Multi-Source Music Generation with Latent Diffusion

Zhongweiyang Xu, Debottam Dutta, Yu-Lin Wei, Romit Roy Choudhury
Department of Electrical and Computer Engineering
University of Illinois Urbana-Champaign

Abstract

Most music generation models directly generate a single music mixture. To allow for more flexible and controllable generation, the Multi-Source Diffusion Model (MSDM) has been proposed to model music as a mixture of multiple instrumental sources (e.g. piano, drums, bass, and guitar). Its goal is to use one single diffusion model to generate mutually-coherent music sources, that are then mixed to form the music. Despite its capabilities, MSDM is unable to generate music with rich melodies and often generates empty sounds. Its waveform diffusion approach also introduces significant Gaussian noise artifacts that compromise audio quality. In response, we introduce a Multi-Source Latent Diffusion Model (MSLDM) that employs Variational Autoencoders (VAEs) to encode each instrumental source into a distinct latent representation. By training a VAE on all music sources, we efficiently capture each source's unique characteristics in a "source latent". The source latents are concatenated and our diffusion model learns this joint latent space. This approach significantly enhances the total and partial generation of music by leveraging the VAE's latent compression and noise-robustness. The compressed source latent also facilitates more efficient generation. Subjective listening tests and Fréchet Audio Distance (FAD) scores confirm that our model outperforms MSDM, showcasing its practical and enhanced applicability in music generation systems. We also emphasize that modeling sources is more effective than direct music mixture modeling. Codes and models are available at https://github.com/XZWY/MSLDM. Demos are available at https://xzwy.github.io/MSLDMDemo/.

1 Introduction

Generative models have shown impressive results not only in language and image generation OpenAI [2023], Ramesh et al. [2021], Brown et al. [2020], but also in music generation. Music generation models primarily fall into two categories: 1) Auto-regressive and 2) Diffusion-based. Auto-regressive models like WaveNet van den Oord et al. [2016] directly model scalar-quantized waveform samples but suffer from low sampling efficiency. Further research Dhariwal et al. [2020], Agostinelli et al. [2023], Copet et al. [2024] starts to auto-regressively model audio/speech/music tokens, which are often extracted from audio tokenizers van den Oord et al. [2018], Dhariwal et al. [2020], **?**], Kumar et al. [2023], Zeghidour et al. [2021].

Diffusion-based models also hold great potential. Noise2Music Huang et al. [2023a] generates either downsampled waveforms or Mel-Spectrograms as an intermediate step before decoding to full waveforms. Inspired by latent diffusion's success in images Rombach et al. [2022], models like DiffSound Yang et al. [2023] use spectrogram VQ-VAE tokenizers with discrete diffusion, while others Huang et al. [2023b], Liu et al. [2023, 2024], Chen et al. [2023], Schneider et al. [2023] leverage continuous latent spaces from spectrogram-based VAEs. In contrast, models like StableAudio Evans et al. [2024a] use waveform VAE latents as diffusion targets.

Moûsai Schneider et al. [2023] introduces a diffusion magnitude autoencoding (DMAE) approach to learn a spectrogram encoder, followed by diffusion modeling on the latent space. StableAudio2 Evans et al. [2024b] further enhances this by generating full songs using waveform VAE latents with diffusion.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

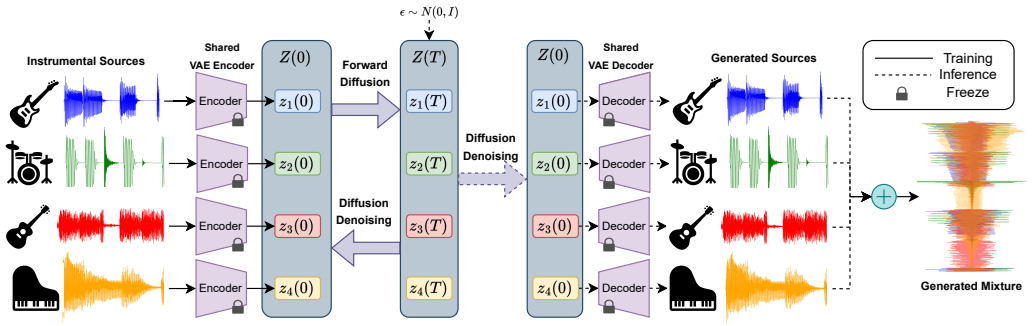

Figure 1: An Overview of the proposed MSLDM framework.

Although rapid progress has been made in music generation, most models directly model the whole music piece, which is a mixture of individual sources. However, the individual sources cannot be disentangled from the mixture. Ideally, a music generation method should be able to generate the individual music sources that together form a piece of music, similar to a human's music composition process. This will allow the music to be more interpretable and controllable (e.g., the piano can be made louder than the drums). To solve this problem, one class of methods directly models the musical notes or midi representations in a multi-track manner Mittal et al. [2021], Dong et al. [2017]. However, the generated notes or midi sequence need to be later decoded to a single waveform using synthesizers. The other type of research directly learns to model several music tracks directly. StemGen Parker et al. [2024] uses a masked language model on Encodec tokens to generate any single instrument source given a music context. Pasini et al. [2024] uses the latent diffusion model to generate bass companions conditioned on mixtures, while SingSong Donahue et al. [2023] generates background companions given the vocal source. Most recently, MSDM Mariani et al. [2024] has been proposed to simultaneously model four instrument sources (piano, drums, bass, guitar) with a single waveform-domain diffusion model, and GMSDI Postolache et al. [2024] has generalized MSDM by training on text-conditioned diffusion models allowing adaption to any music dataset using text descriptions.

In this paper, we propose to simultaneously model four different instrumental sources (piano, drums, bass, guitar) jointly, with a single multi-source latent diffusion model (MSLDM), as shown in Fig.1. We first train a shared SourceVAE on all the sources to perceptually compress the source audio, and then use this VAE's encoder to extract the latent feature of each source. We then apply diffusion to model the generation of the latents of the sources. We claim that 1) with the VAE compressing the source audio into a compact latent, the diffusion can better model semantic and sequential information like melodies and the harmony between sources, and 2) modeling individual sources is better than direct modeling mixtures. Our result in both subjective human evaluation and objective FADs validates our claim.

2 Models

Fig. 1 shows the training and inference pipeline of our model. Just like any latent diffusion model, our model involves two blocks: (1) SourceVAE, which is trained like a VAE but with an adversarial loss for perceptual compression. (2) A diffusion model simultaneously models all the sources' latents concatenated together as one latent. Inference includes two sub-tasks as well: (1) Total generation allows unconditional generation of all instrumental sources at the same time and (2) Partial generation allows the generation of companion sources given any combinations of instrumental sources (e.g., generate bass and guitar to accompany a given piece of piano and drums).

2.1 SourceVAE

The SourceVAE aims to compress waveform-domain instrumental sources into a compact latent space, while still ensuring perceptually indistinguishable reconstruction. This is usually achieved by adversarial training with carefully designed discriminators. We borrow this training framework and model architecture from the DAC Kumar et al. [2023] neural audio codec. DAC is a state-of-the-art waveform-domain neural audio codec trained with both reconstruction and adversarial losses. It can encode, quantize, and decode audio with superior quality. However, we want our latent to be noise-robust and continuous, so we remove the vector quantization module, constrain the intermediate latent size, and add a small KL-divergence loss term as used in vanilla VAEs Kingma and Welling [2022]. The KL-divergence loss is to ensure a noise-robust latent space. For the encoder and decoder, we use DAC's default 24kHz model but with a new latent size of $C = 80$. Given an instrumental source $s \in \mathbb{R}^N$ with N samples, the encoder encodes the waveform to a posterior $\Psi_{enc}(s) = \mathcal{N}(\cdot|\mu_z(s), \Sigma_z(s))$, where $\mu_z(s) \in \mathbb{R}^{C \times \frac{N}{D}}$ is the posterior mean of the latent and Σ_s is the corresponding posterior covariance. D is the time-domain downsampling factor of the encoder, which is 320 in DAC. Then any $z \sim \mathcal{N}(\cdot|\mu_z(s), \Sigma_z(s))$, processed by the decoder ψ_{dec}, should reconstruct s with good quality. To extract

latent features, we take the posterior mean $z_s = \mu_z(s)$. During training, the loss is as shown below:

$$L_{\text{SourceVAE}} = \lambda_1 L_{\text{Mel}} + \lambda_2 L_{\text{feature}} + \lambda_3 L_{\text{adversarial}} + \lambda_4 L_{\text{KL}} \tag{1}$$

where L_{Mel}, L_{feature}, $L_{\text{adversarial}}$ are Mel-reconstruction loss, feature matching loss, and adversarial loss, respectively, as in the DAC training framework. Also $\lambda_1 = 15$, $\lambda_2 = 2$, $\lambda_3 = 1$ according to DAC. L_{KL} is the KL-Divergence between the VAE encoded posterior and $\mathcal{N}(0,I)$ and we set $\lambda_4 = 10$. The implementation details are available in our source code.

2.2 Multi-Source Latent Diffusion

In our setup, assume any music piece $x \in \mathbb{R}^N$ is a mixture of K instrumental sources: $x = \sum_{k=1}^{K} s_k$, where $S = (s_1, s_2, ..., s_K) \in \mathbb{R}^{K \times N}$ coherently added together to form the musical mixture x. Our goal is to sample from the distribution of S to get multi-source music. Instead of directly modeling the generation of S as in Mariani et al. [2024], we propose to model the generation of $Z_S = (z_{s_1}, z_{s_2}, ..., z_{s_K}) \in \mathbb{R}^{K \times C \times \frac{N}{D}}$, where $z_{s_i} \in \mathbb{R}^{C \times \frac{N}{D}}$ is the SourceVAE's latent of $s_i \in \mathbb{R}^N$. For the notation to be more abbreviated, we will ignore the subscript s for the latent, so we are modeling the generation of $Z = (z_1, z_2, ..., z_K) \in \mathbb{R}^{K \times C \times \frac{N}{D}}$.

We model the generation of $Z = (z_1, z_2, ..., z_K)$ with a score-based diffusion model Song et al. [2021]. Following EDM Karras et al. [2022], with the diffusion schedule $\sigma(t) = t$, the forward diffusion process is defined by:

$$dZ(t) = -\sigma(t) \nabla_{Z(t)} \log p(Z(t)) dt \tag{2}$$

where $Z(t) = \mathcal{N}(Z(0), \sigma^2(t) I), Z(0) = Z$. Then with an ODE solver, we sample Z by solving:

$$dZ(t) = \sigma(t) \nabla_{Z(t)} \log p(Z(t)) dt \tag{3}$$

We approximate the score $\nabla_{Z(t)} \log p(Z(t))$ with a neural network $S^\theta(Z(t), \sigma(t))$ and then train the score matching loss following EDM Karras et al. [2022], i.e. $\sigma_{data} = 0.4, p_{train}(\sigma) = \text{Uniform}(0,3)$.

2.3 Inference

The inference pipeline is marked by the dashed objects in Fig. 1. During sampling, we use the same sampler for MSDM Mariani et al. [2024]. The sampler is an Euler method-based ODE solver to integrate Eq. 3 with some stochasticity controlled by the parameter s_{churn}, as proposed in EDM Karras et al. [2022]. We use $\sigma_{min} = 0.01, \sigma_{max} = 3, \rho = 7, s_{churn} = 20, n_{steps} = 150$, also following EDM.

2.3.1 Total Generation

The total generation inference is straightforward. Starting from randomly sampled white noise $Z(T) \sim \mathcal{N}(\cdot|0, \sigma_{max}^2 I)$, the diffusion sampling process gradually transforms $Z(T)$ to $Z(0)$. Then instrumental source latent $z_1, z_2, ..., z_K$ are extracted from $Z(0)$, and are further decoded independently by the SourceVAE decoder to get the generated source waveforms $\{s_i \in \mathbb{R}^N | s_i = \psi_{dec}(z_i), i \in [1,2,...,K]\}$. These generated sources could then be added to form a mixture of music pieces $x = \sum_{k=1}^{K} s_i$.

2.3.2 Partial Generation

Partial generation is the task of generating complementary sources given some existing ones to condition on. Assume a subset of instruments are given by the indices $I \subset \{1, 2, ...K\}$, and the corresponding given sources are denoted by $S_I = \{s_i\}_{i \in I}$. Then the complementary sources to generate are indexed by $\bar{I} = \{1, ..., K\} \backslash I$, which means the source to generate are $S_{\bar{I}} = \{s_i\}_{i \in \bar{I}}$. Since our diffusion model works in the latent domain, we first use SourceVAE to encode each source in S_I to latent $Z_I = \{z_i | z_i = \mu_z(s_i), i \in I\}$. The task is to generate $Z_{\bar{I}}$ conditioned on Z_I, so we need the conditional score $\nabla_{Z_{\bar{I}}(t)} \log p(Z_{\bar{I}}(t) | Z_I(t))$ for sampling. Following diffusion-based imputation Song et al. [2021] and MSDM Mariani et al. [2024], we could estimate the condition score by:

$$\nabla_{Z_{\bar{I}}(t)} \log p(Z_{\bar{I}}(t) | Z_I(t)) \approx \nabla_{Z_{\bar{I}}(t)} \log p([Z_{\bar{I}}(t), \hat{Z}_I(t)]) \tag{4}$$

$$\approx S^\theta([Z_{\bar{I}}(t), \hat{Z}_I(t)], \sigma(t)) \tag{5}$$

where $\hat{Z}_I(t)$ is sampled from $\mathcal{N}(\cdot | Z_I(0), \sigma^2(t))$. Then using Eq. 5, we can use the sampling method in Sec. 2.3 to solve the following ODE (similar to Eq.3) initialized from $Z_{\bar{I}}(T) \sim \mathcal{N}(\cdot|0, \sigma_{max}^2 I)$:

$$dZ_{\bar{I}}(t) = \sigma(t) S^\theta([Z_{\bar{I}}(t), \hat{Z}_I(t)], \sigma(t)) dt \tag{6}$$

With $Z_{\bar{I}}(0)$ sampled, the partially generated sources $S_{\bar{I}}$ can be decoded by the SourceVAE decoder: $S_{\bar{I}} = \{s_i \in \mathbb{R}^N | s_i = \psi_{dec}(z_i), z_i \in Z_{\bar{I}}\}$. Then all the conditional sources and partially generated sources are added to form the final music piece x.

3 Experiments and Dataset

The model architectures and training details are explained in Appendix A. We use the same dataset as MSDM, namely the `slakh2100` music dataset Manilow et al. [2019]. slakh2100 is a MIDI synthesized music dataset with 145 hours of music, containing both the mixed music and the individual tracks labeled with instrument class. Same as MSDM, we use $K=4$ main tracks which are piano, drums, bass, and guitar for multi-source modeling. For fair comparison, we use the identical sampling rate of 22,050Hz.

4 Evaluation Metrics and Results

We evaluate two tasks: (1) Total generation and (2) Partial generation as mentioned in Sec. 2.3. For both tasks, we evaluate the FAD score Kilgour et al. [2019] and human subjective score with a listening test. We compare our performance against 3 baselines.

4.1 Baseline Models

The first baseline is the **MSDM** Mariani et al. [2024] model. To show the effectiveness of modeling sources instead of mixtures, we design another baseline called **MixLDM**, where the latent diffusion model directly models the latent of music mixture, instead of instrumental sources. This is a more common practice in diffusion-based audio/music generation, where the model directly models the music mixture. For training this model, we first train a MixtureVAE which is the same as SourceVAE except that it is trained on Mixture music. The latent size $C=320$, so that the diffusion model's input is of the same dimension as MSLDM. To claim that our model generates sources that are mutually coherent (i.e. in harmony with each other), we design one baseline called **ISLDM** (Independent Source Latent Diffusion Model), where we train *four independent diffusion models on four instruments' latents*, respectively, so each model can generate one single instrument. For each single source model, the input channel size becomes $C=80$, and all the other training parameters are the same as MSLDM. Since all the single source models are independent of each other, they cannot generate mutually coherent sources. This is set as a baseline to validate our model's abilities to generate mutually coherent sources. All models' parameters and inference time to generate one 12-second mixture on a single RTX A6000 GPU are listed in Table 1.

	MSDM	**MixLDM**	**ISLDM**	**MSLDM**	**MSLDM-Large**
# parameters (M)	405	364	364×4	364	1654
Inference Time (S)	7.92	5.44	5.44×4	5.44	7.44

Table 1: Model parameters and inference time for generating a 12-second music mixture.

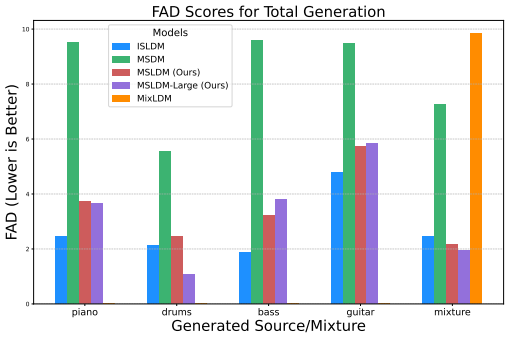

Figure 2: Total Generation FAD.

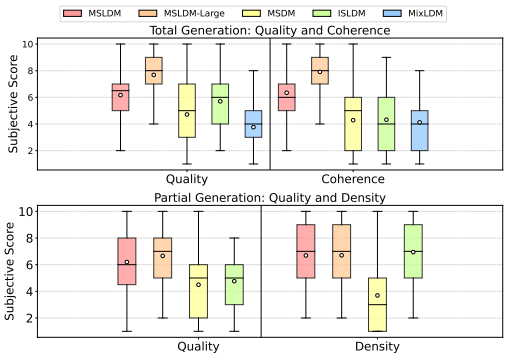

Figure 3: Subjective listening test result.

4.2 Fréchet Audio Distance

We use the Fréchet Audio Distance (FAD) Kilgour et al. [2019] with VGGish feature Hershey et al. [2017] as the objective metric to evaluate total partial generation. We use 200 chunks of music segment for the FAD calculation, where each is about 12 seconds, as in MSDM Mariani et al. [2024]. We randomly sample 200 12-second long music segments from the slakh2100 test set as the reference set.

For the task of **total generation**, we evaluate the generation of all the instrumental sources (piano, drums, bass, guitar) and the summed mixture (note: MixLDM can only be reported for the mixture). The FAD results are reported in Fig. 2. First, for single-source generation, ISLDM in general

has the lowest FAD, showing the best generation performance. This is reasonable because each independent diffusion model only needs to model one single source. Then our proposed MSLDM and MSLDM-Large also show promising results, beating the MSDM baseline by a large margin. The drums generated by MSLDM-Large even achieve lower FAD than ISLDM. For mixture generation, MixLDM exhibits the worst performance. This confirms our belief that modeling the mixture directly is much harder than modeling sources. Observe that ISLDM, MSLDM, MSLDM-Large all outperform MSDM, demonstrating the efficacy of latent diffusion in modeling the harmony in realistic music mixtures. In fact, MSLDM and MSLDM-Large demonstrate lower FAD than ISLDM, showing that our model successfully generates sources in harmony. Note that when generating each single source, MSLDM performs worse than ISLDM for all instruments, but when sources are added together to form mixtures, MSLDM has lower FAD, showing that it is able to model the inter-source harmony.

4.3 Subjective Listening Test

To complement the objective metrics, we also design subjective listening tests with human evaluation. We follow the exact test design in MSDM Mariani et al. [2024] for total and partial generation.

Total Generation: Each model generates 10 segments of 12-second music mixture samples. Then for each sample in the test, the participant is asked to rate the 'quality' and 'coherence', with a score from 1 to 10 (higher the better). 'Quality' corresponds to how realistic the music is (considering white noise is the least realistic) and 'coherence' corresponds to how mutually coherent the sources inside the mixture are. The evaluation scores are shown in the upper half of Figure 3. MSLDM and MSLDM-Large lead other baselines by a large margin in both quality and coherence. Also, MSLDM-Large exhibits a higher score than MSLDM, showing that large model size results in better performance. ISLDM's quality is better than MSDM, but its coherence is the worst because all the sources are generated independently. MixLDM has the worst quality and coherence, showing the difficulty in directly modeling the music mixture. In general, users find that our MSLDM model is consistently better than MSLDM, ISDM, MixLDM in both quality and coherence.

Partial Generation: We randomly sample 10 samples in the test set. Then, for each sample, the target source types (e.g. want to generate piano and drums based on all other instruments) are randomly sampled. Finally, the models are used to sample the specified source type for each data sample. We give the participants three music segments (the music to condition on, the partially generated music, and the synthesized mixture), and then ask them to rate 1-10 for 'quality' and 'density'. The 'quality' here means how coherent the generated sources are when mixed with the given sources. The 'density' corresponds to how much the generated instruments are present in the 12-second chunk. The results are shown in the lower half of Fig. 3. We can see that MSLDM and MSLDM-Large produce the highest quality, showing the best performance in generating consistent sources that match the given ones. Again, ISLDM shows the worst quality because sources are independent. For density, MSLDM, MSLDM-Large, and ISLDM all have high scores, showing the ability to generate rich companions. However, MSDM shows the worst performance in density because it often generates small and naive music segments, or even empty sources at times.

Overall, both the objective metrics and subjective listening test show the MSLDM outperforms MSDM, ISLDM, MixLDM in terms of generating multi-source consistent music pieces. Compared with MSDM, MSLDM is better in terms of generating denser, more melodic, and more harmonious music. Compared with MixLDM, we showed that modeling sources is much easier than directly modeling music mixtures. Compared with ISLDM, we showed that our model is able to model inter-source dependency or harmony between different musical instruments.

5 Conclusion and Future Work

In this paper, we propose a multi-source diffusion model to jointly model the generation of multiple instruments together. Both objective and subjective metrics show better results in the tasks of total and partial generation, implying MSLDM is able to efficiently model melody and inter-source relations. Future research needs to advance MSLDM for weakly-supervised music separation and generalize our model to more instruments.

6 Acknowledgement

We are also grateful to Foxconn and NSF (grants 2008338, 1909568, and MRI-2018966) for partially supporting this research.

References

Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. Musiclm: Generating music from text, 2023. URL https://arxiv.org/abs/2301.11325.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

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, 2023. URL https://arxiv.org/abs/2308.01546.

Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation, 2024. URL https://arxiv.org/abs/2306.05284.

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music, 2020. URL https://arxiv.org/abs/2005.00341.

Chris Donahue, Antoine Caillon, Adam Roberts, Ethan Manilow, Philippe Esling, Andrea Agostinelli, Mauro Verzetti, Ian Simon, Olivier Pietquin, Neil Zeghidour, and Jesse Engel. Singsong: Generating musical accompaniments from singing, 2023. URL https://arxiv.org/abs/2301.12662.

Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment, 2017. URL https://arxiv.org/abs/1709.06298.

Zach Evans, Julian D. Parker, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons. Stable audio open, 2024a. URL https://arxiv.org/abs/2407.14358.

Zach Evans, Julian D. Parker, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons. Long-form music generation with latent diffusion, 2024b. URL https://arxiv.org/abs/2404.10301.

Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson. Cnn architectures for large-scale audio classification, 2017. URL https://arxiv.org/abs/1609.09430.

Qingqing Huang, Daniel S. Park, Tao Wang, Timo I. Denk, Andy Ly, Nanxin Chen, Zhengdong Zhang, Zhishuai Zhang, Jiahui Yu, Christian Frank, Jesse Engel, Quoc V. Le, William Chan, Zhifeng Chen, and Wei Han. Noise2music: Text-conditioned music generation with diffusion models, 2023a. URL https://arxiv.org/abs/2302.03917.

Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models, 2023b. URL https://arxiv.org/abs/2301.12661.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models, 2022. URL https://arxiv.org/abs/2206.00364.

Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fréchet audio distance: A metric for evaluating music enhancement algorithms, 2019. URL https://arxiv.org/abs/1812.08466.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022. URL https://arxiv.org/abs/1312.6114.

Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. High-fidelity audio compression with improved rvqgan, 2023. URL https://arxiv.org/abs/2306.06546.

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, 2023. URL https://arxiv.org/abs/2301.12503.

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, 2024. URL https://arxiv.org/abs/2308.05734.

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, 2019. URL https://arxiv.org/abs/1909.08494.

Giorgio Mariani, Irene Tallini, Emilian Postolache, Michele Mancusi, Luca Cosmo, and Emanuele Rodolà. Multi-source diffusion models for simultaneous music generation and separation, 2024. URL https://arxiv.org/abs/2302.02257.

Gautam Mittal, Jesse Engel, Curtis Hawthorne, and Ian Simon. Symbolic music generation with diffusion models, 2021. URL https://arxiv.org/abs/2103.16091.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. URL https://arxiv.org/abs/2303.08774.

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, 2024. URL https://arxiv.org/abs/2312.08723.

Marco Pasini, Maarten Grachten, and Stefan Lattner. Bass accompaniment generation via latent diffusion, 2024. URL https://arxiv.org/abs/2402.01412.

Emilian Postolache, Giorgio Mariani, Luca Cosmo, Emmanouil Benetos, and Emanuele Rodolà. Generalized multi-source inference for text conditioned music diffusion models, 2024. URL https://arxiv.org/abs/2403.11706.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021. URL https://arxiv.org/abs/2102.12092.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL https://arxiv.org/abs/2112.10752.

Flavio Schneider, Ojasv Kamal, Zhijing Jin, and Bernhard Schölkopf. Moûsai: Text-to-music generation with long-context latent diffusion, 2023. URL https://arxiv.org/abs/2301.11757.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021. URL https://arxiv.org/abs/2011.13456.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio, 2016. URL https://arxiv.org/abs/1609.03499.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2018. URL https://arxiv.org/abs/1711.00937.

Dongchao Yang, Jianwei Yu, Helin Wang, Wen Wang, Chao Weng, Yuexian Zou, and Dong Yu. Diffsound: Discrete diffusion model for text-to-sound generation, 2023. URL https://arxiv.org/abs/2207.09983.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec, 2021. URL https://arxiv.org/abs/2107.03312.

A Model Architectures and Training Details

A.1 SourceVAE

The SourceVAE mentioned in Sec. 2.1 is a 1D-CNN-based encoder-decoder architecture coupled with a DAC loss and a KL-divergence loss. The encoder and decoder all follow the final setup of the DAC architecture for 24kHz. The intermediate latent dimension for SourceVAE is set to be $C = 80$ and the encoder has a downsampling rate of $D = 320$ for the temporal dimension. For training, we use a batch size of 28 and train on one-second-long single-instrumental segments for 100k steps. All other SourceVAE training configurations are the same as the DAC paper with code available at https://github.com/descriptinc/descript-audio-codec.

A.2 Latent Diffusion and Unet Architecture

As mentioned in Sec. 2.2, the score estimation network $S^\theta(Z(t),\sigma(t))$ learns a function mapping as shown below:

$$S^\theta(Z(t),\sigma(t)) : \mathbb{R}^{K\times C\times \frac{N}{D}} \times \mathbb{R} \to \mathbb{R}^{K\times C\times \frac{N}{D}} \tag{7}$$

To accommodate the 1D-Unet architecture used in MSDM, we concatenate the K source latent channel-wise, so the new channel dimension becomes KC, and the input to the 1D-Unet is $Z'(t) \in \mathbb{R}^{KC\times \frac{N}{D}}$, which is a reshaped version of $Z(t)$. KC is treated as the channel dimension and $\frac{N}{D}$ is treated as the temporal dimension for the Unet. In our setup, $K = 4, C = 80, D = 320$. We train our diffusion model on segments with $N = 327672$ samples (about 15 seconds). For the 1D-Unet architecture, similar to MSDM, we adapt the architecture used in Moûsai Schneider et al. [2023] but with some modifications. We set the input channel dimension to be $KC = 320$ and then we experiment on two different configurations. We call one model MSLDM and one larger model MSLDM-`Large`. The MSLDM contains 6 nested U-Net blocks with increasing channels [1024, 2048, 4096, 4096, 4096, 4096]. The downsampling factor for the blocks is [1,1,2,1,1,2]. The self-attention blocks are used for all the blocks except the first one. 12 Attention heads are used, and each head is 64 dimensional. For MSDLM-Large, the Unet contains 8 layers where the corresponding output channel dimensions are [1024, 2048, 4096, 4096, 4096, 4096, 4096, 4096]. The corresponding downsampling factors are [1,1,2,1,1,2,2,2]. All blocks contain self-attention blocks (except the first one) and each attention block contains 12 attention heads that are each 128 dimensional. The Unet and diffusion code setup are adapted from audio-diffusion-pytorch. Similar to MSDM, we train the diffusion model with a batch size 16 and a learning rate of 2e-5 for 400k steps.

All the model training is performed on a single RTX A6000 GPU with 48GB of VRAM. Further details can be found in our code.

B Partial Generation sub-FAD

Table 2: **sub-FAD for Partial Generation.** The sub-FAD (lower is better) is reported for any source combinations (B: Bass, D: Drums, G: Guitar, P: Piano), where **BD** means conditioned on piano and guitar, the task is to generate Bass and Drums.

Model	**B**	**D**	**G**	**P**	**BD**	**BG**	**BP**	**DG**	**DP**	**GP**	**BDG**	**BDP**	**BGP**	**DGP**	**Overall**
MSDM	0.23	0.75	**0.18**	0.49	1.75	0.75	1.40	1.30	1.40	1.77	3.13	2.92	5.54	3.51	1.79
ISLDM	0.30	1.41	0.75	0.42	1.52	1.14	0.76	1.56	1.76	1.33	1.85	2.03	1.78	2.17	1.34
MSLDM	0.24	1.27	0.38	**0.32**	1.22	0.81	0.64	1.00	0.92	**0.98**	1.44	1.57	1.48	1.43	0.98
MSLDM-Large	**0.14**	**0.51**	0.23	0.41	**0.56**	**0.49**	**0.61**	**0.59**	**0.66**	1.05	**0.81**	**1.01**	**1.40**	**1.25**	**0.70**

For **partial generation**, we use the `sub-FAD` as the metric, similar to MSDM. `Sub-FAD` calculates the FAD on a reference set and an evaluation set, where the reference set is the real music mixture, and the evaluated set is the mixture formed by mixing partially generated samples and originally given samples. For the dataset used for `sub-FAD` calculation, we again sample 200 12-second music segments from the slakh2100 test set, but we make sure that all the segments contain four instruments. The `sub-FAD` result is shown in Table 2, where the results are shown for all models and all partial generation setups.

The ISLDM model in this case is only generating the target sources using source-independent models. Overall, MSLDM-Large and MSLDM show much better performance than ISLDM and MSDM, with FADs smaller than 1. Interestingly, the ISLDM's overall FAD is lower than MSDM, implying that even though ISLDM cannot generate mutually coherent sources based on given sources, it is able to model single-source melody much better than MSDM. Across all the detailed setups, we see that MSLDM is consistently better than MSDM and ISLDM, except for the guitar.