and style-editing benchmarks from existing music datasets and conduct extensive experiments, demonstrating effective edit transfer and source preservation across diverse transformations.

2 Related Work

2.1 Generative Models for Audio and Music Editing

Recent advances in generative audio models have established strong priors for audio synthesis and manipulation. Diffusion- and flow-matching-based models have achieved substantial progress in text-to-audio and text-to-music generation Agostinelli (2023); Copet et al. (2023); Huang et al. (2023); Liu et al. (2024); Lan et al. (2024), providing the foundation for controllable audio and music editing. Existing methods can be broadly divided into training-based and training-free approaches. Training-based methods learn editing capabilities from paired source–instruction–target data through instruction tuning or lightweight adaptation Wang et al. (2023a); Han et al. (2023); Zhang et al. (2024a); Lin et al. (2024); Yang et al. (2025), whereas training-free methods manipulate pretrained generative processes through inversion, latent or attention control, and source–target trajectory modification Meng et al. (2022); Mokady et al. (2023); Zhang et al. (2024b); Manor & Michaeli (2024). More recent inversion-free methods further construct direct transitions between source and target states for improved efficiency and fidelity Kulikov et al. (2025); Lei et al. (2026).

Despite these different strategies, most existing methods still rely on textual conditions to specify the desired edit. While intuitive, text may be insufficient for transformations that are difficult to describe precisely. Example-based editing offers an alternative by specifying the desired change through paired demonstrations, allowing richer and more implicit editing semantics to be conveyed directly through examples.

2.2 Example-Based and Demonstration-Guided Editing

Beyond textual instructions, reference examples provide an intuitive way to specify properties that are difficult to describe precisely. Early reference-guided methods mainly transfer attributes such as style or appearance from an exemplar to a new input Huang & Belongie (2017). More recent studies use source–target pairs to specify the transformation itself, allowing edits demonstrated by before–after examples to be transferred to unseen inputs Nguyen et al. (2023); Xu et al. (2025); Srivastava et al. (2025). Other works further learn representations of the semantic change between paired examples, shifting the focus from individual content toward the transformation connecting them Wang et al. (2025); Chen et al. (2025). A closely related direction derives new controls directly in the conditioning space of pretrained generators. Continuous embeddings can be optimized for concepts, reconstruction, or editing while keeping the underlying generator fixed Gal et al. (2023); Kawar et al. (2023); Dong et al. (2023), and recent demonstration-based methods extend this principle to recover editing conditions from example pairs Nguyen et al. (2023); Xu et al. (2025).

These studies establish condition from paired examples as a practical paradigm for demonstration-guided editing. However, fitting the demonstrated target alone does not necessarily isolate the transferable transformation, motivating additional directional or representation-level constraints Nguyen et al. (2023). This challenge is particularly pronounced in music, where an exemplar pair jointly encodes timbre, instrumentation, rhythm, and structure. Existing example-guided audio methods mainly address restricted transformations such as acoustic texture manipulation Cheng et al. (2025), while general music editing from before–after demonstrations remains underexplored. We study this broader setting by learning transformations that can transfer across different musical content.

2.3 Pretrained Music Representation Learning

Large-scale pretraining has established strong general-purpose representations for audio and music. Cross-modal approaches such as CLAP Elizalde et al. (2023) and MuLan Huang et al. (2022) align audio with natural language, enabling semantic understanding of genre, mood, instrumentation, and other high-level attributes. Recent audio–language models further improve compositional and fine-grained music understanding Ghosh et al. (2024); Singh et al. (2026); Ghosh et al. (2026). In parallel, self-supervised models such as MERT Li et al. (2024), MusicFM Won et al. (2024), and MuQ Zhu et al. (2025) learn directly from large-scale music collections, capturing acoustic, rhythmic, harmonic, and structural information without paired text supervision.

Among these models, MuQ learns transferable music representations through self-supervised pretraining with multi-scale discrete targets Zhu et al. (2025). MuQ-MuLan further introduces music–language alignment to enrich high-level semantic understanding. These models provide a strong basis for representing musical content, while our work further explores their ability to capture transformations between paired examples.

3 Preliminaries

Example-Based Generative Music Editing. Generative music editing transforms a source input x^s from its source condition c^s toward a target condition c^t while preserving content unrelated to the desired transformation. Let $\mathcal{G}_\theta$ denote a conditional generative editor with parameters θ. The editing process can be generally written as:

$$\hat{x}^t = \mathcal{G}_\theta\left(x^s; c^s \to c^t\right), \tag{1}$$

where $\hat{x}^t$ denotes the edited output. Unlike text-guided editing, example-based music editing specifies the desired transformation through a before–after demonstration (x_d^s, x_d^t), where x_d^s and x_d^t denote the demonstration source and target, respectively. The goal is not to reproduce the demonstrated target, but to infer an edit condition that captures the underlying transformation and can be

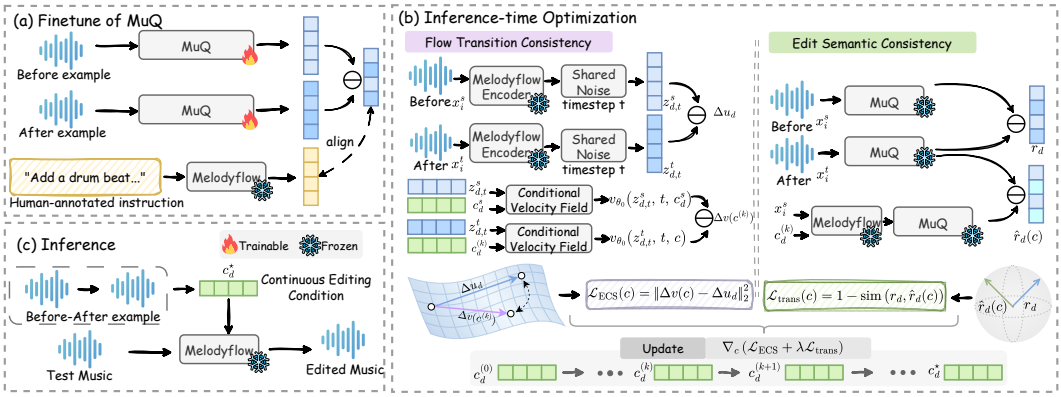


Figure 2: Overview of the proposed framework. Our method consists of three stages: (a) pre-training a MuQ-based edit-transition encoder, (b) optimizing a continuous editing condition from a source–target demonstration, and (c) transferring the optimized condition to inference music of the same edit type.

transferred to a new source input. We denote the inferred condition by $c_d^\star$:

$$c_d^\star = \arg\min_c \mathcal{J}\left(c; x_d^s, x_d^t\right). \tag{2}$$

where $\mathcal{J}$ measures how well a candidate condition c represents the demonstrated transformation. The inferred condition is then applied to a new source x^s as:

$$\hat{x}^t = \mathcal{G}_\theta\left(x^s; c^s \to c_d^\star\right). \tag{3}$$

A useful edit condition should therefore be both *executable* by the underlying editor and *transferable* across music with different content. In the following section, we instantiate $\mathcal{J}$ using the demonstrated flow transition together with a transition-level semantic constraint.

Conditional Flow Matching. Flow matching learns a continuous vector field that transports samples from a simple noise distribution to the data distribution Lipman et al. (2023); Liu et al. (2023). Let z_1 denote the latent representation of a data sample and $z_0 \sim \mathcal{N}(0, I)$ denote Gaussian noise. A linear flow path is defined as $z_t = tz_1 + (1-t)z_0$, where $t \in [0, 1]$ with target velocity $u_t = z_1 - z_0$. Given a continuous condition c, a conditional velocity model v_θ is trained with:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{z_1, z_0, t}\left[\left\|v_\theta(z_t, t, c) - u_t\right\|_2^2\right]. \tag{4}$$

For flow-based editing, a condition change can be characterized by the difference between source- and target-conditioned velocity fields:

$$\Delta v_{\theta_0}(t) = v_{\theta_0}(z_t^t, t, c^t) - v_{\theta_0}(z_t^s, t, c^s), \tag{5}$$

where z_t^s and z_t^t denote the corresponding source- and target-side flow states.

4 Method

4.1 Analysis and Formulation

Given a before–after demonstration $\mathcal{D} = (x_d^s, x_d^t)$ and a music input x^s to be edited, our goal is to infer a reusable edit condition $c_d^\star$ from $\mathcal{D}$ and transfer the demonstrated transformation to x^s. We denote the frozen pretrained flow-based music editor by $\mathcal{G}_{\theta_0}$. Let c^s denote the source condition associated with the input music. The edited result is obtained as $\hat{x}^t = \mathcal{G}_{\theta_0}(x^s, c^s, c_d^\star)$. An overview of the proposed framework is shown in Figure 2.

A common way to identify a continuous condition is to optimize it toward the demonstrated target. Given the demonstration source condition c_d^s and a discrepancy measure $d(\cdot, \cdot)$, such endpoint fitting can be written as:

$$\mathcal{L}_{\text{end}}(c) = d\left(\mathcal{G}_{\theta_0}(x_d^s, c_d^s, c), x_d^t\right). \tag{6}$$

The endpoint represents only one realization of the desired transformation. Different conditions may therefore explain the same demonstration while inducing different edits on a new input:

$$\mathcal{G}_{\theta_0}(x_d^s, c_d^s, c_i) \approx \mathcal{G}_{\theta_0}(x_d^s, c_d^s, c_j) \approx x_d^t, \quad \mathcal{G}_{\theta_0}(x^s, c^s, c_i) \not\approx \mathcal{G}_{\theta_0}(x^s, c^s, c_j). \tag{7}$$

The paired structure of the demonstration provides a richer signal. Let $v_{\theta_0}(z, t, c)$ denote the conditional velocity field of the pretrained flow model at latent state z, flow time t, and condition c. For paired source and target states z_t^s and z_t^t, we define their relative conditional dynamics as:

$$\Delta v_{\theta_0}(z_t^s, z_t^t; c^s, c^t) = v_{\theta_0}(z_t^t, t, c^t) - v_{\theta_0}(z_t^s, t, c^s). \tag{8}$$

This quantity characterizes the change in generative dynamics from the source to the target condition and forms the basis of direct inversion-free flow editing. We use the observed before–after pair to provide an explicit reference for this relative dynamics.

4.2 Edit Condition Synthesis

Let $\mathcal{E}$ denote the audio encoder associated with the pretrained editor. We encode the demonstrated source and target as $z_d^s = \mathcal{E}(x_d^s)$ and $z_d^t = \mathcal{E}(x_d^t)$, respectively. We consider the affine interpolation used by a broad class of flow-matching models:

$$z_t = \alpha_t z + \sigma_t \epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \tag{9}$$

where z denotes a data latent, and α_t and σ_t are the data and noise schedules, respectively. The corresponding target velocity is $u_t(z, \epsilon) = \dot{\alpha}_t z + \dot{\sigma}_t \epsilon$, where $\dot{\alpha}_t$ and $\dot{\sigma}_t$ denote derivatives with respect to t. For the demonstrated source and target, we construct two native flow states using the same noise sample:

$$z_{d,t}^s = \alpha_t z_d^s + \sigma_t \epsilon, \quad z_{d,t}^t = \alpha_t z_d^t + \sigma_t \epsilon. \tag{10}$$

The shared noise gives the following relative target velocity:

$$\Delta u_d(t) = u_t(z_d^t, \epsilon) - u_t(z_d^s, \epsilon) = \left(\dot{\alpha}_t z_d^t + \dot{\sigma}_t \epsilon\right) - \left(\dot{\alpha}_t z_d^s + \dot{\sigma}_t \epsilon\right) = \dot{\alpha}_t \left(z_d^t - z_d^s\right). \tag{11}$$

Eq. 11 provides the transition signal used for condition synthesis. The noise term cancels exactly, leaving a deterministic source–target direction defined by the demonstration, while the shared noise determines where the pretrained velocity field is evaluated.

Given the demonstration source condition c_d^s and a candidate edit condition c, we define the condition-induced relative dynamics as:

$$\Delta v_{\theta_0}^d(t, \epsilon; c) = v_{\theta_0}(z_{d,t}^t, t, c) - v_{\theta_0}(z_{d,t}^s, t, c_d^s). \tag{12}$$

We align these dynamics with the transition observed in the demonstration:

$$\mathcal{L}_{\mathrm{ECS}}(c) = \mathbb{E}_{t,\epsilon}\left[\left\|\Delta v_{\theta_0}^d(t, \epsilon; c) - \dot{\alpha}_t \left(z_d^t - z_d^s\right)\right\|_2^2\right]. \tag{13}$$

The flow-only condition is obtained as:

$$c_d^{\mathrm{flow}} = \arg\min_c \mathcal{L}_{\mathrm{ECS}}(c). \tag{14}$$

Only the continuous condition c is optimized, while all pretrained components remain frozen. The resulting condition therefore remains in the native conditioning space of the editor and can be directly reused for new music inputs. For rectified flow, $\alpha_t = t$ and $\sigma_t = 1 - t$, such that $\dot{\alpha}_t = 1$. Eq. 11 then reduces to $\Delta u_d = z_d^t - z_d^s$.

4.3 Learning Edit Transition Representations

Pretrained music encoders are primarily designed to represent the attributes of individual music clips, whereas our task requires a representation that is sensitive to the change between two clips. We build on a pretrained music representation model F_{ϕ_0} and adapt it to obtain a transition encoder Φ_ϕ, where ϕ_0 and ϕ denote the pretrained and adapted parameters, respectively. In our implementation, we use MuQ Zhu et al. (2025) as the underlying music encoder.

Given a source–target pair (x_i^s, x_i^t), its edit representation is defined as $r_i = \Phi_\phi(x_i^s, x_i^t)$. We train the transition encoder such that pairs sharing the same editing transformation remain close despite differences in musical content. For a minibatch of B transition pairs, let $\mathcal{P}(i)$ denote the set of pairs sharing the same edit as pair i. We use the following contrastive objective:

$$\mathcal{L}_{\text{rep}} = -\frac{1}{B}\sum_{i=1}^{B}\frac{1}{|\mathcal{P}(i)|}\sum_{p\in\mathcal{P}(i)}\log\frac{\exp\left(\text{sim}(r_i, r_p)/\tau_c\right)}{\sum_{j\neq i}\exp\left(\text{sim}(r_i, r_j)/\tau_c\right)}, \tag{15}$$

where $\text{sim}(\cdot,\cdot)$ denotes cosine similarity and τ_c is the contrastive temperature. Details of the adaptation procedure are provided in Appendix A.

The adapted representation is then used as an auxiliary constraint during condition synthesis. The demonstrated transition is represented as $r_d = \Phi_\phi(x_d^s, x_d^t)$. For a candidate condition c, the frozen editor produces $\hat{x}_d^t(c) = \mathcal{G}_{\theta_0}(x_d^s, c_d^s, c)$, whose induced transition is represented as $\hat{r}_d(c) = \Phi_\phi(x_d^s, \hat{x}_d^t(c))$. We define the transition consistency loss as:

$$\mathcal{L}_{\text{trans}}(c) = 1 - \text{sim}\left(r_d, \hat{r}_d(c)\right) = 1 - \frac{r_d^\top \hat{r}_d(c)}{|r_d|_2|\hat{r}_d(c)|_2}. \tag{16}$$

This term favors conditions whose induced edits match the demonstrated transformation in the learned transition space. Given a tolerance η, we define the executable condition set as:

$$\mathcal{C}_{\text{flow}}(\eta) = \left\{c \mid \mathcal{L}_{\text{ECS}}(c) \leq \eta\right\}, \tag{17}$$

where η controls the allowable flow-transition discrepancy. Conceptually, we seek a condition that minimizes $\mathcal{L}_{\text{trans}}$ within $\mathcal{C}_{\text{flow}}(\eta)$. In practice, we optimize the following joint objective:

$$c_d^\star = \arg\min_c \left[\mathcal{L}_{\text{ECS}}(c) + \lambda\mathcal{L}_{\text{trans}}(c)\right], \tag{18}$$

where λ controls the strength of transition-level guidance. The flow objective aligns the condition with the demonstrated change, while the transition representation further constrains $c_d^\star$ toward the transferable edit rather than attributes specific to the demonstrated target.

4.4 Training and Optimization Pipeline

Our framework consists of two offline adaptation stages followed by test-time condition optimization. First, a frozen pretrained music encoder is equipped with two Transformer layers and an MLP projector, which are trained to align audio representations with the editor's text-encoder space and provide an initial continuous condition $c_d^{(0)}$. Separately, another music encoder is adapted on paired music edits to obtain the transition representation model Φ_ϕ used by $\mathcal{L}_{\text{trans}}$. At inference time, all networks remain frozen. Given a demonstration pair (x_d^s, x_d^t), we initialize the soft condition with $c_d^{(0)}$ and optimize only the condition using Eq. 18. In our implementation, both music encoders are based on MuQ Zhu et al. (2025), and MelodyFlow Lan et al. (2024) serves as the pretrained editor. Additional details are provided in Appendix C.

5 Experiments

5.1 Experiment Setup

Dataset Setting. Existing music datasets rarely provide aligned source–edited audio pairs with explicit editing operations. We therefore introduce a paired dataset for exemplar-based music editing, consisting of an *instrument editing* subset derived from Slakh2100-redux Manilow et al. (2019) and a *style editing* subset based on MusicCaps and MeLBench Agostinelli (2023); Chowdhury et al. (2024). The former covers instrument addition, removal, and replacement, while the latter contains transformations among seven representative music styles. The two subsets contain 7,500 and 8,400 samples, respectively. Each sample consists of a source audio x_{src}, an edited audio x_{edit}, and its editing type. We additionally include approximately 10% no-op samples and split the data by musical piece or source audio to avoid overlap across training, validation, and test sets. More details are provided in Appendix D.

Table 1: Quantitative comparison on the instrument editing and style editing subsets.

Method	Instrument Editing						
	Target Consistency				Source Preservation		Edit Alignment
	FAD ↓	FD ↓	KL ↓	CLAP ↑	F_0-PCC ↑	LPAPS ↓	EIA ↑
FlowEdit	4.31±0.24	29.50±1.36	1.06±0.05	0.42±0.02	0.34±0.02	6.16±0.10	0.25±0.02
ChordEdit	3.94±0.21	22.56±1.12	0.79±0.04	0.42±0.02	0.37±0.02	4.81±0.08	0.27±0.02
AP-Adapter	9.80±0.52	22.71±1.08	0.54±0.03	0.25±0.02	0.38±0.02	5.56±0.09	0.23±0.02
MuseControlLite	6.91±0.39	33.50±1.62	0.99±0.05	0.39±0.02	0.30±0.02	7.88±0.12	0.23±0.02
ReNoise	4.91±0.28	17.95±0.91	1.28±0.06	0.52±0.02	0.30±0.02	6.96±0.11	0.29±0.02
VISII	2.22±0.15	11.98±0.62	1.06±0.05	0.48±0.02	0.32±0.02	7.49±0.12	0.16±0.01
EditCLIP	2.57±0.17	10.65±0.55	0.63±0.03	0.51±0.02	0.47±0.02	3.54±0.06	0.28±0.02
EBAA	6.92±0.38	23.70±1.16	0.86±0.04	0.44±0.02	0.51±0.03	4.86±0.08	0.20±0.02
Ours	**1.60**±0.12	**6.48**±0.34	**0.48**±0.03	**0.57**±0.02	**0.60**±0.03	**2.51**±0.05	**0.31**±0.02
	Style Editing						
FlowEdit	6.00±0.33	31.76±1.54	1.01±0.05	0.27±0.02	0.26±0.02	7.04±0.11	0.20±0.02
ChordEdit	5.97±0.31	36.23±1.76	0.95±0.05	0.35±0.02	0.28±0.02	6.40±0.10	0.18±0.02
AP-Adapter	15.86±0.79	22.50±1.12	0.91±0.05	0.37±0.02	0.25±0.02	8.08±0.13	0.24±0.02
MuseControlLite	6.86±0.37	29.63±1.48	0.94±0.05	0.21±0.01	0.27±0.02	7.83±0.12	0.20±0.02
ReNoise	4.02±0.24	18.99±0.95	0.93±0.04	0.38±0.02	0.29±0.02	6.86±0.11	0.30±0.02
VISII	4.05±0.25	19.26±0.97	0.90±0.04	0.38±0.02	0.24±0.02	7.64±0.12	0.27±0.02
EditCLIP	7.33±0.40	23.67±1.19	0.96±0.05	0.31±0.02	0.37±0.02	5.75±0.09	0.25±0.02
EBAA	12.66±0.67	38.20±1.88	1.13±0.06	0.35±0.02	0.38±0.02	8.13±0.13	0.21±0.02
Ours	**3.46**±0.22	**18.84**±0.93	**0.88**±0.04	**0.42**±0.02	**0.45**±0.02	**4.05**±0.07	**0.34**±0.02

Evaluation Metrics. We adopt FAD Kilgour et al. (2018), FD Kong et al. (2020), Kullback–Leibler (KL) divergence Kullback & Leibler (1951), CLAP Elizalde et al. (2023), F_0-PCC Wang et al. (2023b), and LPAPS Iashin & Rahtu (2021) to evaluate generation quality, target consistency, and source preservation. FAD and FD measure feature-distribution discrepancy; KL compares semantic class distributions; CLAP assesses consistency with the target text; F_0-PCC measures melodic and pitch preservation; and LPAPS captures perceptual differences between source and edited audio. We further propose Edit Instruction Alignment (EIA), following transition-aware evaluation Chen et al. (2025), to measure whether the source-to-edited transformation matches the intended edit. Higher EIA indicates better edit alignment. Further details are provided in Appendix E.1.

Training Configuration. All experiments are implemented in PyTorch with BF16 mixed precision. The transition encoder is trained with AdamW for up to 50 epochs using a learning rate of 1×10^{-4}, a weight decay of 0.01, and a batch size of 64. The audio-to-condition mapper is trained separately for 2,000 steps with AdamW and the same learning rate. At inference time, all model parameters are frozen, and only the continuous editing condition is optimized. Further training and implementation details are provided in Appendix C.

5.2 Baseline Models

We compare our method with two groups of baselines under a unified evaluation protocol: text-guided and exemplar-based music editing methods. For text-guided editing, we include FlowEdit Kulikov et al. (2025) and ChordEdit Lu et al. (2026) as flow-based editing approaches, AP-Adapter Tsai et al. (2024) and MuseControlLite Tsai et al. (2025) as lightweight conditional methods, and ReNoise Garibi et al. (2024), adapted to flow matching with MelodyFlow Lan et al. (2024), as an inversion-based baseline.

For exemplar-based editing, directly applicable music-editing methods remain limited. We therefore adapt representative demonstration-based approaches, including VISII Nguyen et al. (2023), which derives a transferable continuous instruction from an exemplar pair, and EditCLIP Wang et al. (2025), which learns an edit representation from source–target examples. Their transformation-

Table 2: Ablation study on the transformation alignment loss $\mathcal{L}_{\text{trans}}$.

Method	FAD ↓	CLAP ↑	F_0-PCC ↑	EIA ↑
w/o $\mathcal{L}_{\text{trans}}$	3.53	0.41	0.38	0.25
Full	**2.53**	**0.50**	**0.53**	**0.33**

Table 3: Ablation study on the optimization of continuous editing condition C_t.

Method	FAD ↓	CLAP ↑	F_0-PCC ↑	EIA ↑
w/o C_t Op.	6.50	0.42	0.42	0.23
Full	**2.53**	**0.50**	**0.53**	**0.33**

Table 4: Comparison of different editing conditions on audio-editing samples.

Condition	FAD ↓	CLAP ↑	F_0-PCC ↑	EIA ↑
No text	3.02	0.47	0.42	0.31
Simple text	3.06	0.51	0.30	0.31
Detailed text	2.56	**0.54**	0.29	0.33
Ours	**2.53**	0.50	**0.53**	**0.33**

Table 5: Sensitivity analysis of the weighting coefficient λ for $\mathcal{L}_{\text{trans}}$.

Objective	FAD ↓	CLAP ↑	F_0-PCC ↑	EIA ↑
$\lambda = 0.01$	2.53	0.49	0.53	0.31
$\lambda = 0.05$	2.53	**0.49**	**0.57**	**0.33**
$\lambda = 0.2$	2.70	0.47	0.56	0.33
$\lambda = 0.3$	**2.10**	0.40	0.50	0.31

inference mechanisms are adapted to the music domain using the same pretrained editor as our method. We additionally include Exemplar-Based Audio Analogy (EBAA) Cheng et al. (2025), which transfers transformations directly from paired audio exemplars.

5.3 Overall Performance

Table 1 reports the results on instrument and style editing, with standard deviations computed over three runs. Our method achieves strong and balanced performance across target consistency, source preservation, and edit alignment, with clear advantages in preserving source content while transferring the intended transformation. This suggests that the synthesized continuous condition effectively captures the demonstrated edit without introducing unnecessary changes. Existing exemplar-based approaches do not consistently outperform text-guided methods, indicating that access to an editing example alone is insufficient for reliable transfer. In contrast, our method converts the demonstrated change into an executable condition in the native conditioning space and further regularizes it with transition-level guidance, leading to consistent performance across both editing subsets.

5.4 Ablation and Sensitivity Analysis

We evaluate the contribution of the main design choices through ablation and sensitivity studies. Unless otherwise specified, all model, training, and inference settings are kept unchanged.

Transition Guidance. We remove $\mathcal{L}_{\text{trans}}$ while keeping the remaining pipeline unchanged to assess its role in condition synthesis. As shown in Table 2, removing transition guidance consistently degrades target consistency, source preservation, and edit alignment. This suggests that $\mathcal{L}_{\text{trans}}$ complements the flow objective by guiding the synthesized condition toward the demonstrated transformation while reducing reference-specific changes.

Condition Optimization. We evaluate the role of condition optimization by using the initial condition directly, without test-time refinement. As shown in Table 3, this variant performs worse across all metrics, indicating that the initial condition alone is insufficient to capture the demonstrated transformation. Optimizing the condition further improves its alignment with the desired edit.

Continuous vs. Textual Conditions. We compare four conditioning settings: no text, simple text, detailed text, and our continuous condition. The no-text setting omits textual guidance, while brief and detailed text describe the demonstrated transformation at different levels of detail. As shown in Table 4, our method achieves competitive performance, suggesting that continuous conditioning better preserves fine-grained edit information. Detailed text achieves the highest CLAP score, potentially benefiting from its richer textual cues. See Appendix D.2 for details.

Sensitivity to λ. The weighting coefficient of $\mathcal{L}_{\text{trans}}$ shows mild sensitivity over the tested range. As shown in Table 5, no clear monotonic trend is observed across metrics. Intermediate weights provide a better balance across objectives, while smaller or larger values mainly cause metric-specific fluctuations.

Table 6: Ablation study on MuQ adaptation for transition guidance.

Adaptation	FAD ↓	CLAP ↑	F_0-PCC ↑	EIA ↑
Frozen	2.61	0.45	0.52	0.31
Partial	2.67	0.46	0.51	0.31
Full	**2.53**	**0.50**	**0.53**	**0.33**

Table 7: Subjective evaluation on instrument and style editing. I and S denote instrument and style.

Method	I-ET ↑	I-SP ↑	I-AQ ↑	S-ET ↑	S-SP ↑	S-AQ ↑
VISII	2.85	2.30	3.90	2.66	2.34	3.16
EditCLIP	2.55	2.70	3.50	2.45	2.30	2.95
Ours	**2.95**	**4.05**	**4.00**	**2.75**	**2.70**	**3.30**

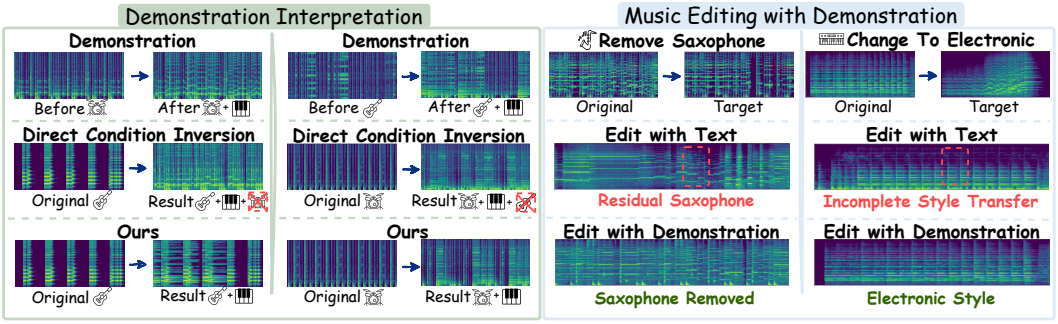


Figure 3: Qualitative analysis of music editing. Left: demonstration interpretation under direct inversion and our method. Right: representative results.

MuQ Adaptation. We evaluate whether transition-specific adaptation of MuQ is necessary for extracting editing guidance from exemplar pairs (Table 6). The frozen variant directly uses the difference between the pretrained MuQ embeddings, while the partially adapted variant aligns the global embedding difference through an MLP. The full model further adapts MuQ to better capture edit-related changes shared across different musical contents.

Direct Condition Inversion. We compare our method with direct condition inversion using the same before–after demonstration in Fig. 3. This variant directly optimizes a continuous condition to fit the observed example, without explicitly modeling the source–target transition. While it can explain the demonstration, the learned condition may also absorb reference-specific attributes and transfer them to new inputs. In contrast, our method explicitly constrains the demonstrated transition, encouraging the condition to capture a more transferable edit. See Appendix D.3 for details.

5.5 Qualitative Evaluation

User Study. We conduct a subjective evaluation with 16 participants, who rate 30 examples on a five-point scale for *Edit Transfer*, *Source Preservation*, and *Audio Quality*. Each example includes a demonstration pair, a source audio, and edited outputs from different methods. As shown in Table 7, our method achieves higher edit transfer and source preservation scores than EditCLIP and VISII across both instrument and style editing, while maintaining competitive audio quality. Further results and evaluation details are provided in the Appendix E.

Visual Case. The right half of Fig. 3 shows examples of transferring demonstrated edits to new inputs. The results illustrate that demonstrations can convey fine-grained musical changes that are difficult to describe through text alone. More qualitative results are provided in the appendix G.

6 Conclusion

We study *editing by demonstration* for music, where a before–after example specifies an edit to be transferred to unseen inputs. To make such demonstrations executable and transferable, we propose Edit Condition Synthesis, which uses paired flow dynamics to optimize a soft condition in the native conditioning space of a frozen music editor. We further introduce a transition-aware semantic constraint based on MuQ to help the condition capture the demonstrated edit rather than reference-specific content. To support systematic evaluation, we construct instrument- and style-editing benchmarks from existing music datasets. Experiments on both benchmarks demonstrate effective edit transfer and source preservation across diverse musical transformations. Future work will explore multimodal demonstrations for more flexible music editing.

AI USE STATEMENT

In this work, we used generative AI tools to clean and reformat the dataset by generating and standardizing textual descriptions associated with the dataset. We have not used generative AI tools for other tasks requiring disclosure, and the remaining required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools to polish the writing and improve the readability of the manuscript. We have reviewed all AI-assisted work. In particular, AI-generated or modified textual descriptions were manually checked for consistency with the corresponding data and intended edit semantics, and all AI-assisted manuscript revisions were reviewed by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We are not aware of any significant ethical concerns associated with this work. The subjective evaluation involved voluntary participation and did not collect personally identifiable or sensitive information.

REFERENCES

Andrea Agostinelli. Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325*, 2023.

Amir Bar, Yossi Gandelsman, Trevor Darrell, Amir Globerson, and Alexei Efros. Visual prompting via image inpainting. *Advances in neural information processing systems*, 35:25005–25017, 2022.

Sherry X Chen, Misha Sra, and Pradeep Sen. Instruct-clip: Improving instruction-guided image editing with automated data refinement using contrastive learning. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28513–28522. IEEE, 2025.

Kan Jen Cheng, Tingle Li, and Gopala Anumanchipalli. Audio texture manipulation by exemplar-based analogy. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025.

Sanjoy Chowdhury, Sayan Nag, KJ Joseph, Balaji Vasan Srinivasan, and Dinesh Manocha. Melfusion: Synthesizing music from image and language cues using diffusion models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26816–26825. IEEE, 2024.

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

Seungheon Doh, Junghyun Tony Koo, Marco Martínez-Ramírez, Woosung Choi, WeiHsiang Liao, Qiyu Wu, Juhan Nam, and Yuki Mitsufuji. Llm2fx-tools: Tool calling for music post-production. In *International Conference on Learning Representations*, volume 2026, pp. 100979–100999, 2026.

Wenkai Dong, Song Xue, Xiaoyue Duan, and Shumin Han. Prompt tuning inversion for text-driven image editing using diffusion models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7396–7406. IEEE, 2023.

Benjamin Elizalde, Soham Deshmukh, Mahmoud Al Ismail, and Huaming Wang. Clap learning audio concepts from natural language supervision. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

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.

Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit Haim Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. In *The Eleventh International Conference on Learning Representations*, 2023.

Daniel Garibi, Or Patashnik, Andrey Voynov, Hadar Averbuch-Elor, and Daniel Cohen-Or. Renoise: Real image inversion through iterative noising. In *European Conference on Computer Vision*, pp. 395–413. Springer, 2024.

Sreyan Ghosh, Ashish Seth, Sonal Kumar, Utkarsh Tyagi, Chandra Kiran Reddy Evuru, Ramaneswaran S, S Sakshi, Oriol Nieto, Ramani Duraiswami, and Dinesh Manocha. Compa: Addressing the gap in compositional reasoning in audio-language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=86NGO8qeWs.

Sreyan Ghosh, Arushi Goel, Lasha Koroshinadze, Sang-gil Lee, Zhifeng Kong, Joao Santos, Ramani Duraiswami, Dinesh Manocha, Wei Ping, Mohammad Shoeybi, et al. Music flamingo: Scaling music understanding in audio language models. In *International Conference on Learning Representations*, volume 2026, pp. 78958–78996, 2026.

Bing Han, Junyu Dai, Weituo Hao, Xinyan He, Dong Guo, Jitong Chen, Yuxuan Wang, Yanmin Qian, and Xuchen Song. Instructme: An instruction guided music edit and remix framework with latent diffusion models. *arXiv preprint arXiv:2308.14360*, 2023.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

Qingqing Huang, Aren Jansen, Joonseok Lee, Ravi Ganti, Judith Yue Li, and Daniel P. W. Ellis. Mulan: A joint embedding of music audio and natural language. In Preeti Rao, Hema A. Murthy, Ajay Srinivasamurthy, Rachel M. Bittner, Rafael Caro Repetto, Masataka Goto, Xavier Serra, and Marius Miron (eds.), *Proceedings of the 23rd International Society for Music Information Retrieval Conference, ISMIR 2022, Bengaluru, India, December 4-8, 2022*, pp. 559–566, 2022.

Qingqing Huang, Daniel S Park, Tao Wang, Timo I Denk, Andy Ly, Nanxin Chen, Zhengdong Zhang, Zhishuai Zhang, Jiahui Yu, Christian Frank, et al. Noise2music: Text-conditioned music generation with diffusion models. *arXiv preprint arXiv:2302.03917*, 2023.

Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *2017 IEEE international conference on computer vision (ICCV)*, pp. 1510–1519. IEEE, 2017.

Vladimir Iashin and Esa Rahtu. Taming visually guided sound generation. *arXiv preprint arXiv:2110.08791*, 2021.

Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6007–6017. IEEE, 2023.

Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fréchet Audio Distance: A metric for evaluating music enhancement algorithms. *arXiv preprint arXiv:1812.08466*, 2018.

Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D Plumbley. Panns: Large-scale pretrained audio neural networks for audio pattern recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2880–2894, 2020.

Vladimir Kulikov, Matan Kleiner, Inbar Huberman-Spiegelglas, and Tomer Michaeli. Flowedit: Inversion-free text-based editing using pre-trained flow models. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19721–19730. IEEE, 2025.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Gael Le Lan, Bowen Shi, Zhaoheng Ni, Sidd Srinivasan, Anurag Kumar, Brian Ellis, David Kant, Varun Nagaraja, Ernie Chang, Wei-Ning Hsu, et al. High fidelity text-guided music editing via single-stage flow matching. *arXiv preprint arXiv:2407.03648*, 2024.

Zitong Lan, Yiduo Hao, and Mingmin Zhao. Smartdj: Declarative audio editing with audio language model. In *The Fourteenth International Conference on Learning Representations*, 2026.

Yue Lei, Siqi Yang, Ting Zhong, and Fan Zhou. Harmonic canvas: Inversion-free editing for visually-guided music style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 29727–29737, 2026.

Shuyu Li, Shulei Ji, Zihao Wang, Songruoyao Wu, Jiaxing Yu, and Kejun Zhang. A survey on music generation from single-modal, cross-modal, and multi-modal perspectives. *ACM Computing Surveys*, 58(11):1–38, 2026.

Yizhi Li, Ruibin Yuan, Ge Zhang, Yinghao Ma, Xingran Chen, Hanzhi Yin, Chenghao Xiao, Chenghua Lin, Anton Ragni, Emmanouil Benetos, et al. Mert: Acoustic music understanding model with large-scale self-supervised training. In *International Conference on Learning Representations*, volume 2024, pp. 12181–12204, 2024.

Liwei Lin, Gus Xia, Yixiao Zhang, and Junyan Jiang. Arrange, inpaint, and refine: Steerable long-term music audio generation and editing via content-based controls. *arXiv preprint arXiv:2402.09508*, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

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.

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Chien-Yu Lu, Min-Xin Xue, Chia-Che Chang, Che-Rung Lee, and Li Su. Play as you like: Timbre-enhanced multi-modal music style transfer. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pp. 1061–1068, 2019.

Liangsi Lu, Xuhang Chen, Minzhe Guo, Shichu Li, Jingchao Wang, and Yang Shi. Chordedit: One-step low-energy transport for image editing. *arXiv preprint arXiv:2602.19083*, 2026.

Ethan Manilow, Gordon Wichern, Prem Seetharaman, and Jonathan Le Roux. Cutting music source separation some slakh: A dataset to study the impact of training data quality and quantity. In *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 45–49. IEEE, 2019.

Hila Manor and Tomer Michaeli. Zero-shot unsupervised and text-based audio editing using DDPM inversion. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=mCzyRdDak5.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2022.

Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6038–6047. IEEE, 2023.

Thao Nguyen, Yuheng Li, Utkarsh Ojha, and Yong Jae Lee. Visual instruction inversion: Image editing via image prompting. *Advances in Neural Information Processing Systems*, 36:9598–9613, 2023.

Nikhil Singh, Manuel Cherep, and Pattie Maes. Discovering and steering interpretable concepts in large generative music models. In *International Conference on Learning Representations*, volume 2026, pp. 84922–84941, 2026.

Royal Skousen. *Analogical modeling of language*. Springer Science & Business Media, 1989.

Ashutosh Srivastava, Tarun Ram Menta, Abhinav Java, Avadhoot Gorakh Jadhav, Silky Singh, Surgan Jandial, and Balaji Krishnamurthy. Reedit: Multimodal exemplar-based image editing. In *Proceedings of the Winter Conference on Applications of Computer Vision*, pp. 929–939, 2025.

Qwen Team. Qwen3. 5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

Fang-Duo Tsai, Shih-Lun Wu, Haven Kim, Bo-Yu Chen, Hao-Chung Cheng, and Yi-Hsuan Yang. Audio prompt adapter: Unleashing music editing abilities for text-to-music with lightweight finetuning. *arXiv preprint arXiv:2407.16564*, 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. In *Forty-second International Conference on Machine Learning*, 2025.

Qian Wang, Aleksandar Cvejić, Abdelrahman Eldesokey, and Peter Wonka. Editclip: Representation learning for image editing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15960–15970, 2025.

Yuancheng Wang, Zeqian Ju, Xu Tan, Lei He, Zhizheng Wu, Jiang Bian, et al. Audit: Audio editing by following instructions with latent diffusion models. *Advances in Neural Information Processing Systems*, 36:71340–71357, 2023a.

Zhichao Wang, Xinsheng Wang, Qicong Xie, Tao Li, Lei Xie, Qiao Tian, and Yuping Wang. Msmvc: High-fidelity source style transfer for non-parallel voice conversion by multi-scale style modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:3883–3895, 2023b.

Minz Won, Yun-Ning Hung, and Duc Le. A foundation model for music informatics. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1226–1230. IEEE, 2024.

Shangda Wu, Guo Zhancheng, Ruibin Yuan, Junyan Jiang, Seungheon Doh, Gus Xia, Juhan Nam, Xiaobing Li, Feng Yu, and Maosong Sun. Clamp 3: Universal music information retrieval across unaligned modalities and unseen languages. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 2605–2625, 2025.

Manjie Xu, Chenxing Li, Duzhen Zhang, Dan Su, Wei Liang, and Dong Yu. Prompt-guided precise audio editing with diffusion models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 55126–55143. PMLR, 21–27 Jul 2024.

Pengcheng Xu, Qingnan Fan, Fei Kou, Shuai Qin, Hong Gu, Ruoyu Zhao, Charles Ling, and Boyu Wang. Textualize visual prompt for image editing via diffusion bridge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 21779–21787, 2025.

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

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. In *International Conference on Learning Representations*, volume 2026, pp. 101737–101778, 2026.

Yixiao Zhang, Yukara Ikemiya, Woosung Choi, Naoki Murata, Marco A Martínez-Ramírez, Liwei Lin, Gus Xia, Wei-Hsiang Liao, Yuki Mitsufuji, and Simon Dixon. Instruct-musicgen: Unlocking text-to-music editing for music language models via instruction tuning. *arXiv preprint arXiv:2405.18386*, 2024a.

Yixiao Zhang, Yukara Ikemiya, Gus Xia, Naoki Murata, Marco A. Martínez Ramírez, Wei-Hsiang Liao, Yuki Mitsufuji, and Simon Dixon. Musicmagus: Zero-shot text-to-music editing via diffusion models. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pp. 7805–7813. ijcai.org, 2024b.

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.

APPENDIX

This appendix provides additional theoretical derivations, implementation details, experimental settings, and qualitative analyses that complement the main paper:

- **A: Fine-tuning MuQ for Edit Transition Representation.** We describe the construction and fine-tuning of the MuQ-based transition encoder used to represent source-to-target editing changes. Additional details on the training objective and representation learning strategy are also provided.
- **B: Additional Experimental Analysis.** We provide further analyses of the proposed framework, including the effect of attention guidance and the runtime efficiency of condition optimization and inference. These experiments complement the main results by examining both editing behavior and computational cost.
- **C: Training and Inference Pipeline.** We provide implementation details of the overall training and inference procedures, including model initialization, parameter optimization, and the condition-synthesis process used at inference time.
- **D: Dataset and Experimental Details.** We describe the construction of the instrument and style editing subsets, together with additional details on data organization and the textualization strategies used for editing conditions.
- **E: Additional Evaluation Details.** We provide supplementary details of the evaluation protocol, including the design of Edit Instruction Alignment (EIA) and the subjective evaluation procedure used in the user study.
- **F: Analysis of Source Conditioning in Flow-based Editing.** We provide a supplementary analysis of source conditioning under inversion-based and inversion-free formulations. In particular, we clarify why an explicit source description is not required in principle, while it may still influence practical editing when the source trajectory is approximated.
- **G: Additional Visual Cases.** We present additional qualitative results across different editing categories to further illustrate transformation transfer and source-content preservation under diverse musical examples.

A FINE-TUNING MUQ FOR EDIT TRANSITION REPRESENTATION

We adapt MuQ Zhu et al. (2025) to encode source–target music transitions. The fine-tuning data consist of triplets $\{(x_i^s, x_i^t, y_i)\}_{i=1}^N$, where x_i^s and x_i^t denote the source and edited music clips, respectively, and y_i denotes the corresponding edit category. Given a source–target pair, a shared MuQ encoder F_ϕ extracts endpoint representations $h_i^s = \mathrm{Pool}(F_\phi(x_i^s))$ and $h_i^t = \mathrm{Pool}(F_\phi(x_i^t))$. To preserve both endpoint information and edit direction, we construct the transition representation as:

$$r_i = \mathrm{Norm}\left(P_\psi\left([h_i^s; h_i^t; h_i^t - h_i^s]\right)\right), \tag{19}$$

where P_ψ denotes a lightweight projection head, $[\cdot;\cdot]$ denotes feature concatenation, and $\mathrm{Norm}(\cdot)$ denotes ℓ_2 normalization. For simplicity, we use Φ_ϕ to denote the complete transition encoder, including the MuQ backbone and the projection head.

We train the transition encoder to capture editing transformations that remain consistent across different musical contents. For a minibatch of B source–target pairs, let

$$P(i) = \{p \neq i \mid y_p = y_i\} \tag{20}$$

denote the set of samples sharing the same edit category as sample i. The transition encoder is optimized using the supervised contrastive objective:

$$\mathcal{L}_{\mathrm{rep}} = -\frac{1}{B}\sum_{i=1}^{B}\frac{1}{|P(i)|}\sum_{p\in P(i)}\log\frac{\exp\left(\mathrm{sim}(r_i, r_p)/\tau_c\right)}{\sum_{j\neq i}\exp\left(\mathrm{sim}(r_i, r_j)/\tau_c\right)}, \tag{21}$$

where $\mathrm{sim}(\cdot,\cdot)$ denotes cosine similarity and τ_c is the contrastive temperature. Each minibatch is constructed with at least two samples from every selected edit category, ensuring that $P(i)$ is

Table 8: Adaptation and editing time on 10-second audio clips.

Method	Adaptation time (s)	Editing time (s/clip)
Editor fine-tuning	1172.30	0.83
Ours	285.04	0.72

Table 9: Analysis of source-side attention guidance during condition synthesis.

Method	FAD ↓	FD ↓	KL ↓	CLAP ↑	F_0-PCC ↑	LPAPS ↓	EIA ↑
Ours	2.53	12.66	0.68	0.50	0.53	3.28	0.33
Ours + Attention	2.53	12.80	0.70	0.49	0.52	3.33	0.34

non-empty for all anchors. This objective encourages transitions corresponding to the same editing operation to remain close despite differences in musical content, while separating representations of different transformations. Through this adaptation, MuQ shifts from representing individual music attributes toward modeling transferable source-to-target editing transitions.

After fine-tuning, the transition encoder Φ_ϕ is frozen and reused during edit condition synthesis. Given a demonstration pair, its transition representation is computed as $r_d = \Phi_\phi(x_d^s, x_d^t)$. For a candidate edit condition c, the frozen editor generates $\hat{x}_d^t(c) = \mathcal{G}_{\theta_0}(x_d^s, c_d^s, c)$, and its induced transition is represented as $\hat{r}_d(c) = \Phi_\phi(x_d^s, \hat{x}_d^t(c))$. We measure transition consistency by:

$$\mathcal{L}_{\text{trans}}(c) = 1 - \frac{r_d^\top \hat{r}_d(c)}{\|r_d\|_2 \|\hat{r}_d(c)\|_2}. \tag{22}$$

This transition-level objective serves as an auxiliary constraint during edit condition synthesis. The generative editor remains frozen throughout the adaptation process, while only the MuQ-based transition encoder and projection head are updated during transition representation learning. Once trained, the transition encoder remains fixed, and test-time optimization updates only the soft edit condition.

B Additional Experimental Analysis

B.1 Runtime Analysis

Table 8 compares the adaptation and editing time of our method with an editor fine-tuning baseline following the approach of Wang et al. (2025). The baseline adapts the pretrained editor through parameter fine-tuning, whereas our method keeps the editor frozen and optimizes only a continuous edit condition at inference time. Adaptation time measures the cost of preparing each method for editing, while editing time measures the runtime per 10-second audio clip after adaptation. Our method requires substantially less adaptation time while achieving comparable per-clip editing speed, demonstrating the efficiency of condition-level adaptation over editor fine-tuning.

B.2 Analysis of Attention Guidance

We further examine whether source-side attention guidance benefits our framework. Attention manipulation has been explored in diffusion-based editing to preserve source structure and control editing regions Hertz et al. (2022). Following the attention-map manipulation strategy of PPAE Xu et al. (2024), we incorporate source-side cross-attention guidance into our condition-synthesis pipeline. As shown in Table 9, most metrics remain largely unchanged at the reported precision, while several show slight degradation. A possible explanation is that the effectiveness of attention manipulation depends on the underlying model architecture, and the guidance strategy may not transfer directly to our editor. Since we observe no consistent improvement, we omit attention guidance from the final framework.

Algorithm 1 Editing by Demonstration with Edit Condition Synthesis

Require: Transition pairs $\mathcal{D}$, demonstration (x_d^s, x_d^t, c_d^s), pretrained editor $(\mathcal{E}, \mathcal{G}_\theta, v_\theta)$, pretrained music encoder F_{ϕ_0}, weight λ
Ensure: Synthesized edit condition c_d^*
Transition adaptation: initialize $\phi \leftarrow \phi_0$
for each minibatch $\{(x_i^s, x_i^t)\}_{i=1}^B \subset \mathcal{D}$ **do**
    Encode transitions $r_i \leftarrow \Phi_\phi(x_i^s, x_i^t)$, $i = 1, \dots, B$
    Update ϕ by minimizing $\mathcal{L}_{\text{rep}}$ in Eq. 15
end for
Freeze Φ_ϕ
$z_d^s \leftarrow \mathcal{E}(x_d^s), \quad z_d^t \leftarrow \mathcal{E}(x_d^t), \quad r_d \leftarrow \Phi_\phi(x_d^s, x_d^t)$
Initialize continuous condition c
for each condition-optimization step **do**
    Sample t and $\epsilon \sim \mathcal{N}(0, I)$
    $z_{d,t}^s \leftarrow \alpha_t z_d^s + \sigma_t \epsilon, \quad z_{d,t}^t \leftarrow \alpha_t z_d^t + \sigma_t \epsilon$
    $\Delta v \leftarrow v_\theta(z_{d,t}^t, t, c) - v_\theta(z_{d,t}^s, t, c_d^s), \quad \Delta u \leftarrow \dot{\alpha}_t(z_d^t - z_d^s)$
    $\hat{x}_d^t \leftarrow \mathcal{G}_\theta(x_d^s, c_d^s, c), \quad \hat{r}_d \leftarrow \Phi_\phi(x_d^s, \hat{x}_d^t)$
    $\mathcal{L} \leftarrow \|\Delta v - \Delta u\|_2^2 + \lambda[1 - \text{sim}(r_d, \hat{r}_d)]$
    Update c using $\nabla_c \mathcal{L}$
end for
return $c_d^* \leftarrow c$

C Training and Inference Pipeline

Our framework contains two offline preparation stages followed by test-time soft-condition optimization. For condition initialization, we use the original pretrained MuQ encoder as a frozen audio backbone and append two Transformer layers and an MLP projector. Only the added layers are trained on paired music–text data to align audio representations with the embedding space of the editor's text encoder. Given the demonstration target x_d^t, this mapper produces an initial soft condition $c_d^{(0)}$, providing a suitable starting point for subsequent optimization without explicitly modeling the source–target edit.

Separately, we adapt a second MuQ model to obtain the transition encoder Φ_ϕ. Unlike the frozen MuQ used for initialization, this encoder is fine-tuned on source–target edit pairs to represent their relative change. For a pair (x_i^s, x_i^t), it produces $r_i = \Phi_\phi(x_i^s, x_i^t)$ and is trained with Eq. 15, encouraging the same editing transformation to remain close across different musical content. After training, Φ_ϕ is frozen and used only to provide transition-level guidance during condition synthesis.

We adopt MelodyFlow Lan et al. (2024) as the pretrained music editing backbone. MelodyFlow employs a quantizer-free EnCodec autoencoder to encode 48-kHz stereo audio into continuous latent representations at 25 frames per second. A flow-matching Diffusion Transformer operates on these latents, predicting a velocity field conditioned on the current latent state, flow time, and text embeddings. For editing, the source audio is encoded into the latent space, and the model generates an edited latent sequence under the target condition, which is subsequently decoded into audio by the EnCodec decoder. In our framework, we retain the pretrained audio codec and generative backbone, replacing the text embedding with the synthesized soft condition without updating the editor's parameters.

At inference time, all model parameters remain fixed and only the soft condition is optimized. Given a demonstration pair (x_d^s, x_d^t), we initialize c with $c_d^{(0)}$. The flow objective $\mathcal{L}_{\text{ECS}}$ aligns the condition-induced dynamics with the demonstrated source–target transition, while the frozen transition encoder compares the demonstrated edit $r_d = \Phi_\phi(x_d^s, x_d^t)$ with the edit induced by the current condition, providing $\mathcal{L}_{\text{trans}}$. The weighting coefficient for $\mathcal{L}_{\text{trans}}$ is set as $\lambda = 0.1$. The final condition $c_d^\star$ is obtained by optimizing Eq. 18 and is then reused to transfer the demonstrated transformation to unseen query music. Algorithm 1 summarizes the complete procedure.

D Dataset and Experimental Details

D.1 Datasets

Our paired dataset for exemplar-based music editing consists of two complementary subsets: instrument editing and style editing. The former focuses on fine-grained modifications to musical composition, while the latter captures higher-level stylistic transformations. Each sample contains a source audio x_{src}, an edited audio x_{edit}, and the corresponding edit type. We additionally include approximately 10% no-op samples with $x_{\mathrm{src}} = x_{\mathrm{edit}}$ in both subsets. Both subsets are split into training, validation, and test sets with an 80%/10%/10% ratio, with source-level separation to avoid overlap across splits.

Instrument Editing. The instrument editing subset is constructed from the MIDI data of Slakh2100-redux Manilow et al. (2019), covering 1,710 valid pieces. We consider nine common instruments: piano, acoustic guitar, electric guitar, bass, drums, violin, trumpet, saxophone, and flute. Using the isolated instrument tracks, we construct three types of edits: addition, removal, and replacement. Addition introduces an instrument absent from the source, removal deletes an existing instrument, and replacement substitutes one instrument with another. Each operation contains 2,500 samples, yielding 7,500 samples in total. All clips are 10 seconds long and sampled at 44.1 kHz. For this subset, splitting is performed by musical piece.

Style Editing. The style editing subset is constructed from music audio in MusicCaps and MeLBench Agostinelli (2023); Chowdhury et al. (2024). We consider seven representative styles: rock, jazz, classical, electronic, pop, folk, and country. All ordered transformations between distinct styles are included, resulting in $7 \times 6 = 42$ editing directions. For each direction, we select 200 distinct source clips, yielding 8,400 source–target pairs in total. Editing instructions are generated from ten fixed sentence templates instantiated with the source and target style labels, such as "Convert the track from jazz to classical." Each source clip and its corresponding instruction are then fed to Stable Audio 3 Small-Music Evans et al. (2025) in audio-to-audio mode to generate a 10-second edited target. For this subset, splitting is performed by source audio.

D.2 Textualization Prompts and Examples

For the textual conditioning baselines evaluated in Table 4, we use Qwen3.5-Omni-Plus Team (2026) to convert the same exemplar pairs used by our continuous condition into natural-language instructions. For each pair, the model receives the source and edited audio in a single request and produces both a *simple-text* description, summarizing the primary editing operation, and a *detailed-text* description, capturing finer musical changes. The pair-level prompt is shown below.

Pair-level textualization prompt.

Audio 1 is the SOURCE audio before editing: [source audio]
Audio 2 is the EDITED audio after editing: [edited audio]

Compare Audio 1 (SOURCE, before editing) with Audio 2 (EDITED, after editing). Infer only the audible transformation from SOURCE to EDITED.
Return two English imperative audio-editing instructions:
(1) A simple description of at most 20 words containing only the core transformation.
(2) A detailed description containing the supported audible changes and what should be preserved.
Return exactly one valid JSON object on a single line, with no Markdown or additional text:
`{"simple_description":"...", "detailed_description":"..."}`

- Describe the change precisely, including affected genre, instruments, timbre, rhythm, dynamics, spatial qualities, and other musical attributes when actually supported by the pair.
- Explicitly preserve important shared content.
- Do not invent changes caused merely by rendering noise or minor performance variation.

The following example illustrates the difference between the two textualization settings for the same demonstration.

Example: Instrument Removal

Editing type: Remove piano

Simple text:
"Remove the piano instrument while preserving all other musical elements."

Detailed text:
"Completely remove the piano track, including any melodies, chords, or arpeggios played by the instrument. Preserve all remaining musical elements exactly as they appear in the source audio, including the rhythm section (drums, bass), other melodic instruments, tempo, dynamics, and overall mix balance."

D.3 Direct Condition Inversion

To construct the comparison in Fig. 3, we consider a direct condition-inversion baseline using the same before–after demonstration (x_d^s, x_d^t) as our method. Given the frozen pretrained editor, we directly optimize a continuous condition c such that the demonstrated source is mapped to its target:

$$c_{\text{inv}}^{\star} = \arg\min_{c} d\left(\mathcal{G}_{\theta_0}(x_d^s, c_d^s, c), x_d^t\right), \tag{23}$$

where $d(\cdot,\cdot)$ denotes the endpoint reconstruction discrepancy. The pretrained editor remains frozen and only the continuous condition is optimized.

For a fair comparison, we use the same editor, demonstration, and condition initialization as in our method. The key difference is that direct inversion only requires the optimized condition to explain the observed source–target example, without explicitly constraining the relative transition between them. It may therefore recover a condition that reproduces the demonstrated outcome while also encoding attributes specific to the reference. Our method instead optimizes the condition using the demonstrated transition, encouraging the resulting representation to capture the change that can be transferred to new music inputs.

E Additional Evaluation Details

E.1 Design of Edit Instruction Alignment

EIA is designed to evaluate whether the transformation produced by the edited audio follows the intended editing direction, rather than assessing the final output alone. Following the transition-aware evaluation principle in instruction-based editing Chen et al. (2025), we explicitly measure the source-to-edited change. We independently encode the source and edited audio using the frozen CLaMP3 audio encoder. Their normalized embedding difference represents the realized edit direction, while the corresponding editing instructions are encoded by the CLaMP3 text encoder in the same cross-modal space. For each target editing category y, its text prototype t_y is obtained using the CLaMP3 text encoder Wu et al. (2025). To distinguish the target edit from semantically similar incorrect directions, we select five hard negatives from the same editing domain:

$$\mathcal{N}_y = \text{TopK}_{j\neq y,\ d_j=d_y}^{K=5} \text{sim}(t_y, t_j),$$

where d_y denotes the editing domain, such as instrument or style editing, and $\text{sim}(\cdot,\cdot)$ denotes cosine similarity. The hard negatives are determined solely by the similarity between CLaMP3 text prototypes, without manually defined category priorities.

EIA is then defined as the normalized probability assigned to the target editing direction among the target category and its five hard negatives:

$$\text{EIA}(x^s, \hat{x}^t, y) = \frac{\exp(\text{sim}(r, t_y)/\tau)}{\exp(\text{sim}(r, t_y)/\tau) + \sum_{j\in\mathcal{N}_y} \exp(\text{sim}(r, t_j)/\tau)},$$

where τ is the temperature parameter. EIA ranges from 0 to 1, with higher values indicating that the generated transformation better matches the intended editing direction and is more distinguishable from semantically similar incorrect edits. It therefore complements endpoint-based metrics by directly evaluating the realized editing change.

E.2 Subjective Evaluation Protocol.

We recruit 16 participants and randomly sample 30 examples from the editing subset for subjective evaluation. For each example, participants first listen to the before–after demonstration to understand the intended transformation, followed by the query audio and edited outputs from different methods presented in randomized order. Method identities are hidden throughout the evaluation. Figure 4 illustrates the evaluation interface using instrument addition, removal, and replacement as representative examples.

Each output is rated on a five-point Likert scale along three dimensions. *Edit Transfer* measures how faithfully the demonstrated transformation is reproduced on the query audio. *Source Preservation* evaluates whether musical content unrelated to the intended edit is retained. *Audio Quality* measures the overall perceptual quality, naturalness, and absence of audible artifacts.

F Analysis of Source Conditioning in Flow-based Editing

After synthesizing an edit condition from a demonstration, our method transfers it to a new source music input, where the source condition c^s may be explicitly provided or simply left empty. Although the choice of source conditioning is not central to our framework, it leads to a broader question about how source conditions behave in flow-based editing. We therefore provide the following supplementary analysis and share several observations.

In particular, we examine source conditioning under inversion-based and inversion-free formulations. We show that an explicit source description is not required by the direct formulation itself: if the corresponding inversion process uses an empty source condition, its equivalent direct form can retain the same empty condition. We further analyze why practical inversion-free methods may nevertheless exhibit different behavior when the source-specific inversion trajectory is replaced by a surrogate trajectory, in which case source conditioning can provide additional semantic anchoring.

Let x^s denote the source latent, c_s and c_t the source and target conditions, and $0 = t_0 < \cdots < t_K = 1$ the discrete flow timesteps, where $t = 0$ corresponds to the data domain and $t = 1$ to the noise domain. We denote the source inversion and target generation trajectories by z_k^s and z_k^t, respectively, which share the same terminal noise $z_K^s = z_K^t = \epsilon^s$. Under the same first-order solver with step size $h_k = t_k - t_{k-1}$, define the direct editing state as $q_k = x^s + z_k^t - z_k^s$. Since $q_K = x^s$ and $q_0 = x^t$, the sequence $\{q_k\}_{k=0}^K$ connects the observed source directly to the edited target, although it is constructed algebraically from the two inversion-based trajectories. Substituting their solver updates gives:

$$q_{k-1} = q_k - h_k \left[v_\theta(z_k^t, t_k, c_t) - v_\theta(z_k^s, t_k, c_s) \right]. \tag{24}$$

Therefore, inversion-based editing admits an exact direct-path interpretation driven by the relative velocity between the target and source branches. Importantly, this rewriting does not modify the states at which the pretrained velocity field is evaluated, and thus introduces no additional approximation beyond the numerical solver already used by the original inversion procedure.

The same argument immediately applies when the source inversion branch uses an empty condition. Setting $c_s = c_\emptyset$ in Eq. 24 gives the relative velocity $v_\theta(z_k^t, t_k, c_t) - v_\theta(z_k^s, t_k, c_\emptyset)$. Hence:

$$\boxed{\mathrm{Inv}(c_\emptyset) + \mathrm{Gen}(c_t) \;\equiv\; \mathrm{Direct}\left[v(c_t) - v(c_\emptyset)\right].} \tag{25}$$

This observation is useful because it separates two issues that are easily conflated. Removing explicit inversion does not itself require introducing a semantic source condition. The exact direct formulation simply inherits whatever condition is used by the source inversion branch. If the original inversion uses $c_\emptyset$, then its algebraically equivalent direct formulation also uses $c_\emptyset$.

The practical FlowEdit algorithm Kulikov et al. (2025), however, makes an additional approximation in order to avoid explicitly computing the source inversion trajectory. Instead of evaluating the relative velocity on the source-specific state z_k^s, it constructs a surrogate state $\hat{z}_k^s = (1 - t_k)x^s + t_k\epsilon_k$, where $\epsilon_k \sim \mathcal{N}(0, I)$. This construction follows the desired noisy interpolation and therefore provides an appropriate marginal state at timestep t_k, but it is generally not the same sample-wise state that would be obtained by deterministic inversion of the particular source audio. Let $\delta_k = \hat{z}_k^s - z_k^s$ denote this trajectory mismatch and $e_k = \hat{q}_k - q_k$ the accumulated error between the surrogate

and exact direct paths. Since the target-side state is constructed relative to the same surrogate source state, its perturbation becomes $e_k + \delta_k$. The resulting local velocity error is:

$$\begin{aligned}\xi_k = &\Big[v_\theta(z_k^t + e_k + \delta_k, t_k, c_t) - v_\theta(z_k^t, t_k, c_t)\Big] \\ &- \Big[v_\theta(z_k^s + \delta_k, t_k, c_s) - v_\theta(z_k^s, t_k, c_s)\Big], \quad e_{k-1} = e_k - h_k \xi_k.\end{aligned} \tag{26}$$

This expression identifies where the exact inversion-to-direct equivalence is lost. The source and target branches are both perturbed by the surrogate trajectory, but the two conditional velocity fields need not respond identically to the same perturbation. For a nonlinear conditional flow model, their responses generally depend on both the local state and the conditioning signal, so the two perturbation terms in Eq. 26 are not guaranteed to cancel. The resulting discrepancy therefore originates from trajectory substitution rather than from the direct reformulation itself.

This distinction also explains why an empty source condition can be theoretically valid in the exact formulation yet behave differently in practical inversion-free editing. Consider two FlowEdit variants evaluated on the same surrogate source state $\hat{z}_k^s$, one using a source description c_{src} and the other using $c_\emptyset$. Their difference is entirely determined by the source-side conditional velocity, which we denote as $b_k = v_\theta(\hat{z}_k^s, t_k, c_{\mathrm{src}}) - v_\theta(\hat{z}_k^s, t_k, c_\emptyset)$. Ignoring higher-order feedback through the evolving editing state, this discrepancy accumulates approximately as:

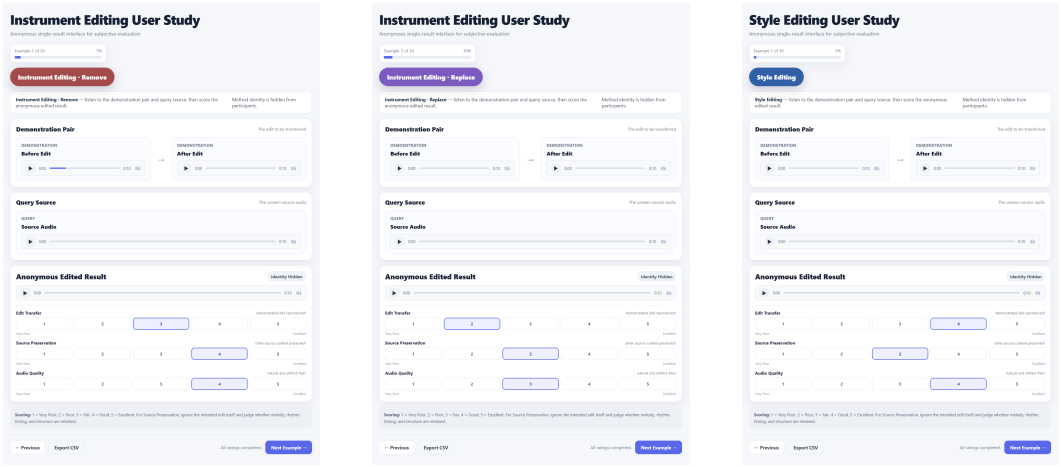
$$\hat{q}_0^\emptyset - \hat{q}_0^{\mathrm{src}} \approx -\sum_{k=1}^{K} h_k b_k. \tag{27}$$

This term does not contradict Eq. 25. On the true inversion trajectory, the state z_k^s is itself tied to the particular source sample through deterministic inversion. After replacing it with $\hat{z}_k^s$, part of this sample-specific anchoring is lost, and the conditional velocity field plays a larger role in maintaining the source semantics.

Overall, this analysis shows that an explicit source condition is not theoretically necessary for direct editing: an empty condition remains valid when inherited from the corresponding inversion formulation. However, once the source-specific trajectory is replaced by a surrogate trajectory, source conditioning can still affect the resulting edit by providing additional semantic information about the input. Thus, source conditioning should be viewed as an optional source-side anchor rather than a prerequisite of the editing formulation, while the target condition remains responsible for specifying the desired transformation.

G Additional Visual Cases

We provide additional qualitative examples in Figs. 5–9 to further illustrate the behavior of our method across diverse editing scenarios. The cases cover both style transformations and instrument-level operations, including addition, removal, and replacement. For each example, we present the demonstration pair, the music to be edited, and the corresponding edited result together with its time–frequency representation. These examples complement the main-text results by showing that the demonstrated transformation can be consistently transferred across different musical contents while preserving source-dependent structures unrelated to the intended edit.



(a) Instrument Addition (b) Instrument Removal (c) Style Editing

Figure 4: User study interface for instrument addition, removal, and style editing.

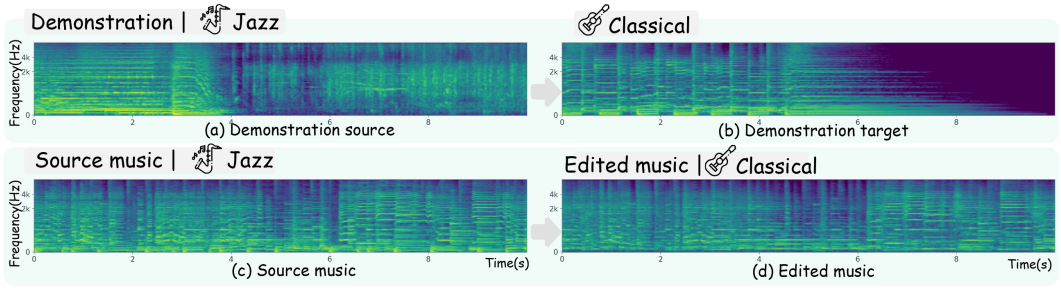


Figure 5: Jazz → Classical: The demonstration shows a jazz piece and its classical counterpart, defining the corresponding style transformation. The music to be edited is another jazz piece with different content, to which the demonstrated transformation is transferred.

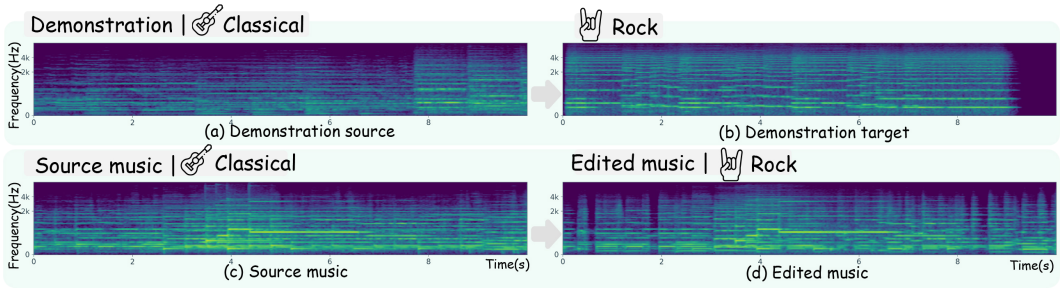


Figure 6: Classical → Rock: The demonstration defines a classical-to-rock transformation. The model applies this change to another classical piece while preserving its original musical content.

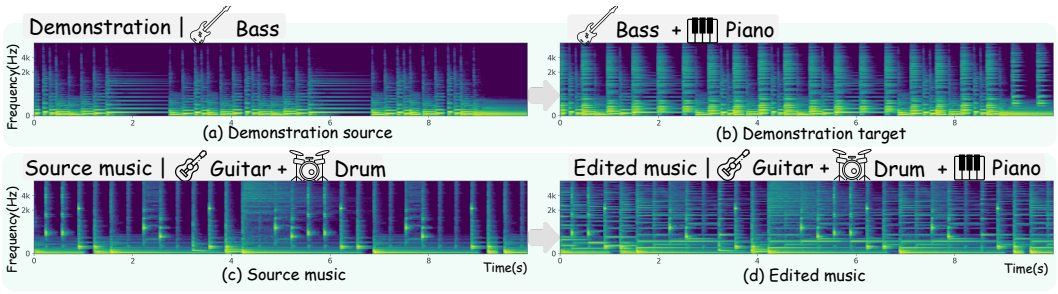


Figure 7: Add Piano: The demonstration edits a bass-only piece into piano + bass, defining the addition of piano. The music to be edited contains acoustic guitar and drums, and the model adds piano while preserving the existing instruments.

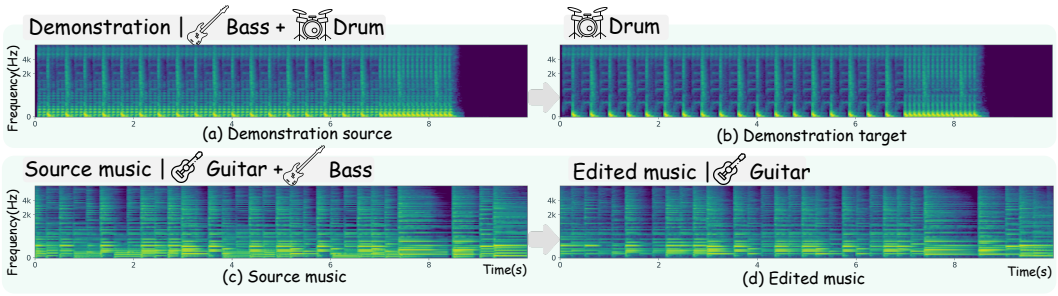


Figure 8: Remove Bass: The demonstration edits bass + drums into drums only, defining the removal of bass. The music to be edited contains piano, acoustic guitar, and bass, and the model removes the bass while preserving the remaining instrument.

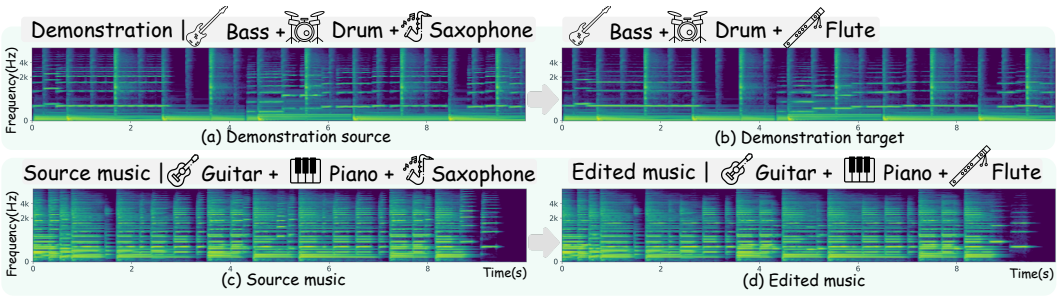


Figure 9: Saxophone → Flute: The demonstration edits bass + drums + saxophone into bass + drums + flute, defining the saxophone-to-flute replacement. The music to be edited contains piano, electric guitar, and saxophone, and the model replaces the saxophone with flute while preserving the other instruments.