# MGE-LDM: Joint Latent Diffusion for Simultaneous Music Generation and Source Extraction

**Yunkee Chae**[1,2] **Kyogu Lee** [1,2,3,4]
[1] Music and Audio Research Group (MARG)
[2] Interdisciplinary Program in Artificial Intelligence (IPAI)
[3] AIIS, [4] Department of Intelligence and Information
Seoul National University
{yunkimo95, kglee}@snu.ac.kr

## Abstract

We present **MGE-LDM**, a unified latent diffusion framework for simultaneous music generation, source imputation, and query-driven source separation. Unlike prior approaches constrained to fixed instrument classes, MGE-LDM learns a joint distribution over full mixtures, submixtures, and individual stems within a single compact latent diffusion model. At inference, MGE-LDM enables (1) complete mixture generation, (2) partial generation (i.e., source imputation), and (3) text-conditioned extraction of arbitrary sources. By formulating both separation and imputation as conditional inpainting tasks in the latent space, our approach supports flexible, class-agnostic manipulation of arbitrary instrument sources. Notably, MGE-LDM can be trained jointly across heterogeneous multi-track datasets (e.g., Slakh2100, MUSDB18, MoisesDB) without relying on predefined instrument categories. Audio samples are available at: `anonymous_link`.

## 1 Introduction

Recent advances in generative modeling have significantly accelerated progress in the music domain, especially in music audio synthesis [1–4], accompaniment generation [5–7], and music source separation [8, 9]. A recent line of work has investigated solving these tasks simultaneously within a single model by modeling the joint distribution of multi-track stems within a unified diffusion backbone [10–12]. However, these approaches typically rely on predefined instrument classes for each track or assume that the mixture waveform is the linear sum of its constituent stems. While the additive assumption is valid in the waveform domain, it is incompatible with the nonlinear encoder-decoder structure of latent diffusion models, limiting the applicability of such methods in compressed latent spaces.

To address these limitations, we introduce **MGE-LDM**, a class-agnostic latent diffusion framework that jointly unifies music generation, partial generation (source imputation), and arbitrary source extraction. Our approach models three interrelated latent variables: mixture, submixture, and source, within a single diffusion backbone.

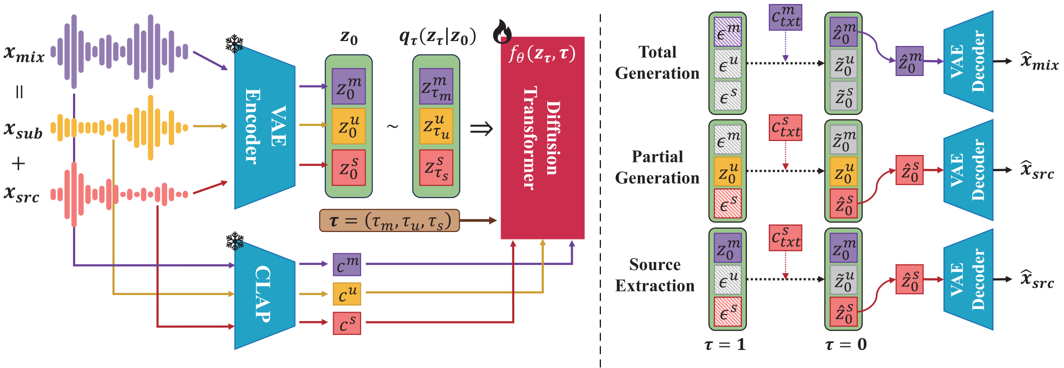
## 2 Method

Figure 1 provides an overview of the proposed MGE-LDM pipeline. We begin by defining the construction of training triplets and then describe how they are used for joint diffusion-based training and inpainting-based inference.

### 2.1 Formulating Joint Latent Representation

Let $\{x_i\}_{i \in I}$ denote the set of time-domain audio stems, where the number of sources $|I|$ may vary across mixtures depending on their instrumentation. We uniformly sample an index $j \in I$ and define

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: AI for Music.



(a) Training pipeline

(b) Inference pipeline

Figure 1: **Overview of MGE-LDM.** (a) *Training pipeline*: We train a three-track latent diffusion model on mixtures, submixtures, and sources. (b) *Inference pipeline*: At test time, task-specific latents are either generated or inpainted based on available context and text prompts.

the mixture, submixture, and source triplet $\left(x^{(m)}, x^{(u)}, x^{(s)}\right)$ as:

$$x^{(m)} = \sum_{i \in I} x_i, \quad x^{(s)} = x_j, \quad x^{(u)} = \sum_{i \in I \setminus \{j\}} x_i.$$

We encode each element of this triplet using a pretrained VAE [13] encoder $E$, resulting in latent representations $z^{(m)}, z^{(u)}, z^{(s)} \in \mathbb{R}^{C \times L}$, where $C$ and $L$ denote the latent channel and temporal dimensions, respectively. This formulation naturally accommodates mixtures with a variable number of stems. Regardless of the number of instruments present, any publicly available multi-track dataset can be decomposed into mixture, submixture, and source components for joint latent modeling, even with loosely labeled tracks such as `other` that aggregate multiple instruments, as in MUSDB18 [14].

### 2.2 Latent Diffusion Training with Three-Track Embeddings

We build upon the Stable Audio framework [3], employing a Diffusion Transformer (DiT) backbone [15] and training the model under the v-objective [16]. Let the composite latent input be defined as $\mathbf{z}_0 = \left(z_0^{(m)}, z_0^{(u)}, z_0^{(s)}\right) \in \mathbb{R}^{3 \times C \times L}$, where $z_0^{(k)} \in \mathbb{R}^{C \times L}$ are (clean) track embeddings, with $k \in K = \{m, u, s\}$ denoting the track types – mixture, submixture, and source, respectively.

We aim to estimate the score $\nabla_{\mathbf{z}_\tau} \log q_\tau(\mathbf{z}_\tau)$ of the perturbed latent $\mathbf{z}_\tau$ across continuous noise levels $\tau \in [\tau_{\text{min}}, 1]$. To do so, we perturb $\mathbf{z}_0$ with Gaussian noise according to:

$$\mathbf{z}_\tau = \alpha_\tau \mathbf{z}_0 + \beta_\tau \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{1}$$

where the noise scaling coefficients are defined as $\alpha_\tau = \cos(\phi_\tau)$ and $\beta_\tau = \sin(\phi_\tau)$, where $\phi_\tau = \frac{\pi}{2}\tau$. Here, $\tau \sim \mathcal{U}([\tau_{\text{min}}, 1])$ is sampled from a truncated uniform distribution with $\tau_{\text{min}} = 0.02$ for stability.

A denoising network $f_\theta(\mathbf{z}_\tau, \tau, \mathbf{c})$ is trained to estimate the score $\nabla_{\mathbf{z}_\tau} \log q_\tau(\mathbf{z}_\tau | \mathbf{c})$ using the v-objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{z}_0, \boldsymbol{\epsilon}, \tau} \left|\left| f_\theta(\mathbf{z}_\tau, \tau, \mathbf{c}) - \boldsymbol{v}_\tau \right|\right|_2^2, \quad \boldsymbol{v}_\tau = \frac{\partial \mathbf{z}_\tau}{\partial \phi_\tau} = \alpha_\tau \boldsymbol{\epsilon} - \beta_\tau \mathbf{z}_0. \tag{2}$$

The conditioning vector $\mathbf{c} = (c^{(m)}, c^{(u)}, c^{(s)})$ is derived using the audio branch of a pretrained CLAP encoder [17], applied to each component. To enable classifier-free guidance (CFG) [18], each $c^{(k)}$ is independently dropped out with probability $p$ during training.

### 2.3 Inference via Conditional Sampling in Latent Space

In the image domain, *inpainting* refers to reconstructing missing or corrupted regions of an image by conditioning on the surrounding pixels. Diffusion models have demonstrated strong zero-shot inpainting capabilities, enabling arbitrary mask completion without retraining [19, 20]. We extend this paradigm to the latent domain of music, operating over a joint distribution of mixture, submixture, and source embeddings.

Downstream tasks are formulated as conditional generation problems, where known latents are treated as observed and unknown ones are sampled as missing components. In all inference modes, we condition on natural-language queries using CLAP embeddings.

Table 1: **Total generation results.** Reported scores are FAD ↓, computed against mixture references from each test set. Values in parentheses indicate generation results conditioned on the prompt "*The sound of the bass, drums, guitar, and piano*".

| Model | | Train Set | | | | Test Set | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\mathbf{S}_A$ | $\mathbf{S}_B$ | $\mathbf{M}_u$ | $\mathbf{M}_o$ | $\mathbf{S}_A$ | $\mathbf{S}_{\text{Full}}$ | $\mathbf{M}_u$ | $\mathbf{M}_o$ |
| MSDM [10] | | ✓ | × | × | × | 4.21 | 6.04 | 7.92 | 7.41 |
| MSG-LD [12] | | ✓ | × | × | × | 1.38 | 1.55 | 4.61 | 4.26 |
| MGE (ours) | $\mathcal{T}_1$ | ✓ | × | × | × | **0.47** (3.57) | 1.79 | 6.34 | 5.90 |
| | $\mathcal{T}_2$ | ✓ | ✓ | × | × | 3.14 (2.24) | **0.63** | 5.46 | 4.73 |
| | $\mathcal{T}_3$ | × | × | ✓ | ✓ | 8.80 (3.96) | 6.56 | 2.87 | 1.59 |
| | $\mathcal{T}_4$ | ✓ | ✓ | ✓ | ✓ | 6.83 (5.05) | 4.22 | **2.78** | **1.47** |

When text conditioning is required, we use the text branch of CLAP to produce the prompt embedding $c^{(k)} = \text{CLAP}_{\text{text}}(c^{(k)}_{\text{text}})$, where $c^{(k)}_{\text{text}}$ is a free-form natural language description (e.g., "*the sound of an electric guitar*").

**Total Generation.** Let $p_\theta(z^{(m)}, z^{(u)}, z^{(s)})$ denote the implicit model distribution whose score function is induced by the denoising network $f_\theta$. To synthesize a complete mixture, we condition only on the mixture prompt embedding $c^{(m)}$, or omit all conditions for unconditional generation. We sample mixture latent $\hat{z}^{(m)}$ as:

$$\hat{z}^{(m)}, \tilde{z}^{(u)}, \tilde{z}^{(s)} \sim p_\theta(z^{(m)}, z^{(u)}, z^{(s)} | c^{(m)}, \varnothing, \varnothing), \quad (3)$$

where $\tilde{z}^{(u)}$ and $\tilde{z}^{(s)}$ are auxiliary latents that are discarded. Finally, the synthesized mixture waveform $\hat{x}^{(m)}$ is obtained by decoding $\hat{z}^{(m)}$ through the pretrained VAE decoder $D$ (i.e., $\hat{x}^{(m)} = D(\hat{z}^{(m)})$).

Hereafter, we use $\tilde{z}^{(k)}$ to denote any dummy latent that is not retained during inference.

**Partial Generation.** *Partial generation*, also known as *source imputation*, refers to the task of generating missing stems given partially observed sources. We approach this iteratively to progressively reconstruct the full mixture from the partial input.

Let $\bar{\mathcal{I}} = \{c_1, ..., c_J\}$ be an (ordered) set of CLAP-derived text embeddings, each corresponding to a target source description to be imputed. Let $x^{(u)}_0$ denote the waveform mixture of the observed sources, and let $z^{(u)}_0 = E(x^{(u)}_0)$ be its latent representation. We initialize the submixture latent with $z^{(u)}_0$, and generate each missing source sequentially.

At each step $j \in \{1, ..., J\}$, we sample a new source latent $\hat{z}^{(s)}_j$ conditioned on the current submixture and the text embedding $c^{(s)}_j$:

$$\tilde{z}^{(m)}_j, \hat{z}^{(s)}_j \sim p_\theta(z^{(m)}, z^{(s)} | z^{(u)}_{j-1}, \varnothing, \varnothing, c^{(s)}_j). \quad (4)$$

We then update the submixture by accumulating the decoded sources: $z^{(u)}_j = E\left(\sum_{l=0}^{j-1} D(\hat{z}^{(s)}_l)\right)$.

After $J$ iterations, we obtain the full set of imputed sources $\{\hat{z}^{(s)}_j\}_{j=1}^{J}$, and reconstruct the final mixture waveform as $x^{(m)} = x^{(u)}_0 + \sum_{j=1}^{J} D(\hat{z}^{(s)}_j)$.

**Source Extraction.** Text-driven extraction of an arbitrary stem is performed by conditioning on a natural-language prompt. Given a prompt embedding $c^{(s)}$, we treat the mixture latent $z^{(m)}$ as observed and inpaint the submixture and source tracks:

$$\tilde{z}^{(u)}, \hat{z}^{(s)} \sim p_\theta(z^{(u)}, z^{(s)} \mid z^{(m)}, \varnothing, \varnothing, c^{(s)}), \quad (5)$$

where $\tilde{z}^{(u)}$ is an auxiliary prediction that is discarded. Finally, the isolated source waveform is reconstructed via $\hat{x}^{(s)} = D(\hat{z}^{(s)})$.

We also introduce an advanced training framework for improving the inpainting performance of our three-track diffusion model in Appendix C. All reported results are obtained using this framework with track-wise adaptive timesteps.

## 3 Results

We evaluate MGE-LDM on three tasks: total generation, partial generation, and source extraction. We train and evaluate on three multi-track music datasets: Slakh2100 [21], MUSDB18 [14] (denoted $\mathbf{M}_u$ ), and MoisesDB [22] (denoted $\mathbf{M}_o$ ). For Slakh2100, we define two subsets: $\mathbf{S}_A$ , containing only `bass`, `drums`, `guitar`, and `piano` stems to match the MSDM and MSG-LD setup; and $\mathbf{S}_B$ , which includes all remaining stems. Each result table indicates the training dataset(s) used and reports performance across multiple test sets. We train our models on various dataset combinations to

Table 2: **Partial generation results.** Scores are reported using *sub*-FAD ↓, which measures the distance between the reference mixture and the sum of given and generated sources. Each column header (e.g., B, D, G) indicates the target source being generated, conditioned on the remaining stems.

| **Model** | | Train Set | | | | $\mathbf{S}_A$ | | | | | | | | | | | | | | $\mathbf{S}_B$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\mathbf{S}_A$ | $\mathbf{S}_B$ | $\mathbf{M}_u$ | $\mathbf{M}_o$ | B | D | G | P | BD | BG | BP | DG | DP | GP | BDG | BDP | BGP | DGP | Brs. | C.P. | Org. | Pipe | Reed | Str. | S.Lead | S.Pad |
| MSDM [10] | | ✓ | × | × | × | 0.56 | 1.06 | 0.49 | 0.7 | 2.23 | 1.56 | 1.95 | 1.64 | 1.83 | 2.31 | 3.09 | 3.53 | 5.72 | 3.86 | - | - | - | - | - | - | - | - |
| MSG-LD [12] | | ✓ | × | × | × | **0.33** | **0.34** | **0.49** | **0.48** | **0.70** | **1.08** | **1.05** | **0.86** | **0.83** | 1.47 | **1.43** | **1.42** | 2.31 | **1.76** | - | - | - | - | - | - | - | - |
| | $\mathcal{T}_1$ | ✓ | × | × | × | 1.02 | 1.41 | 1.17 | 1.19 | 1.15 | 1.29 | 1.25 | 1.69 | 1.65 | **1.14** | 1.80 | 1.84 | **1.45** | 1.84 | **1.45** | 0.68 | 0.23 | 3.48 | 5.58 | 1.38 | 4.47 | 1.08 |
| MGE | $\mathcal{T}_2$ | ✓ | ✓ | × | × | 2.11 | 2.99 | 1.99 | 2.74 | 4.07 | 2.32 | 4.18 | 3.54 | 3.9 | 3.18 | 4.93 | 5.69 | 4.25 | 4.66 | 5.96 | **0.41** | 1.03 | 3.66 | 3.52 | 2.79 | 0.88 | 2.32 |
| (ours) | $\mathcal{T}_3$ | × | × | ✓ | ✓ | 1.43 | 1.29 | 3.34 | 2.30 | 1.85 | 3.64 | 2.83 | 2.95 | 2.36 | 4.39 | 3.30 | 3.57 | 6.03 | 3.86 | 3.58 | 0.58 | **0.15** | 0.22 | **0.56** | **0.61** | 0.54 | 0.48 |
| | $\mathcal{T}_4$ | ✓ | ✓ | ✓ | ✓ | 1.14 | 1.50 | 3.75 | 2.47 | 2.06 | 4.06 | 2.82 | 3.37 | 2.74 | 4.55 | 3.94 | 4.05 | 5.66 | 4.06 | 5.09 | 0.42 | 0.56 | **0.20** | 3.14 | 3.95 | **0.31** | **0.40** |

Table 3: **Source extraction results.** Metrics are reported as Log-Mel L1 distance ↓. For baseline models, scores are shown only for stems included in their fixed output set.

| **Model** | | Train Set | | | | $\mathbf{S}_A$ | | | | $\mathbf{S}_B$ | | | | | | | | $\mathbf{M}_u$ | | | $\mathbf{M}_o$ | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\mathbf{S}_A$ | $\mathbf{S}_B$ | $\mathbf{M}_u$ | $\mathbf{M}_o$ | B | D | G | P | Brs. | C.P. | Org. | Pipe | Reed | Str. | S.Lead | S.Pad | V | B | D | V | B | D | G | P | B.Str | Perc. |
| HDemucs | | × | × | ✓ | × | 1.49 | 0.90 | - | - | - | - | - | - | - | - | - | - | **1.50** | 1.99 | 1.53 | **0.83** | 1.71 | 1.10 | - | - | - | - |
| AudioSep [23] | | × | × | × | × | 2.36 | 1.67 | 3.41 | 2.42 | **3.13** | 2.84 | 3.26 | 3.04 | 3.15 | 2.57 | 2.8 | 2.06 | 2.66 | 4.07 | 1.89 | 1.54 | 3.37 | 1.87 | 1.31 | 1.42 | 1.70 | **2.36** |
| MSDM [10] | | ✓ | × | × | × | 1.90 | 1.51 | 3.32 | 2.70 | - | - | - | - | - | - | - | - | - | 2.56 | 1.69 | - | 2.15 | 1.31 | 1.28 | 1.51 | - | - |
| MSG-LD [12] | | ✓ | × | × | × | **1.20** | 1.24 | 2.24 | 1.85 | - | - | - | - | - | - | - | - | - | 1.96 | 1.60 | - | 1.72 | 1.49 | 2.36 | 2.06 | - | - |
| | $\mathcal{T}_1$ | ✓ | × | × | × | 1.28 | **0.66** | **1.27** | **1.07** | 3.22 | 3.07 | 3.13 | 3.11 | 3.30 | 2.77 | 2.68 | 2.30 | 3.80 | 1.91 | 1.33 | 5.15 | 1.61 | 1.10 | 2.86 | 2.68 | 2.03 | 2.94 |
| MGE | $\mathcal{T}_2$ | ✓ | ✓ | × | × | 1.68 | 2.71 | 2.69 | 2.16 | 3.43 | **2.16** | **1.84** | 2.33 | 3.07 | 2.44 | **2.31** | **1.93** | 3.55 | 2.14 | 2.15 | 4.66 | 1.86 | 2.11 | 2.28 | 2.18 | 1.93 | **2.36** |
| (ours) | $\mathcal{T}_3$ | × | × | ✓ | ✓ | 1.80 | 0.99 | 2.89 | 2.01 | 3.17 | 2.51 | 3.61 | 2.13 | 2.86 | 2.22 | 2.78 | 2.22 | 1.85 | **1.56** | 1.17 | 0.98 | 1.10 | 0.90 | 1.04 | 1.58 | **1.62** | 2.49 |
| | $\mathcal{T}_4$ | ✓ | ✓ | ✓ | ✓ | 1.67 | 0.83 | 2.61 | 1.77 | 3.15 | 2.29 | 2.22 | **1.95** | **2.61** | **1.85** | 2.71 | 3.68 | 1.76 | **1.56** | **1.13** | 1.01 | **1.07** | **0.86** | **1.02** | **1.40** | 2.25 | 2.69 |

evaluate robustness under different source distributions and stem configurations. Unless otherwise specified, partial generation and source extraction are performed using text prompts of the form "*The sound of the* `{label}`." Abbreviations for all stem labels are listed in Appendix Table 4, and additional ablation results are provided in Appendix F.

Table 1 presents FAD [24] scores for the total generation task. Note that the $\mathbf{S}_A$ test set contains only mixtures of `bass`, `drums`, `guitar`, and `piano`, while $\mathbf{S}_{\text{Full}}$ corresponds to the full Slakh2100 test set, which includes a broader and more diverse set of instruments. We first compare our model $\mathcal{T}_1$ against MSDM [10] and MSG-LD [12], where all models are trained on $\mathbf{S}_A$ . Our model achieves the lowest FAD on the $\mathbf{S}_A$ test set, demonstrating superior fidelity in generating standard four-stem mixtures. Furthermore, our model can be trained on extended combinations of datasets, as in $\mathcal{T}_2$ , $\mathcal{T}_3$ , and $\mathcal{T}_4$ , which leads to improved FAD scores on the $\mathbf{S}_{\text{Full}}$ , $\mathbf{M}_u$ , and $\mathbf{M}_o$ test sets.

Table  2 presents results for partial generation, evaluated using the *sub*-FAD metric, which is adopted in [10, 12, 25]. On $\mathbf{S}_A$ , our model $\mathcal{T}_1$ performs worse than MSG-LD for single-source imputation but shows competitive or better performance as the number of generated stems increases. For imputation tasks involving broader instrument classes in $\mathbf{S}_B$ , models trained on more diverse datasets perform better.

We also evaluate text-queried source extraction using the Log Mel L1 distance, following MSG-LD [12]. Table 3 presents results for a variety of stems across the $\mathbf{S}_A$ , $\mathbf{S}_B$ , $\mathbf{M}_u$ , and $\mathbf{M}_o$ test sets. Alongside MSDM and MSG-LD, we compare two additional baselines: HDemucs [26] and AudioSep [23]. Our model $\mathcal{T}_1$ , trained solely on $\mathbf{S}_A$ , performs strongly on the canonical Slakh stems (`bass`, `drums`, `guitar`, `piano`), outperforming MSG-LD on all but `bass`. However, it generalizes poorly to less common stems and real-world recordings. By expanding the training set to encompass the full Slakh2100 dataset, model $\mathcal{T}_2$ achieves improved performance on categories such as `chromatic percussion`, `organ`, `synth lead`, and `synth pad`, demonstrating the importance of broader intra-domain coverage. Model $\mathcal{T}_3$ generalizes competitively with $\mathcal{T}_2$ to synthetic stems, even outperforming some stems such as `drums`, and model $\mathcal{T}_4$ exhibits robust performance across both synthetic and real-world domains.

# 4 Conclusion

We have presented MGE-LDM, a unified class-agnostic latent diffusion framework that jointly models mixtures, submixtures, and individual sources for music generation, stem completion, and text-driven extraction. By formulating stem completion and source extraction as conditional inpainting in a shared latent space, and by introducing track-dependent timestep conditioning, we overcome the limitations of fixed-class, additive mixing assumptions and achieve flexible manipulation of arbitrary instrument tracks. Empirically, MGE-LDM matches or exceeds specialized baselines on the Slakh2100 generation and separation benchmarks while uniquely supporting zero-shot, language-guided extraction across heterogeneous multi-track datasets.

# References

[1] 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)*, pages 1206–1210. IEEE, 2024.

[2] 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*, 2024.

[3] Zach Evans, Julian D Parker, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons. Long-form music generation with latent diffusion. *International Society for Music Information Retrieval Conference*, 2024.

[4] 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)*, pages 1–5. IEEE, 2025.

[5] Javier Nistal, Marco Pasini, Cyran Aouameur, Maarten Grachten, and Stefan Lattner. Diff-a-riff: Musical accompaniment co-creation via latent diffusion models. *International Society for Music Information Retrieval Conference*, 2024.

[6] Javier Nistal, Marco Pasini, and Stefan Lattner. Improving musical accompaniment co-creation via diffusion transformers. *arXiv preprint arXiv:2410.23005*, 2024.

[7] Marco Pasini, Maarten Grachten, and Stefan Lattner. Bass accompaniment generation via latent diffusion. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1166–1170. IEEE, 2024.

[8] Ge Zhu, Jordan Darefsky, Fei Jiang, Anton Selitskiy, and Zhiyao Duan. Music source separation with generative flow. *IEEE Signal Processing Letters*, 29:2288–2292, 2022.

[9] Noah Schaffer, Boaz Cogan, Ethan Manilow, Max Morrison, Prem Seetharaman, and Bryan Pardo. Music separation enhancement with generative modeling. *arXiv preprint arXiv:2208.12387*, 2022.

[10] Giorgio Mariani, Irene Tallini, Emilian Postolache, Michele Mancusi, Luca Cosmo, and Emanuele Rodolà. Multi-source diffusion models for simultaneous music generation and separation. In *The Twelfth International Conference on Learning Representations*, 2024.

[11] Emilian Postolache, Giorgio Mariani, Luca Cosmo, Emmanouil Benetos, and Emanuele Rodolà. Generalized multi-source inference for text conditioned music diffusion models. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6980–6984. IEEE, 2024.

[12] Tornike Karchkhadze, Mohammad Rasool Izadi, and Shlomo Dubnov. Simultaneous music separation and generation using multi-track latent diffusion models. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2025.

[13] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

[14] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. The MUSDB18 corpus for music separation, December 2017. URL `https://doi.org/10.5281/zenodo.1117372`.

[15] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[16] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

[17] Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[18] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *NeurIPS Workshop on Deep Generative Models and Downstream Applications*, 2021.

[19] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.

[20] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[21] 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)*, pages 45–49. IEEE, 2019.

[22] Igor Pereira, Felipe Araújo, Filip Korzeniowski, and Richard Vogl. Moisesdb: A dataset for source separation beyond 4-stems. *International Society for Music Information Retrieval Conference*, 2023.

[23] Xubo Liu, Qiuqiang Kong, Yan Zhao, Haohe Liu, Yi Yuan, Yuzhuo Liu, Rui Xia, Yuxuan Wang, Mark D Plumbley, and Wenwu Wang. Separate anything you describe. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2024.

[24] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fr\'echet audio distance: A metric for evaluating music enhancement algorithms. In *INTERSPEECH*, 2018.

[25] 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.

[26] Alexandre Défossez. Hybrid spectrogram and waveform source separation. *arXiv preprint arXiv:2111.03600*, 2021.

[27] Yi Luo and Nima Mesgarani. Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation. *IEEE/ACM transactions on audio, speech, and language processing*, 27(8):1256–1266, 2019.

[28] Enric Gusó, Jordi Pons, Santiago Pascual, and Joan Serrà. On loss functions and evaluation metrics for music source separation. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 306–310. IEEE, 2022.

[29] Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis Bach. Music source separation in the waveform domain. *arXiv preprint arXiv:1911.13254*, 2019.

[30] Yi Luo and Jianwei Yu. Music source separation with band-split rnn. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:1893–1901, 2023.

[31] Ui-Hyeop Shin, Sangyoun Lee, Taehan Kim, and Hyung-Min Park. Separate and reconstruct: Asymmetric encoder-decoder for speech separation. In *Advances in Neural Information Processing Systems*, volume 37, pages 52215–52240, 2024.

[32] Y Cem Subakan and Paris Smaragdis. Generative adversarial source separation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 26–30. IEEE, 2018.

[33] Qiuqiang Kong, Yong Xu, Wenwu Wang, Philip J. B. Jackson, and Mark D. Plumbley. Single-channel signal separation and deconvolution with generative adversarial networks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, page 2747–2753. AAAI Press, 2019.

[34] Vivek Narayanaswamy, Jayaraman J Thiagarajan, Rushil Anirudh, and Andreas Spanias. Unsupervised audio source separation using generative priors. In *INTERSPEECH*, 2020.

[35] Vivek Jayaram and John Thickstun. Source separation with deep generative priors. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[36] Ge Zhu, Jordan Darefsky, Fei Jiang, Anton Selitskiy, and Zhiyao Duan. Music source separation with generative flow. *IEEE Signal Processing Letters*, 29:2288–2292, 2022.

[37] Robin Scheibler, Youna Ji, Soo-Whan Chung, Jaeuk Byun, Soyeon Choe, and Min-Seok Choi. Diffusion-based generative speech source separation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[38] Bo Chen, Chao Wu, and Wenbin Zhao. Sepdiff: Speech separation based on denoising diffusion model. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[39] Joan Serrà, Santiago Pascual, Jordi Pons, R Oguz Araz, and Davide Scaini. Universal speech enhancement with score-based diffusion. *arXiv preprint arXiv:2206.03065*, 2022.

[40] Yen-Ju Lu, Zhong-Qiu Wang, Shinji Watanabe, Alexander Richard, Cheng Yu, and Yu Tsao. Conditional diffusion probabilistic model for speech enhancement. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7402–7406. Ieee, 2022.

[41] Hao Yen, François G Germain, Gordon Wichern, and Jonathan Le Roux. Cold diffusion for speech enhancement. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[42] Julius Richter, Simon Welker, Jean-Marie Lemercier, Bunlong Lay, and Timo Gerkmann. Speech enhancement and dereverberation with diffusion-based generative models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:2351–2364, 2023.

[43] Tsubasa Ochiai, Marc Delcroix, Yuma Koizumi, Hiroaki Ito, Keisuke Kinoshita, and Shoko Araki. Listen to what you want: Neural network-based universal sound selector. In *INTERSPEECH*, 2020.

[44] Marc Delcroix, Jorge Bennasar Vázquez, Tsubasa Ochiai, Keisuke Kinoshita, Yasunori Ohishi, and Shoko Araki. Soundbeam: Target sound extraction conditioned on sound-class labels and enrollment clues for increased performance and continuous learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:121–136, 2022.

[45] Bandhav Veluri, Justin Chan, Malek Itani, Tuochao Chen, Takuya Yoshioka, and Shyamnath Gollakota. Real-time target sound extraction. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[46] Hang Zhao, Chuang Gan, Andrew Rouditchenko, Carl Vondrick, Josh McDermott, and Antonio Torralba. The sound of pixels. In *Proceedings of the European conference on computer vision (ECCV)*, pages 570–586, 2018.

[47] Hao-Wen Dong, Naoya Takahashi, Yuki Mitsufuji, Julian McAuley, and Taylor Berg-Kirkpatrick. Clipsep: Learning text-queried sound separation with noisy unlabeled videos. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2023.

[48] Jie Hwan Lee, Hyeong-Seok Choi, and Kyogu Lee. Audio query-based music source separation. In *International Society for Music Information Retrieval Conference*, 2019.

[49] Qiuqiang Kong, Ke Chen, Haohe Liu, Xingjian Du, Taylor Berg-Kirkpatrick, Shlomo Dubnov, and Mark D Plumbley. Universal source separation with weakly labelled data. *arXiv preprint arXiv:2305.07447*, 2023.

[50] Kevin Kilgour, Beat Gfeller, Qingqing Huang, Aren Jansen, Scott Wisdom, and Marco Tagliasacchi. Text-driven separation of arbitrary sounds. In *INTERSPEECH*, 2022.

[51] Xubo Liu, Haohe Liu, Qiuqiang Kong, Xinhao Mei, Jinzheng Zhao, Qiushi Huang, Mark D Plumbley, and Wenwu Wang. Separate what you describe: Language-queried audio source separation. *arXiv preprint arXiv:2203.15147*, 2022.

[52] Hao Ma, Zhiyuan Peng, Xu Li, Mingjie Shao, Xixin Wu, and Ju Liu. Clapsep: Leveraging contrastive pre-trained model for multi-modal query-conditioned target sound extraction. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2024.

[53] Hao Ma, Zhiyuan Peng, Xu Li, Yukai Li, Mingjie Shao, Qiuqiang Kong, and Ju Liu. Language-queried target sound extraction without parallel training data. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2025.

[54] Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, et al. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[55] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. SampleRNN: An unconditional end-to-end neural audio generation model. In *International Conference on Learning Representations*, 2017.

[56] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. In *International Conference on Learning Representations*, 2019.

[57] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. GANSynth: Adversarial neural audio synthesis. In *International Conference on Learning Representations*, 2019.

[58] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in neural information processing systems*, volume 30, 2017.

[59] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

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

[61] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. SoundStream: an end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2021.

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

[63] Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. High-fidelity audio compression with improved rvqgan. *Advances in Neural Information Processing Systems*, 36:27980–27993, 2023.

[64] 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.

[65] Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. Audiogen: Textually guided audio generation. In *The Eleventh International Conference on Learning Representations*, 2023.

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

[67] 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.

[68] 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*, 2024.

[69] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11315–11325, 2022.

[70] Hugo Flores Garcia, Prem Seetharaman, Rithesh Kumar, and Bryan Pardo. Vampnet: Music generation via masked acoustic token modeling. In *International Society for Music Information Retrieval Conference*, 2023.

[71] Marco Comunità, Zhi Zhong, Akira Takahashi, Shiqi Yang, Mengjie Zhao, Koichi Saito, Yukara Ikemiya, Takashi Shibuya, Shusuke Takahashi, and Yuki Mitsufuji. Specmaskgit: Masked generative modeling of audio spectrograms for efficient audio synthesis and beyond. In *International Society for Music Information Retrieval Conference*, 2024.

[72] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021.

[73] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. In *International Conference on Learning Representations*, 2021.

[74] 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.

[75] 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, 2024.

[76] 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*, pages 10684–10695, 2022.

[77] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D. Plumbley. Audioldm: text-to-audio generation with latent diffusion models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23, 2023.

[78] 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*, 2024.

[79] 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, 2023.

[80] 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 framework with latent diffusion models. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24, 2024.

[81] Julian D Parker, Janne Spijkervet, Katerina Kosta, Furkan Yesiler, Boris Kuznetsov, Ju-Chiang Wang, Matt Avent, Jitong Chen, and Duc Le. Stemgen: A music generation model that listens. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1116–1120. IEEE, 2024.

[82] Yao Yao, Peike Li, Boyu Chen, and Alex Wang. Jen-1 composer: A unified framework for high-fidelity multi-track music generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 14459–14467, 2025.

[83] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35: 26565–26577, 2022.

[84] Zhongweiyang Xu, Debottam Dutta, Yu-Lin Wei, and Romit Roy Choudhury. Multi-source music generation with latent diffusion. *arXiv preprint arXiv:2409.06190*, 2024.

[85] Tornike Karchkhadze, Mohammad Rasool Izadi, Ke Chen, Gerard Assayag, and Shlomo Dubnov. Multi-track musicldm: Towards versatile music generation with latent diffusion model. *arXiv preprint arXiv:2409.02845*, 2024.

[86] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11461–11471, 2022.

[87] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[88] Ciprian Corneanu, Raghudeep Gadde, and Aleix M Martinez. Latentpaint: Image inpainting in latent space with diffusion models. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 4334–4343, 2024.

[89] Sora Kim, Sungho Suh, and Minsik Lee. Rad: Region-aware diffusion models for image inpainting. *arXiv preprint arXiv:2412.09191*, 2024.

[90] Tsiry Mayet, Pourya Shamsolmoali, Simon Bernard, Eric Granger, Romain HÉRAULT, and Clement Chatelain. TD-paint: Faster diffusion inpainting through time aware pixel conditioning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[91] Colin Raffel. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Columbia University, 2016.

[92] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[93] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[94] Timo Lüddecke and Alexander Ecker. Image segmentation using text and image prompts. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7086–7096, 2022.

[95] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[96] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017.

[97] Victor Weixin Liang, Yuhui Zhang, Yongchan Kwon, Serena Yeung, and James Y Zou. Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning. *Advances in Neural Information Processing Systems*, 35:17612–17625, 2022.

[98] Emilian Postolache, Giorgio Mariani, Michele Mancusi, Andrea Santilli, Luca Cosmo, and Emanuele Rodolà. Latent autoregressive source separation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9444–9452, 2023.

[99] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10696–10706, 2022.

[100] Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

[101] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, 2024.

[102] Marco Pasini, Stefan Lattner, and George Fazekas. Music2Latent: Consistency autoencoders for latent audio compression. *International Society for Music Information Retrieval Conference*, 2024.

[103] Marco Pasini, Stefan Lattner, and György Fazekas. Music2Latent2: Audio compression with summary embeddings and autoregressive decoding. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2025.

[104] Junghyun Koo, Yunkee Chae, Chang-Bin Jeon, and Kyogu Lee. Self-refining of pseudo labels for music source separation with noisy labeled data. *International Society for Music Information Retrieval Conference*, 2023.

[105] Dmitry Bogdanov, Minz Won, Philip Tovstogan, Alastair Porter, and Xavier Serra. The mtg-jamendo dataset for automatic music tagging. In *Machine Learning for Music Discovery Workshop, International Conference on Machine Learning (ICML 2019)*, Long Beach, CA, United States, 2019. URL `http://hdl.handle.net/10230/42015`.

[106] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. Fma: A dataset for music analysis. In *International Society for Music Information Retrieval Conference*, 2016.

# A Related Work

**Audio Source Separation and Extraction.** Audio source separation aims to decompose a polyphonic mixture into its constituent tracks, while source extraction focuses on isolating a particular target sound, often guided by metadata, text prompts, or reference examples. Two dominant paradigms have emerged: discriminative models learn a direct mapping from the input mixture to each target stem via regression losses, either in the waveform domain or on spectrogram representations [26–31]. In contrast, generative approaches learn probabilistic priors over source distributions and recover individual stems via sampling [32–36].

Recently, diffusion-based techniques have emerged as a powerful paradigm for audio decomposition, achieving strong results in both speech separation [37, 38] and enhancement [39–42]. These methods iteratively denoise a mixture under a learned score function, offering flexible and high-fidelity source recovery.

Query-based extraction further extends separation by conditioning the model on external cues such as class labels [43–45], visual signals [46, 47], or audio exemplars [48, 49]. Several studies have also demonstrated the effectiveness of natural language prompts for flexible, user-driven source isolation [23, 47, 50–53]. In our framework, we employ the pretrained CLAP model [17] to obtain shared audio-text embeddings, enabling seamless, language-guided extraction of arbitrary stems within a multi-track latent diffusion architecture.

**Audio Generation Models.** Early neural audio synthesis methods focused on autoregressive architectures that model waveform dependencies sample by sample. WaveNet [54] demonstrated the effectiveness of dilated convolutions for end-to-end generation, while SampleRNN [55] extended this with hierarchical recurrence. Subsequent work adopted adversarial objectives to improve fidelity, using GANs to generate perceptually sharp outputs [56, 57].

Parallel efforts introduced discrete-token models, where audio is encoded into compact code sequences using vector quantization (e.g., VQ-VAE [58]). Jukebox [59] models long-range dependencies over codes using Transformers [60], while recent systems enhance fidelity through residual quantization [61–63] and hierarchical token modeling, where coarse-to-fine code representations are generated over multiple levels [25, 64–66]. MusicGen [67] improves decoding efficiency with delayed-token generation, and Instruct-MusicGen [68] extends it for targeted editing via instruction-tuned prompts. Concurrently, token-based masked generative modeling techniques—originally developed for the vision domain [69]—have been extended to audio, enabling efficient non-autoregressive synthesis and precise spectrogram inpainting [70, 71].

Diffusion-based generation emerged with DiffWave [72] and WaveGrad [73], which learn to iteratively denoise Gaussian-corrupted waveforms. These techniques have since been adapted for music-specific generation with structure and style conditioning [74, 75]. Latent diffusion models (LDMs) [76], which perform denoising in a compressed embedding space, have further advanced generation fidelity and scalability. LDM-based audio models such as AudioLDM [77, 78], MusicLDM [1], and Stable Audio [2–4] achieve state-of-the-art performance. Recent frameworks like AUDIT [79], InstructME [80] explore the use of diffusion for controllable and interactive audio editing.

**Multi-Track Music Audio Modeling.** Recent studies model multi-track music as a structured composition of interdependent stems. StemGen [81] employs an iterative, non-autoregressive transformer over discrete tokens to generate stems conditioned on text prompts. Jen-1 Composer [82] applies latent diffusion to jointly model four canonical stems (bass, drums, instrument, melody), producing coherent multi-track compositions. MusicGen-Stem [81] combines per-stem vector quantization with an autoregressive decoder to synthesize bass, drums, and aggregated `other` components, and supports mixture-conditioned accompaniment generation.

Other work explores joint modeling of synthesis and decomposition within a single diffusion backbone. Multi-Source Diffusion Models (MSDM) [10] model a fixed set of stems (`bass`, `drums`, `guitar`, and `piano`) within a shared diffusion framework, relying on an additive mixture assumption and a Dirac delta-based posterior sampler, following the EDM formulation for ODE-based sampling [83]. This line of work has since been extended in GMSDI [11], MSG-LD [12], and others [84, 85]. GMSDI enables variable-stem modeling and text-based conditioning but remains grounded in waveform-space additive mixing. MSG-LD adapts latent diffusion for four-stem modeling and classifier-free guidance [18], though it still assumes fixed instrument classes.

In contrast, our approach jointly models mixture, submixture, and source embeddings in latent space and casts both synthesis and arbitrary-source extraction as text-conditioned inpainting tasks, offering fully class-agnostic multi-track music processing without reliance on fixed instrument vocabularies or linear mixing assumptions.

# B Background

## B.1 Canonical Inpainting in Score-Based Models

The core idea behind inpainting in score-based generative models is to estimate the score of the unknown region conditioned on known region [10, 20].

Let K denote the set of all tracks. Suppose a subset $\Omega \subset K$ is observed (i.e., known), and let $\Gamma = K \backslash \Omega$ denote the complement, i.e., the unobserved tracks we aim to inpaint. Define $\mathbf{z}^{\Omega} := \{z^{(k)}\}_{k \in \Omega}$ and $\mathbf{z}^{\Gamma} := \{z^{(k)}\}_{k \in \Gamma}$. The goal is to approximate the conditional score:

$$\nabla_{\mathbf{z}_{\tau}^{\Gamma}} \log q_{\tau}(\mathbf{z}_{\tau}^{\Gamma} | \mathbf{z}_{0}^{\Omega}). \tag{6}$$

This conditional gradient is generally intractable for a score model trained only on joint marginals. However, following Song et al. [20], we can approximate it via:

$$\begin{align} q_{\tau}(\mathbf{z}_{\tau}^{\Gamma} | \mathbf{z}_{0}^{\Omega}) &= \int q_{\tau}(\mathbf{z}_{\tau}^{\Gamma}, \mathbf{z}_{\tau}^{\Omega} | \mathbf{z}_{0}^{\Omega}) d\mathbf{z}_{\tau}^{\Omega} \nonumber \\ &= \int q_{\tau}(\mathbf{z}_{\tau}^{\Gamma} | \mathbf{z}_{\tau}^{\Omega}, \mathbf{z}_{0}^{\Omega}) q_{\tau}(\mathbf{z}_{\tau}^{\Omega} | \mathbf{z}_{0}^{\Omega}) d\mathbf{z}_{\tau}^{\Omega} \nonumber \\ &= \mathbb{E}_{q_{\tau}(\mathbf{z}_{\tau}^{\Omega} | \mathbf{z}_{0}^{\Omega})} \left[ q_{\tau}(\mathbf{z}_{\tau}^{\Gamma} | \mathbf{z}_{\tau}^{\Omega}, \mathbf{z}_{0}^{\Omega}) \right] \nonumber \\ &\approx \mathbb{E}_{q_{\tau}(\mathbf{z}_{\tau}^{\Omega} | \mathbf{z}_{0}^{\Omega})} \left[ q_{\tau}(\mathbf{z}_{\tau}^{\Gamma} | \mathbf{z}_{\tau}^{\Omega}) \right] \tag{7} \\ &\approx q_{\tau}(\mathbf{z}_{\tau}^{\Gamma} | \hat{\mathbf{z}}_{\tau}^{\Omega}), \tag{8} \end{align}$$

where $\hat{\mathbf{z}}_{\tau}^{\Omega} \sim q_{\tau}(\mathbf{z}_{\tau}^{\Omega} | \mathbf{z}_{0}^{\Omega}) = \mathcal{N}(\mathbf{z}_{\tau}^{\Omega}; \alpha_{\tau} \mathbf{z}_{0}^{\Omega}, \beta_{\tau}^{2} \mathbf{I})$ is a noised sample of the known region. Accordingly, the conditional score can be approximated as:

$$\begin{aligned} \nabla_{\mathbf{z}_{\tau}^{\Gamma}} \log q_{\tau}(\mathbf{z}_{\tau}^{\Gamma} | \mathbf{z}_{0}^{\Omega}) &\approx \nabla_{\mathbf{z}_{\tau}^{\Gamma}} \log q_{\tau}(\mathbf{z}_{\tau}^{\Gamma} | \hat{\mathbf{z}}_{\tau}^{\Omega}) \\ &= \nabla_{\mathbf{z}_{\tau}^{\Gamma}} \log q_{\tau}([\mathbf{z}_{\tau}^{\Gamma}; \hat{\mathbf{z}}_{\tau}^{\Omega}]), \end{aligned}$$

where $[\mathbf{z}_{\tau}^{\Gamma}; \hat{\mathbf{z}}_{\tau}^{\Omega}]$ denotes a composite latent vector such that the known region is replaced by $\hat{\mathbf{z}}_{\tau}^{\Omega}$ while the unknown region remains as $\mathbf{z}_{\tau}^{\Gamma}$, adopting the same notation as Song et al. [20].

This approximation enables zero-shot inpainting without requiring retraining: at each diffusion timestep, a noised version of the known latents is sampled, concatenated with the current estimate of the unknown latents, and passed to the score model. The resulting gradient is then applied to update only the unknown region. This process is repeated throughout the reverse diffusion trajectory.

## B.2 RePaint

Lugmayr et al. proposed *RePaint* [86], a resampling-based mechanism that improves score-based inpainting by repeating the diffusion process across multiple forward–reverse cycles. Their key insight is that, in conventional inpainting (as described in Equation (8)), the sampled noise for the known region is independent of the generated (inpainted) region. This lack of synchronization can lead to semantic inconsistencies and disharmony between known and unknown parts of the sample.

To address this, RePaint introduces a resampling mechanism during generation. At each denoising timestep, the algorithm alternates between one reverse diffusion step and one forward diffusion step, repeating this cycle $U$ times. These micro-steps refine the sampling distribution and can have the effect of partially marginalizing over the known region at noise level $\tau$ in Equation (8), thereby reducing the approximation error inherent in conditional score estimation. This iterative resampling procedure improves consistency but incurs higher computational cost, as each denoising step requires multiple forward–reverse passes–making RePaint significantly more expensive than standard inpainting methods.

**Algorithm 1** Inpainting using the RePaint approach.

```
Input:
    Number of timesteps $T$;
    Re-denoising steps per reverse step $U$;
    Noise schedule $\{\tau_i\}_{i=0}^{T}$ with $\alpha_\tau$, $\beta_\tau$;
    Binary mask $m$ (1 for known, 0 for unknown);
    Known clean (masked) latents $\mathbf{z}_0^{\text{known}}$;
    Denoiser network $f_\theta$
$\mathbf{z}_{\tau_T} \sim \mathcal{N}(\mathbf{0}, I)$
for $i = T, \ldots, 1$ do
    for $u = 1, \ldots, U$ do
        ▷ DDIM sampling step
        $\hat{\boldsymbol{v}}_{\tau_i} \leftarrow f_\theta(\mathbf{z}_{\tau_i}, \tau_i)$
        $\hat{\mathbf{z}}_0 \leftarrow \alpha_{\tau_i}\mathbf{z}_{\tau_i} - \beta_{\tau_i}\hat{\boldsymbol{v}}_{\tau_i}$
        $\hat{\boldsymbol{\epsilon}} \leftarrow \beta_{\tau_i}\mathbf{z}_{\tau_i} + \alpha_{\tau_i}\hat{\boldsymbol{v}}_{\tau_i}$
        $\hat{\mathbf{z}}_{\tau_{i-1}}^{\text{unknown}} \leftarrow \alpha_{\tau_{i-1}}\hat{\mathbf{z}}_0 + \beta_{\tau_{i-1}}\hat{\boldsymbol{\epsilon}}$
        ▷ Sample the known regions
        $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I)$ if $i > 1$, else 0
        $\mathbf{z}_{\tau_{i-1}}^{\text{known}} \leftarrow \alpha_{\tau_{i-1}}\mathbf{z}_0^{\text{known}} + \beta_{\tau_{i-1}}\boldsymbol{\epsilon}$
        ▷ Combine known and generated regions
        $\mathbf{z}_{\tau_{i-1}} \leftarrow m \odot \mathbf{z}_{\tau_{i-1}}^{\text{known}} + (1-m) \odot \hat{\mathbf{z}}_{\tau_{i-1}}^{\text{unknown}}$
        ▷ Reapply forward process
        if $u < U$ and $t > 1$ then
            $\mathbf{z}_{\tau_i} \sim \mathcal{N}\left(\frac{\alpha_{\tau_i}}{\alpha_{\tau_{i-1}}}\mathbf{z}_{\tau_{i-1}}, \left(1 - \frac{\alpha_{\tau_i}^2}{\alpha_{\tau_{i-1}}^2}\right) I\right)$
        end if
    end for
end for
return $\mathbf{z}_{\tau_0}$
```

While originally proposed for DDPM-based models, RePaint can be adapted to velocity-based objectives as used in our framework. We apply this adaptation in the sampling procedure described in Algorithm 1. Setting the resampling count $U = 1$ recovers the canonical single-sample inpainting method described in Appendix B.1.

## C Track-aware Inpainting Model with Adaptive Timesteps

Conventional diffusion-based inpainting methods apply a uniform noise schedule across both observed and missing regions, failing to account for their differing uncertainty characteristics [19, 20, 86–88]. In the standard setup, a denoising model $f_\theta(\mathbf{z}_\tau, \tau)$ is trained to approximate the joint score of a perturbed latent variable.

Recently, region-aware adaptations of diffusion inpainting – such as spatially varying noise schedules [89] and per-pixel timestep conditioning in TD-Paint [90] – have demonstrated substantial improvements in semantic consistency by preserving fidelity in observed regions. Inspired by TD-Paint, we extend this idea to three-track music audio by assigning distinct timestep conditions to each track, thereby improving inpainting quality in the latent space.

We describe our track-wise adaptive timestep conditioned model using general notation. Let $K$ be a set of tracks, and let $N = |K|$ be the number of tracks. Define the clean latent tensor and corresponding noise levels as:

$$\mathbf{z}_0 = (z_0^{(k)})_{k \in K} \in \mathbb{R}^{N \times C \times L}, \quad \boldsymbol{\tau} = (\tau_k)_{k \in K} \in [\tau_{\text{min}}, 1]^N,$$

where each $\tau_k$ is either zero (for observed tracks) or equal to a shared sample $\tau \sim \mathcal{U}([\tau_{\text{min}}, 1])$, depending on the inpainting configuration.

We define the track-wise product between a vector $x_{\boldsymbol{\tau}} \in \mathbb{R}^N$ and a latent tensor $\mathbf{z} \in \mathbb{R}^{N \times C \times L}$ as:

$$x_{\boldsymbol{\tau}} \odot \mathbf{z} := (x_{\boldsymbol{\tau}_k} z^{(k)})_{k \in K},$$

**Algorithm 2** Inpainting using adaptive timestep approach

```
Input:
    Number of timesteps $T$;
    Re-denoising steps per reverse step $U$;
    Noise schedule $\{\tau_i\}_{i=0}^{T}$ with $\alpha_\tau$, $\beta_\tau$;
    Binary mask $m$ (1 for known, 0 for unknown);
    Known clean (masked) latents $\mathbf{z}_0^{\text{known}}$;
    Denoiser network $f_\theta$
$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
$\boldsymbol{\tau}_T \leftarrow \tau_T(1-m)$
$\mathbf{z}_{\boldsymbol{\tau}_T} = m\mathbf{z}_0^{\text{known}} + (1-m)\boldsymbol{\epsilon}$
for $i = T, \ldots, 1$ do
                                        ▷ Partial DDIM sampling over unknown region
    $\hat{\boldsymbol{v}}_{\boldsymbol{\tau}_i} \leftarrow f_\theta(\mathbf{z}_{\boldsymbol{\tau}_i}, \boldsymbol{\tau}_i)$
    $\hat{\mathbf{z}}_0 \leftarrow \alpha_{\tau_i}\mathbf{z}_{\boldsymbol{\tau}_i} - \beta_{\tau_i}\hat{\boldsymbol{v}}_{\boldsymbol{\tau}_i}$
    $\hat{\boldsymbol{\epsilon}} \leftarrow \beta_{\tau_i}\mathbf{z}_{\boldsymbol{\tau}_i} + \alpha_{\tau_i}\hat{\boldsymbol{v}}_{\boldsymbol{\tau}_i}$
    $\hat{\mathbf{z}}^{\text{unknown}}_{\boldsymbol{\tau}_{i-1}} \leftarrow \alpha_{\tau_{i-1}}\hat{\mathbf{z}}_0 + \beta_{\tau_{i-1}}\hat{\boldsymbol{\epsilon}}$
    $\boldsymbol{\tau}_{i-1} \leftarrow \tau_{i-1}(1-m)$
    $\mathbf{z}_{\boldsymbol{\tau}_{i-1}} \leftarrow m\mathbf{z}_0^{\text{known}} + (1-m)\hat{\mathbf{z}}^{\text{unknown}}_{\boldsymbol{\tau}_{i-1}}$
end for
return $\mathbf{z}_{\boldsymbol{\tau}_0}$
```

and extend this notation to the cosine noise schedule terms as:

$$\alpha_{\boldsymbol{\tau}} = (\alpha_{\tau_k})_{k \in K}, \quad \beta_{\boldsymbol{\tau}} = (\beta_{\tau_k})_{k \in K}.$$

We perturb the joint latent using track-wise noise:

$$\mathbf{z}_{\boldsymbol{\tau}} = \alpha_{\boldsymbol{\tau}} \odot \mathbf{z}_0 + \beta_{\boldsymbol{\tau}} \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{9}$$

where each track is independently scaled by its corresponding noise factor.

The denoiser $f_\theta(\mathbf{z}_{\boldsymbol{\tau}}, \boldsymbol{\tau})$ is trained to regress the velocity target under v-objective:

$$\boldsymbol{v}_{\boldsymbol{\tau}} = \alpha_{\boldsymbol{\tau}} \odot \boldsymbol{\epsilon} - \beta_{\boldsymbol{\tau}} \odot \mathbf{z}_0, \tag{10}$$

resulting in the following training loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{z}_0, \boldsymbol{\epsilon}, \boldsymbol{\tau}} || f_\theta(\mathbf{z}_{\boldsymbol{\tau}}, \boldsymbol{\tau}) - \boldsymbol{v}_{\boldsymbol{\tau}} ||_2^2. \tag{11}$$

In our setup, we use $N = 3$ with $K = \{m, u, s\}$, corresponding to the mixture, submixture, and source tracks, respectively. In practice, the loss is computed only over the unknown tracks.

During training, we first sample a noise level $\boldsymbol{\tau} \sim \mathcal{U}([\tau_{\text{min}}, 1])$, and set the per-track timestep vector $\boldsymbol{\tau} \in \mathbb{R}^3$ according to one of the following four patterns:

$$\boldsymbol{\tau} \in \{(\tau, \tau, \tau), (0, \tau, \tau), (\tau, 0, \tau), (\tau, \tau, 0)\},$$

where each configuration is selected randomly for each training step.

Under this conditioning strategy, the full-noise setting $\boldsymbol{\tau} = (\tau, \tau, \tau)$ corresponds to learning the standard joint score. In contrast, a "single-zero" pattern allows the model to learn conditional score functions for the unobserved tracks while treating the others as fixed observations.

For example when $(\tau_m, \tau_u, \tau_s) = (0, \tau, \tau)$, the model is trained to approximate the gradient:

$$\nabla_{(z_\tau^{(u)}, z_\tau^{(s)})} \log q_\tau(z_\tau^{(u)}, z_\tau^{(s)} | z_0^{(m)}). \tag{12}$$

At inference time, we clamp observed tracks by setting their noise levels to zero, and apply standard reverse diffusion updates to the remaining (missing) tracks.

### C.1 Interpretation from the Perspective of Score Approximation

Conventional inpainting approaches approximate the conditional score in Eq. (6) using a single sampled estimate of the known latents. This corresponds to a high-variance Monte Carlo estimate of the expectation over $q_\tau(\mathbf{z}_\tau^\Omega|\mathbf{z}_0^\Omega)$, which may lead to instability–especially at high noise levels.

By contrast, the adaptive timestep model–inspired by TD-Paint [90]–circumvents this marginalization by training the model to directly approximate the conditional score.

We assume $\mathbf{z}_0 = [\mathbf{z}_0^\Gamma; \mathbf{z}_0^\Omega]$ is a clean sample from the dataset. Then, using an alternative factorization:

$$\begin{aligned} q_\tau(\mathbf{z}_\tau^\Gamma|\mathbf{z}_0^\Omega) &= \int q_\tau(\mathbf{z}_\tau^\Gamma, \mathbf{x}_0^\Gamma|\mathbf{z}_0^\Omega) d\mathbf{x}_0^\Gamma \\ &= \int q_\tau(\mathbf{z}_\tau^\Gamma|\mathbf{x}_0^\Gamma, \mathbf{z}_0^\Omega) q(\mathbf{x}_0^\Gamma|\mathbf{z}_0^\Omega) d\mathbf{x}_0^\Gamma \\ &= \mathbb{E}_{q(\mathbf{x}_0^\Gamma|\mathbf{z}_0^\Omega)} \left[ q_\tau(\mathbf{z}_\tau^\Gamma|\mathbf{x}_0^\Gamma, \mathbf{z}_0^\Omega) \right] \qquad (13) \\ &\approx q_\tau(\mathbf{z}_\tau^\Gamma|\mathbf{z}_0^\Gamma, \mathbf{z}_0^\Omega) = q_\tau(\mathbf{z}_\tau^\Gamma|\mathbf{z}_0^\Gamma) \qquad (14) \\ &= \mathcal{N}(\mathbf{z}_\tau^\Gamma; \alpha_\tau \mathbf{z}_0^\Gamma, \beta_\tau^2 \mathbf{I}), \qquad (15) \end{aligned}$$

where the approximation assumes that $\mathbf{z}_0^\Gamma \sim q(\mathbf{x}_0^\Gamma|\mathbf{z}_0^\Omega)$ is available from the dataset. Unlike the marginalization-based approximation in Eq (8), this expression introduces no sampling noise during inference, thereby reducing variance.

From this, the conditional score can be written as:

$$\begin{aligned} \nabla_{\mathbf{z}_\tau^\Gamma} \log q_\tau(\mathbf{z}_\tau^\Gamma|\mathbf{z}_0^\Omega) &\approx \frac{\alpha_\tau \mathbf{z}_0^\Gamma - \mathbf{z}_\tau^\Gamma}{\beta^2} \qquad (16) \\ &\approx \frac{\alpha_\tau \hat{\mathbf{z}}_\theta(\mathbf{z}_\tau^\Gamma, \mathbf{z}_0^\Omega, \boldsymbol{\tau}^\Gamma) - \mathbf{z}_\tau^\Gamma}{\beta^2}, \qquad (17) \\ &= -\mathbf{z}_\tau^\Gamma - \frac{\alpha_\tau}{\beta_\tau} f_\theta(\mathbf{z}_\tau^\Gamma, \mathbf{z}_0^\Omega, \boldsymbol{\tau}^\Gamma)_\Gamma, \qquad (18) \end{aligned}$$

where $f_\theta(\cdot)_\Gamma$ denotes the output corresponding to the unknown region. The per-track timestep vector $\boldsymbol{\tau}^\Gamma$ is defined as:

$$\boldsymbol{\tau}_k^\Gamma = \begin{cases} \tau, & \text{if } k \in \Gamma \\ 0, & \text{if } k \in \Omega \end{cases} \quad \text{for each } k \in K. \qquad (19)$$

The model is trained using the velocity objective in Eq. (11), restricted to the unknown region. This allows the denoiser to explicitly learn the conditional score on $\mathbf{z}_\tau^\Gamma$, avoiding the need for the stochastic marginalization and improving accuracy in conditional inpainting tasks. The full sampling procedure is detailed in Algorithm 2.

## D Iterative Generation

In addition to the one-stage mixture generation described in Section 2.3, MGE-LDM also supports an iterative, stem-by-stem synthesis procedure. This approach constructs a full mixture by sequentially generating individual sources, leveraging the partial generation mechanism at each step.

Let $\mathcal{I} = \{c_i^{(s)}\}_i$ be an (ordered) set of CLAP embeddings corresponding to the desired instrument description. At the first iteration ($i = 1$), we generate an initial source latent $\hat{z}_1^{(s)}$ by sampling with the model conditioned only on the prompt $c_1^{(s)}$:

$$\tilde{z}^{(m)}, \tilde{z}^{(u)}, \hat{z}_1^{(s)} \sim p_\theta(z^{(m)}, z^{(u)}, z^{(s)}|\varnothing, \varnothing, c_1^{(s)}),$$

and set $z_1^{(u)} = \hat{z}_1^{(s)}$ as the initial submixture latent. For each subsequent iteration $i = 2, ..., |\mathcal{I}|$, we follow the iterative imputation strategy of partial generation, treating the current submixture as the accumulated sum of decoded sources from the previous steps.

Table 4: **Abbreviations of instrument stems.** The table lists all abbreviations used throughout the paper along side their corresponding full instrument labels, grouped by dataset.

| Abbr. | Common | | | | | Slakh2100 | | | | | | | | MoisesDB | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | B | D | G | P | V | Brs. | C.P. | Org. | Pipe | Reed | Str. | S.Lead | S.Pad | B.str | Perc. |
| **Inst.** | bass | drums | guitar | piano | vocals | brass | chromatic percussion | organ | pipe | reed | strings | synth lead | synth pad | bowed strings | percussion |

Finally, the full mixture waveform is constructed via decoding and summing the generated source latents:

$$x^{(m)} = \sum_{i=1}^{|\mathcal{I}|} D(\hat{z}_i^{(s)}).$$

A preliminary evaluation of this iterative procedure is presented in Appendix F

# E Experimental Setup

In this section, we outline our experimental protocol, including baseline models, datasets, and implementation details. All baseline results are re-evaluated using our test sets to ensure consistency with our experimental setup.

## E.1 Datasets

We train and evaluate on three multi-track music datasets: Slakh2100 [21], MUSDB18 [14], and MoisesDB [22]. Each dataset follows its predefined train/test split. We train our models on various dataset combinations to evaluate robustness under different source distributions and stem configurations. A summary of stem abbreviations is provided in Table 4.

**Slakh2100** is derived from the Lakh MIDI Dataset v0.1 [91] and contains synthesized tracks rendered with sample-based virtual instruments. It comprises 2100 songs divided into training (1500), validation (375), and test (225) splits, totaling approximately 145 hours of audio. It includes a wide variety of instrument classes (e.g., bass, drums, guitar, piano, strings, synth pad, etc.). We adopt the naming $\mathbf{S}_A$ to denote a subset containing only bass, drums, guitar, and piano–the four classes used by MSDM and MSG-LD–and $\mathbf{S}_B$ to denote the complementary subset of remaining stems. We follow the official dataset splits provided by Slakh2100 for training, validation, and testing.

**MUSDB18** consists of 150 real-world music recordings with four stems: drums, bass, other, and vocals. We use all 100 tracks from the official training split for training, and the 50-track test split for evaluation. The total dataset length is approximately 10 hours.

**MoisesDB** comprises 240 songs (14 hours total) contributed by 47 artists across 12 genres. Each stem in the song is annotated with a two-tier stem taxonomy. Each track is decomposed into its constituent sources and annotated using a two-level hierarchical taxonomy of stem classes. We aggregate all second-level tracks into their corresponding top-level class. Among the 11 stem classes, we evaluate only the 7 unambiguous stems (e.g., bass, percussion, vocals, etc.). For evaluation, we randomly sample 24 tracks (10%) as the test set and use the remaining tracks for training.

**Data Construction.** We train our model using randomly constructed 3-track tuples (mix, sub, src). A source stem is randomly selected from the available stems, and the remaining stems are aggregated into a submixture. We select non-silent segments from the source track whenever possible, allowing up to 10 random resampling attempts per instance. The same temporal offset is applied across all stems to ensure alignment. For generation evaluation, we sample 300 random segments per test set. For source extraction, we sample between 150 and 700 non-silent segments per instrument class. All audio is downsampled to 16 kHz.

## E.2 Baselines

We use two recent multi-track diffusion models – MSDM [10] and MSG-LD [12] – as baselines, both of which operate on a fixed set of stems: bass, drums, guitar, and piano. In addition to generative and inpainting performance, we assess source extraction capabilities against Demucs [29], which separates the mixture into `bass`, `drums`, `other`, and `vocals` stems, and AudioSep [23], which performs text-conditioned separation based on natural language queries.

All baseline metrics are recomputed on our test splits for fair comparison, and baseline models are implemented as follows:

- MSDM [10]: We use the official implementation and pretrained checkpoint.[1] Since MSDM operates at 22 kHz, we upsample our 16 kHz test audio for inference and downsample the output back to 16 kHz.
- MSG-LD [12]: As no checkpoint is publicly released, we reproduce the model by retraining it from the official codebase.[2]
- HDemucs [26]: We train a 16 kHz version of Hybrid Demucs using the `demucs_lightning` implmentation.[3]
- AudioSep [23]: We evaluate using the publicly available implementation and checkpoint provided by the authors.[4] Since AudioSep operates at a sampling rate of 32 kHz, we upsample all test audio from 16 kHz to 32 kHz before inference, and subsequently downsample the separated outputs back to 16 kHz for evaluation consistency.

### E.3 Implementation Details

Our models use the Stable Audio backbone [3], comprising an autoencoder and a DiT-based diffusion model. To better accommodate per-track variability in the joint latent space, we replace LayerNorm [92] with GroupNorm [93], using three groups to reflect the number of tracks.

To bridge the audio-text modality gap, we adopt stochastic linear interpolation between audio and text embeddings on the source track, following prior work on multimodal fusion [52, 94]. Concretely, we generate the prompt "*The sound of the* `{label}`" and compute the source conditioning vector $c^{(s)}$ as a convex combination of the CLAP text embedding and its corresponding audio embedding, where the interpolation weight $\alpha \sim \mathcal{U}([0, 1])$ is sampled randomly for each training example.

All of our models – except the one trained on the full dataset combination ($\mathbf{S}_A$ , $\mathbf{S}_B$ , $\mathbf{M}_u$ , $\mathbf{M}_o$ ) – are trained for 200K iterations with a batch size of 64, using 16 kHz audio segments of 10.24 seconds. The full combination model is trained for 320K iterations with a batch size of 128. During sampling and inpainting, we apply classifier-free guidance (CFG) with a guidance scale of 2.0 and a per-track dropout probability of $p = 0.1$. All diffusion-based samples – including those from baseline models – are generated using 250 inference steps. We adopt DDIM sampling [95] for all our models, while each baseline uses its originally proposed sampling method.

**Autoencoder.** We adopt the VAE-based architecture from Stable Audio [3], with a downsampling ratio of 2048, yielding a 7.8125 Hz latent resolution and 64 latent channels. We train the autoencoder on all training subsets from Slakh2100, MUSDB18, and MoisesDB using 16 kHz mono audio for 600K steps with batch size 16.

**Diffusion Model.** In practice, the three latent representations are concatenated along the channel dimension, such that the input to the diffusion model becomes *Concat*$[z^{(m)}, z^{(u)}, z^{(s)}] \in \mathbb{R}^{3C \times L}$. We use a DiT backbone [15] with 24 transformer blocks with 48 heads, and a projected latent dimension of 1536 (3 tracks × 512 each). Timestep embeddings are prepended to the input vector of the transformer. CLAP embeddings for each track are processed by independent projection layers (without weight sharing) to produce scale and shift parameters for AdaIN-style conditioning [96]. These are applied group-wise via GroupNorm within each DiT layer to modulate the corresponding track-specific activations. Text embeddings are obtained from CLAP [17] using the checkpoint `music_audioset_epoch_15_esc_90.14.pt` via the `laion-clap` library.[5] Our implementation builds upon the official `stable-audio-tools` repository from Stability AI[6] and the

[1] https://github.com/gladia-research-group/multi-source-diffusion-models

[2] https://github.com/karchkha/MSG-LD

[3] https://github.com/KinWaiCheuk/demucs_lightning

[4] https://github.com/Audio-AGI/AudioSep

[5] https://github.com/LAION-AI/CLAP

[6] https://github.com/Stability-AI/stable-audio-tools

Table 5: **Source extraction performance of RePaint-based methods vs. MGE-LDM.** Metrics are reported as Log-Mel L1 distance ↓. $T$ indicates the number of reverse timesteps, and $U$ specifies the number of denoising operations per reverse step (i.e., $U-1$ intermediate resampling steps). The case $U=1$ corresponds to the canonical single-sample conditional score approximation [20], as described in Appendix B.1. MGE-LDM uses adaptive timestep conditioning without resampling.

| **Model** | $T$ | $U$ | $\mathbf{S}_A$ | | | | | $\mathbf{S}_B$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | B | D | G | P | Avg. | Brs. | C.P. | Org. | Pipe | Reed | Str. | S.Lead | S.Pad | Avg. |
| MGE (ours) | 250 | 1 | **1.68** | 2.71 | **2.69** | **2.16** | **2.31** | **3.43** | **2.16** | **1.84** | **2.33** | **3.07** | **2.44** | **2.31** | **1.93** | **2.48** |
| RePaint [86] | 250 | 1 | 2.00 | 3.20 | 3.15 | 2.83 | 2.79 | 4.75 | 2.47 | 2.79 | 4.27 | 4.65 | 3.06 | 3.73 | 2.80 | 3.56 |
| | 125 | 2 | 1.89 | 2.75 | 2.91 | 2.68 | 2.55 | 4.33 | 2.37 | 6.67 | 3.84 | 4.27 | 2.82 | 3.64 | 2.58 | 3.81 |
| | 50 | 5 | 1.80 | 2.43 | 2.83 | 2.55 | 2.40 | 3.89 | 2.32 | 2.57 | 3.35 | 3.81 | 2.64 | 3.56 | 2.42 | 3.07 |
| | 25 | 10 | 1.77 | **2.28** | 2.80 | 2.51 | 2.34 | 3.79 | 2.31 | 2.50 | 3.15 | 3.71 | 2.66 | 3.44 | 2.40 | 2.99 |
| | 250 | 2 | 1.90 | 2.65 | 2.94 | 2.65 | 2.53 | 4.23 | 2.41 | 2.71 | 3.82 | 4.27 | 2.84 | 3.66 | 2.59 | 3.31 |
| | 250 | 4 | 1.79 | 2.34 | 2.83 | 2.59 | 2.38 | 3.95 | 2.34 | 2.66 | 3.42 | 3.88 | 2.69 | 3.55 | 2.45 | 3.12 |

training framework from `friendly-stable-audio-tools`.[7] All models were trained on a single NVIDIA RTX 6000 GPU (48 GB memory).

# F Ablation Study

This section presents ablation experiments designed to further analyze key components of our framework. Unless otherwise specified, all models are trained on the combined $\mathbf{S}_A + \mathbf{S}_B$ dataset. Each of our models is evaluated with the same configuration as in Section 3, using $T = 250$ denoising steps during sampling.

## F.1 Comparison with Canonical Inpainting Methods

We assess the effectiveness of our adaptive timestep conditioning strategy by comparing it against two prior approaches: the canonical one-sample conditional score approximation (Appendix B.1) and the RePaint method [86] (Appendix B.2). Table 5 reports the results for the source extraction task.

In RePaint, $U$ denotes the number of denoising steps performed per reverse timestep: one denoising step followed by $U-1$ forward (resampling) steps. As a result, the total number of denoising steps becomes $T \times U$ during the full inpainting process. Note that setting $U = 1$ recovers the canonical single-sample estimator in Eq. (8).

We observe that, for a fixed number of denoising steps, using fewer timesteps $T$ with more resampling cycles $U$ generally improves performance, confirming observations in the original RePaint paper. We hypothesize that repeated resampling helps stabilize conditional generation by mitigating the noise mismatch between observed and unobserved regions, particularly at high noise levels, where observed latents contain little informative content and single-sample approximations of Eq. (8) become highly unreliable. While this approach does not yield a precise marginal score estimate, it heuristically improves inpainting quality through localized refinement.

Interestingly, we also observe that RePaint configurations with larger total denoising steps – such as $T = 250, U = 2$ and $T = 250, U = 4$ – consistently underperform compared to $T = 25, U = 10$, across all stems in both $\mathbf{S}_A$ and $\mathbf{S}_B$ . This suggests that, for inpainting tasks, accurately modeling the conditional score at each timestep is more critical than simply increasing the number of reverse steps. As RePaint approximates the conditional score by marginalizing over perturbed conditions via resampling, performance benefits are observed primarily through increased resampling ($U$), not longer trajectories ($T$).

Nevertheless, our adaptive timestep model outperforms all RePaint variants across both datasets, with the sole exception of `drums` in $\mathbf{S}_A$ . By directly learning track-specific conditional scores during training, our method eliminates the need for inference-time marginalization, resulting in lower variance and improved reconstruction quality.

[7] `https://github.com/yukara-ikemiya/friendly-stable-audio-tools`

Table 6: **Total generation performance across modeling variants.** Metrics are reported as FAD ↓. All models are trained on $\mathbf{S}_A + \mathbf{S}_B$ . The baseline model uses uniform (non-adaptive) timesteps across all tracks. MGE variants apply adaptive timestep conditioning and test the impact of normalizations and CFG dropout rates. Values in parentheses indicate generation conditioned on the text prompt "*The sound of the bass, drums, guitar, and piano*".

| **Model** | Testset | | | |
|---|---|---|---|---|
| | $\mathbf{S}_A$ | $\mathbf{S}_{\text{Full}}$ | $\mathbf{M}_u$ | $\mathbf{M}_o$ |
| Non-adaptive | 3.26 (2.00) | 0.79 | **5.12** | **4.51** |
| MGE (adaptive) | 3.14 (2.24) | 0.63 | 5.46 | 4.73 |
| - w/o GroupNorm | 3.48 (2.44) | 0.76 | 5.61 | 4.88 |
| - CFG dropout $p$=0.5 | **3.12** (2.67) | **0.58** | 5.43 | 4.82 |

Table 7: **Source extraction performance under different architectural and CFG settings.** Metrics are reported as Log-Mel L1 distance ↓. All models are trained on $\mathbf{S}_A + \mathbf{S}_B$ . GN and LN denote GroupNorm and LayerNorm, respectively. $p$ indicates the classifier-free guidance (CFG) dropout rate applied to each track's conditioning vector, and $s$ refers to the CFG guidance scale.

| Norm. | $p$ | $s$ | $\mathbf{S}_A$ | | | | | $\mathbf{S}_B$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | B | D | G | P | Avg. | Brs. | C.P. | Org. | Pipe | Reed | Str. | S.Lead | S.Pad | Avg. |
| GN | 0.1 | 2.0 | 1.68 | 2.71 | 2.69 | 2.16 | 2.31 | 3.43 | **2.16** | **1.84** | 2.33 | 3.07 | 2.44 | 2.31 | 1.93 | 2.43 |
| LN | 0.1 | 2.0 | **1.67** | 4.22 | 2.65 | 2.15 | 2.67 | 3.42 | 2.35 | 1.97 | 2.40 | 3.45 | 2.51 | 2.32 | 2.03 | 2.55 |
| GN | 0.5 | 2.0 | 1.78 | **1.96** | **2.62** | **1.96** | **2.08** | 3.37 | 2.22 | 1.97 | 2.36 | 2.89 | 2.44 | **2.07** | 1.89 | 2.40 |
| GN | 0.1 | 1.0 | 1.67 | 2.79 | 2.70 | 2.05 | 2.30 | **3.24** | 2.23 | 1.85 | **2.25** | **2.88** | 2.28 | 2.27 | **1.87** | **2.35** |
| GN | 0.1 | 4.0 | 1.77 | 2.49 | 2.79 | 2.27 | 2.33 | 3.64 | 2.17 | 1.91 | 2.42 | 3.20 | 2.66 | 2.31 | 2.06 | 2.54 |
| GN | 0.1 | 8.0 | 1.93 | 2.49 | 2.93 | 2.41 | 2.44 | 4.35 | 2.28 | 2.01 | 2.57 | 3.47 | **3.27** | 2.42 | 2.30 | 2.83 |

A potential concern is whether optimizing for adaptive timestep-conditional inference might degrade generation quality when using uniform timestep schedules across tracks. To assess this, we evaluate our adaptive timestep model with a uniform timestep vector $\boldsymbol{\tau} = (\tau, \tau, \tau)$, which corresponds to total generation task, and compare it to a baseline trained with non-adaptive, shared timesteps.

As shown in Table 6, comaprining non-adaptive uniform timstep basline model and our model, both models achieve comparable FAD scores, indicating that timestep adaptation preserves generation performance under uniform scheduling while providing significant advantages for inpainting tasks.

### F.2 Additional Design Ablations

We additionally investigate the impact of various modeling and training choices, including normalization strategies and classifier-free guidance (CFG) dropout rates.

Table 6 includes results from models trained with LayerNorm instead of GroupNorm, following the original DiT architecture, as well as a variant using a higher CFG dropout rate of $p = 0.5$. We observe that GroupNorm slightly outperforms LayerNorm across all test sets, supporting the use of track-wise normalization in our multi-track setting. Regarding CFG dropout, increasing the dropout rate improves unconditional generation performance, particularly on $\mathbf{S}_A$ and $\mathbf{S}_B$ . However, when conditioned on the text prompt (values in parentheses), the model trained with $p = 0.5$ performs worse, suggesting that overly aggressive dropout may impair semantic conditioning for total mixture generation.

We further examine how modeling and training design choices–such as normalization layers, classifier-free guidance (CFG) dropout probability, and CFG scale–affect extraction performance, and report the results in Table 7. When comparing normalization strategies, GroupNorm consistently matches or outperforms LayerNorm across most stems, demonstrating the advantage of modeling track-wise statistics in our multi-track architecture. This observation aligns with trends seen in mixture generation results. For CFG dropout, a higher dropout probability ($p = 0.5$) leads to improved performance compared to the default $p = 0.1$, suggesting that stronger stochastic conditioning is beneficial during source extraction. While this differs from the trend observed in mixture generation

Table 8: **Preliminary results for iterative stem-wise generation.** Metrics are reported as FAD ↓. Evaluation is conducted using a model trained exclusively on $\mathbf{S}_A$ . Each column header indicates the generation order of stems (e.g., BDGP denotes `bass`→ `drums`→ `guitar`→ `piano`).

| | $\mathbf{S}_A$ | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Total Gen. | BDGP | BDPG | DBGP | DBPG | DGBP | DPGB | GPBD | GPDB | PDGB | PGBD |
| MGE | **0.47** | 0.60 | 0.66 | 0.78 | 0.75 | 0.70 | 0.77 | 0.61 | 0.66 | 0.63 | 0.60 |

(Table 6), the discrepancy may be explained by the fact that, in extraction, non-target tracks are effectively treated as unconditioned. This makes overall performance more sensitive to the model's ability to generalize in the presence of dropout. We also evaluate various CFG scales (1, 2, 4, 8). A scale of 1 yields the best performance overall, although scales 2 and 4 remain competitive. Performance degrades at scale 8, indicating that overly strong guidance can impair extraction quality.

### F.3 Iterative Generation Variants

Table 8 presents preliminary results for the iterative generation procedure described in Appendix D, applied to a model trained on $\mathbf{S}_A$ . The task involves sequentially generating the four canonical stems (`bass`, `drums`, `guitar`, and `piano`) in various orders.

Across all tested permutations, iterative generation produced higher FAD scores compared to one-stage mixture generation, indicating a degradation in perceptual quality. Nonetheless, iterative generation may offer utility in settings that require fine-grained, source-specific control.

An interesting trend observed: generation sequences that began with `drums`consistently resulted in poorer performance relative to other orderings. This suggests that the model may be more effective at first establishing harmonic or melodic content before aligning rhythmic elements. While this observation is speculative, it highlights a potential inductive bias in the model that warrants further investigation, particularly in scenarios beyond the four-instrument configuration.

## G Limitations

While MGE-LDM provides a flexible, class-agnostic framework for multi-track audio modeling, several limitations remain. First, all experiments are conducted using 16 kHz monaural audio, which constrains upper-frequency resolution and omits spatial cues, thereby limiting realism for high-fidelity or stereo music applications. Second, the model relies on CLAP-based semantic conditioning, which introduces a modality gap between text and audio [97]. This can occasionally lead to semantic drift during extraction

Third, although MGE-LDM reduces dependence on fixed instrument classes, it still requires multi-stem supervision during training. This dependency restricts applicability to fully unlabeled or large-scale web audio collections. Fourth, training on MUSDB18 alone with the same number of iterations as other configurations leads to overfitting, likely due to the limited duration (approximately 10 hours) of its training split. This highlights the challenge of achieving robust performance in low-resource multi-track settings.

Finally, our model is trained using triplets $(\textit{mix}, \textit{submix}, \textit{source})$ that satisfy $\textit{mix} = \textit{submix} + \textit{source}$ in waveform space; however, the latent diffusion process does not enforce an explicit additivity constraint for generated triplets. We believe this omission contributes directly to hallucination penomena, where the model extracts source absent from the mixture. Postolache et al. [98] addresses a related issue by enforcing additivity in a discrete VQ-VAQE latent space, estimating the joint likelihood of two sources by counting codebook co-occurrences — effectively modeling $p(z_{\text{mix}}|z_{\text{src}_1}, z_{\text{src}_2})$, where $z_*$ are quantized latent codes. Our current pipeline, however, operates in a continuous latent space, which precludes the direct use of such discrete bin-counting methods. Adapting this latent-domain likelihood formulation to continuous spaces, for example, by designing suitable regularizers or adopting a VQ-VAE-based encoder with discrete diffusion [99–101] or MaskGiT [69]-style generation, represents a promising direction for future work.

## H Future Work

Future extensions of MGE-LDM include scaling to higher-resolution formats such as 44.1 kHz stereo audio, enabling richer timbral detail and spatial fidelity. In particular, this can be achieved by leveraging high-quality latent representations recently developed for the music domain [102, 103]. To reduce the modality gap in text-conditioned extraction, fine-tuning on curated audio–text datasets like MusicCaps [66] is a promising direction. Given its minimal reliance on precise stem boundaries, MGE-LDM is naturally suited for incorporating weakly or noisily labeled multi-track data [22, 104], which may expand training diversity.

Another promising avenue is to pre-train MGE-LDM on large-scale mixture-only corpora such as MTG-Jamendo [105] or the Free Music Archive [106] to learn general audio priors for mixture tracks, followed by fine-tuning on multi-track datasets for source-aware generation. This two-stage training strategy is expected to enhance generative quality and improve generalization.

We also plan to extend MGE-LDM to text-based music editing tasks, drawing inspiration from recent instruction-guided frameworks such as AUDIT [79], InstructME [80], and Instruct-MusicGen [68]. Leveraging MGE-LDM's latent inpainting capabilities and language-conditioned generation, this extension could enable user-directed operations such as instrument replacement and style transformation via natural language prompts, building upon the model's unified training scheme and class-agnostic design.

## I Spectrogram Examples of Generated Samples

We present Mel-spectrogram visualizations of generated audio samples across the three primary tasks: total generation, partial generation (imputation), and source extraction. All examples are produced by MGE-LDM trained on the combined Slakh2100 ($\mathbf{S}_A + \mathbf{S}_B$), MUSDB18 ($\mathbf{M}_u$), and MoisesDB ($\mathbf{M}_o$) datasets.

We note that the model is capable of generating `vocals` in the unconditional setting, as `vocals`stems are present in the training data. Although MGE-LDM does not currently support fine-grained control over vocal generation, this points to a promising direction for future work, such as incorporating explicit vocal prompts or segment-level control for more expressive and structured multi-track modeling.

## J Ethics Statement

This work introduces a class-agnostic generative framework for multi-track music modeling, trained exclusively on publicly available datasets (Slakh2100, MUSDB18, and MoisesDB). While the model enables flexible music generation, source imputation, and source extraction, it also carries potential risks, such as unauthorized manipulation, misuse in derivative content, or generation of audio resembling copyrighted material. To mitigate these concerns, we commit to releasing the model and code under a license with clear usage guidelines, emphasizing responsible research and ethical creative applications.

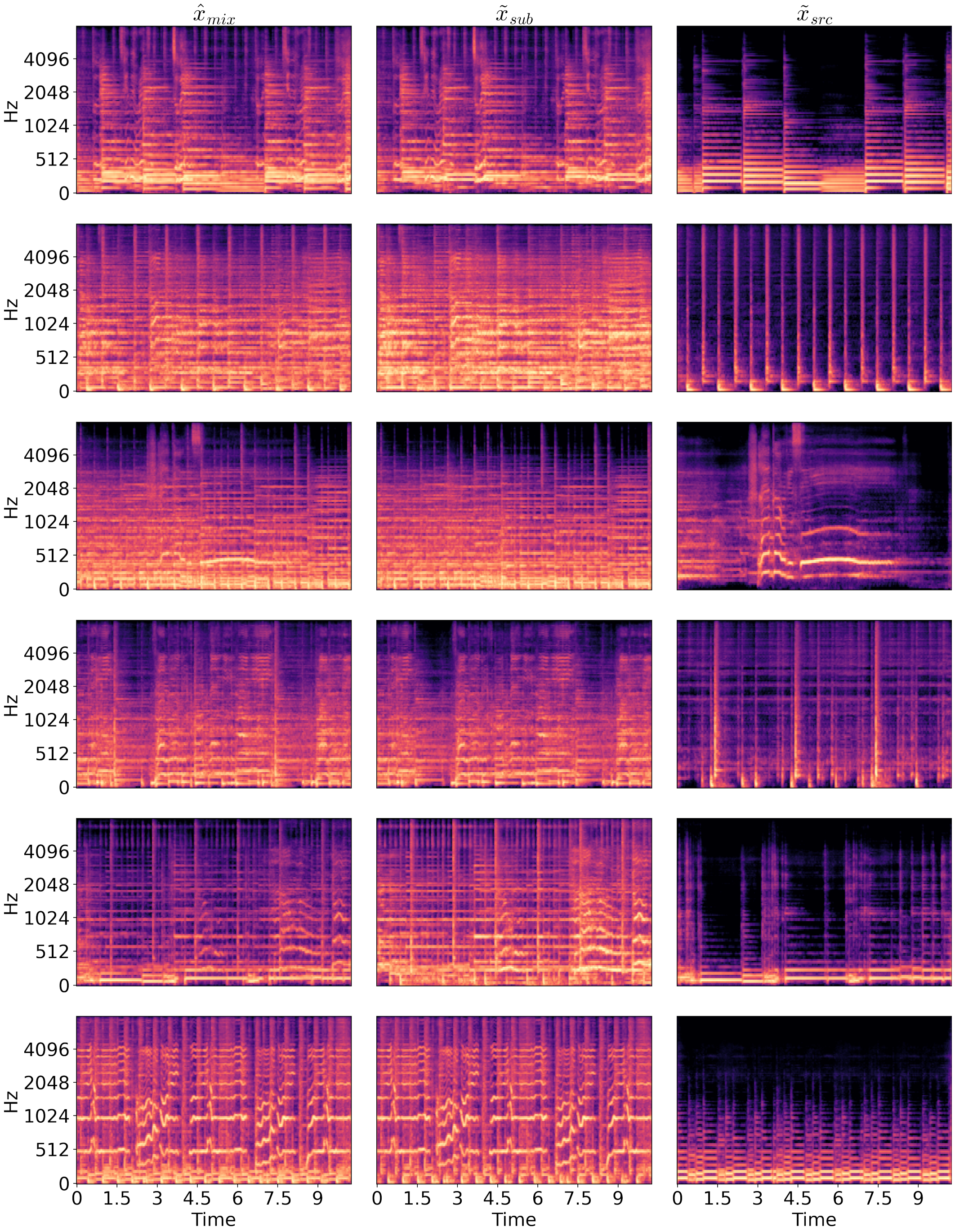


Figure 2: **Total generation examples.** Each sample displays Mel-spectrograms of the mixture, submixture, and source tracks, all generated simultaneously by MGE-LDM. The mixture track is used to evaluate the total generation output.

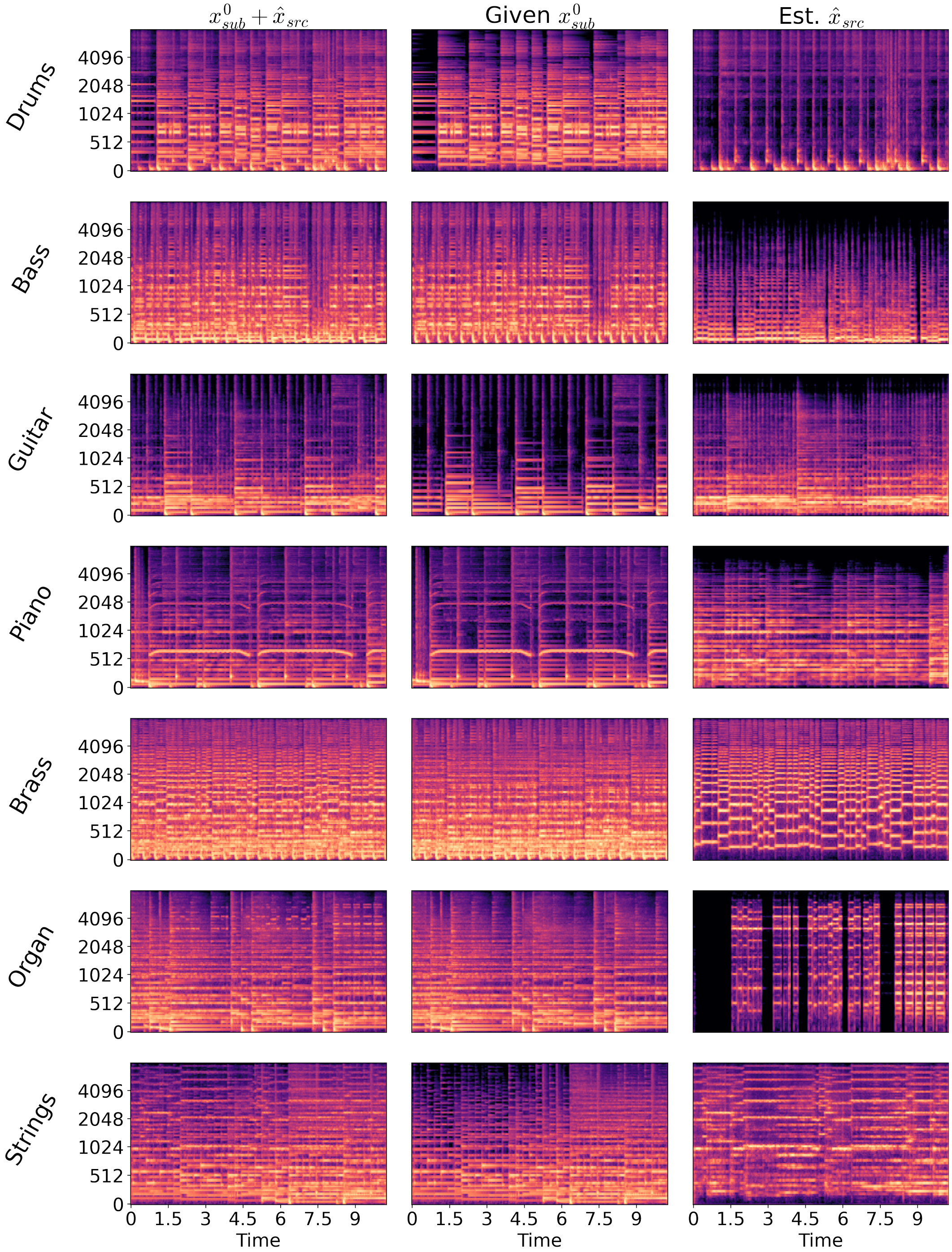


Figure 3: **Source imputation examples.** Each row illustrates source inpainting results by MGE-LDM, conditioned on the text prompt "*The sound of the* {`label`}". The middle column shows the provided context mixture (submix), the rightmost column is the generated source, and the leftmost column is the recombined mixture of the submix and generated source. While some stems are imputed accurately, others fail due to data imbalance during training.

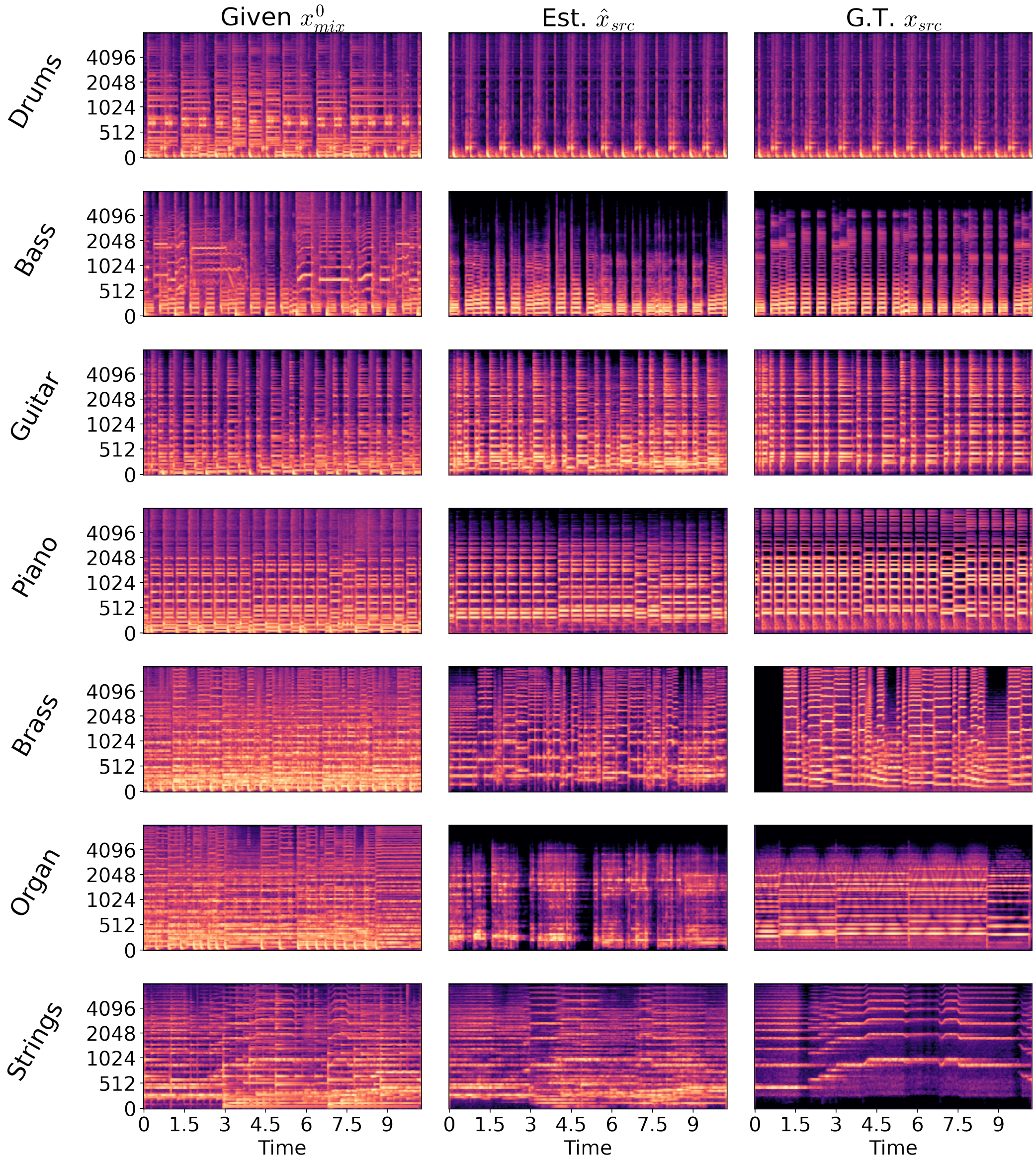


Figure 4: **Source extraction examples.** Source extraction results produced by MGE-LDM, conditioned on the text query "*The sound of the* `{label}`". The leftmost column shows the input mixture, the middle column is the extracted source predicted by the model, and the rightmost column is the ground-truth source. We observe that extraction quality may degrade for underrepresented classes such as strings, and in some cases, the model hallucinates unrelated instruments or incorrect timbres.