

---

# Multi-Source Music Generation with Latent Diffusion

---

Anonymous Author(s)

Affiliation

Address

email

## Abstract

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

## 22 1 Introduction

23 Generative models have shown impressive results not only in language and image generation OpenAI  
24 [2023], Ramesh et al. [2021], Brown et al. [2020], but also in music generation. Music generation  
25 models primarily fall into two categories: 1) Auto-regressive and 2) Diffusion-based. Auto-regressive  
26 models like WaveNet van den Oord et al. [2016] directly model scalar-quantized waveform samples  
27 but suffer from low sampling efficiency. Further research Dhariwal et al. [2020], Agostinelli et al.  
28 [2023], Copet et al. [2024] starts to auto-regressively model audio/speech/music tokens, which are  
29 often extracted from audio tokenizers van den Oord et al. [2018], Dhariwal et al. [2020], [?], Kumar  
30 et al. [2023], Zeghidour et al. [2021].

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

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

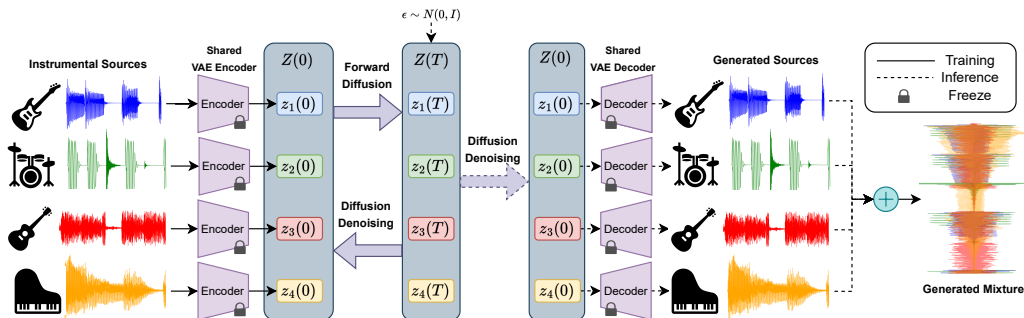


Figure 1: An Overview of the proposed MSLDM framework.

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

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

## 65 2 Models

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

### 73 2.1 SourceVAE

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

87  $\mathcal{N}(\cdot|\mu_z(s),\Sigma_z(s))$ , processed by the decoder  $\psi_{\text{dec}}$ , should reconstruct  $s$  with good quality. To extract  
 88 latent features, we take the posterior mean  $z_s = \mu_z(s)$ . During training, the loss is as shown below:

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

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

## 93 2.2 Multi-Source Latent Diffusion

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

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

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

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

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

105 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  
 106 score matching loss following EDM Karras et al. [2022], i.e.  $\sigma_{\text{data}} = 0.4, p_{\text{train}}(\sigma) = \text{Uniform}(0, 3)$ .

## 107 2.3 Inference

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

### 112 2.3.1 Total Generation

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

### 118 2.3.2 Partial Generation

119 Partial generation is the task of generating complementary sources given some existing ones to  
 120 condition on. Assume a subset of instruments are given by the indices  $I \subset \{1, 2, \dots, K\}$ , and the  
 121 corresponding given sources are denoted by  $S_I = \{s_i\}_{i \in I}$ . Then the complementary sources to  
 122 generate are indexed by  $\bar{I} = \{1, \dots, K\} \setminus I$ , which means the source to generate are  $S_{\bar{I}} = \{s_i\}_{i \in \bar{I}}$ . Since  
 123 our diffusion model works in the latent domain, we first use SourceVAE to encode each source in  $S_I$   
 124 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  
 125 conditional score  $\nabla_{Z_{\bar{I}}(t)}\log p(Z_{\bar{I}}(t)|Z_I(t))$  for sampling. Following diffusion-based imputation Song  
 126 et al. [2021] and MSDM Mariani et al. [2024], we could estimate the condition score by:

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

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

127 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  
 128 in Sec. 2.3 to solve the following ODE (similar to Eq.3) initialized from  $Z_{\bar{I}}(T) \sim \mathcal{N}(\cdot|0, \sigma_{\text{max}}^2 I)$ :

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

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

### 132 3 Experiments and Dataset

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

### 138 4 Evaluation Metrics and Results

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

#### 142 4.1 Baseline Models

143 The first baseline is the **MSDM** Mariani et al. [2024] model. To show the effectiveness of modeling  
 144 sources instead of mixtures, we design another baseline called **MixLDM**, where the latent diffusion  
 145 model directly models the latent of music mixture, instead of instrumental sources. This is a more  
 146 common practice in diffusion-based audio/music generation, where the model directly models the  
 147 music mixture. For training this model, we first train a MixtureVAE which is the same as SourceVAE  
 148 except that it is trained on Mixture music. The latent size  $C = 320$ , so that the diffusion model’s input  
 149 is of the same dimension as MSLDM. To claim that our model generates sources that are mutually  
 150 coherent (i.e. in harmony with each other), we design one baseline called **ISLDM** (Independent Source  
 151 Latent Diffusion Model), where we train *four independent diffusion models on four instruments’*  
 152 *latents*, respectively, so each model can generate one single instrument. For each single source model,  
 153 the input channel size becomes  $C = 80$ , and all the other training parameters are the same as MSLDM.  
 154 Since all the single source models are independent of each other, they cannot generate mutually  
 155 coherent sources. This is set as a baseline to validate our model’s abilities to generate mutually  
 156 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	$364 \times 4$	364	1654
Inference Time (S)	7.92	5.44	$5.44 \times 4$	5.44	7.44

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

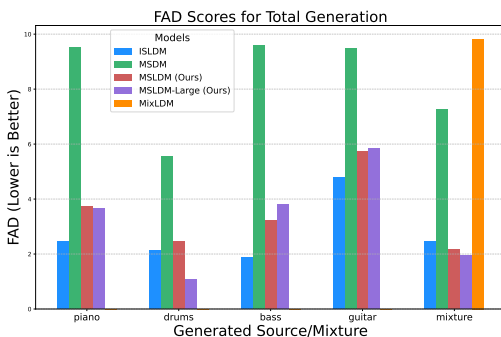


Figure 2: Total Generation FAD.

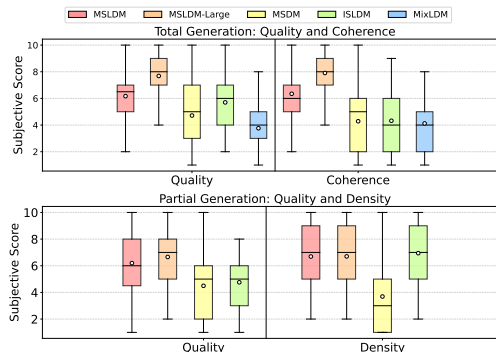


Figure 3: Subjective listening test result.

#### 158 4.2 Fréchet Audio Distance

159 We use the Fréchet Audio Distance (FAD) Kilgour et al. [2019] with VGGish feature Hershey et al.  
 160 [2017] as the objective metric to evaluate total partial generation. We use 200 chunks of music segment  
 161 for the FAD calculation, where each is about 12 seconds, as in MSDM Mariani et al. [2024]. We  
 162 randomly sample 200 12-second long music segments from the slakh2100 test set as the reference set.

163 For the task of **total generation**, we evaluate the generation of all the instrumental sources (piano,  
164 drums, bass, guitar) and the summed mixture (note: MixLDM can only be reported for the mixture).  
165 The FAD results are reported in Fig. 2. First, for single-source generation, ISLDM in general  
166 has the lowest FAD, showing the best generation performance. This is reasonable because each  
167 independent diffusion model only needs to model one single source. Then our proposed MSLDM  
168 and MSLDM-Large also show promising results, beating the MSDM baseline by a large margin. The  
169 drums generated by MSLDM-Large even achieve lower FAD than ISLDM. For mixture generation,  
170 MixLDM exhibits the worst performance. This confirms our belief that modeling the mixture directly  
171 is much harder than modeling sources. Observe that ISLDM, MSLDM, MSLDM-Large all outperform  
172 MSDM, demonstrating the efficacy of latent diffusion in modeling the harmony in realistic music  
173 mixtures. In fact, MSLDM and MSLDM-Large demonstrate lower FAD than ISLDM, showing that  
174 our model successfully generates sources in harmony. Note that when generating each single source,  
175 MSLDM performs worse than ISLDM for all instruments, but when sources are added together to  
176 form mixtures, MSLDM has lower FAD, showing that it is able to model the inter-source harmony.

### 177 4.3 Subjective Listening Test

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

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

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

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

## 210 5 Conclusion and Future Work

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

## 216 References

- 217 Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Cail-  
218 lon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil  
219 Zeghidour, and Christian Frank. Musiclm: Generating music from text, 2023. URL  
220 <https://arxiv.org/abs/2301.11325>.
- 221 Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal,  
222 Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel  
223 Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M.  
224 Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz  
225 Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec  
226 Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL  
227 <https://arxiv.org/abs/2005.14165>.
- 228 Ke Chen, Yusong Wu, Haohe Liu, Marianna Nezhurina, Taylor Berg-Kirkpatrick, and Shlomo Dubnov.  
229 Musicldm: Enhancing novelty in text-to-music generation using beat-synchronous mixup strategies,  
230 2023. URL <https://arxiv.org/abs/2308.01546>.
- 231 Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi  
232 Adi, and Alexandre Défossez. Simple and controllable music generation, 2024. URL  
233 <https://arxiv.org/abs/2306.05284>.
- 234 Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever.  
235 Jukebox: A generative model for music, 2020. URL <https://arxiv.org/abs/2005.00341>.
- 236 Chris Donahue, Antoine Caillon, Adam Roberts, Ethan Manilow, Philippe Esling, Andrea Agostinelli,  
237 Mauro Verzetti, Ian Simon, Olivier Pietquin, Neil Zeghidour, and Jesse Engel. Singsong: Generating  
238 musical accompaniments from singing, 2023. URL <https://arxiv.org/abs/2301.12662>.
- 239 Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential  
240 generative adversarial networks for symbolic music generation and accompaniment, 2017. URL  
241 <https://arxiv.org/abs/1709.06298>.
- 242 Zach Evans, Julian D. Parker, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons. Stable audio  
243 open, 2024a. URL <https://arxiv.org/abs/2407.14358>.
- 244 Zach Evans, Julian D. Parker, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons. Long-form  
245 music generation with latent diffusion, 2024b. URL <https://arxiv.org/abs/2404.10301>.
- 246 Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing  
247 Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J.  
248 Weiss, and Kevin Wilson. Cnn architectures for large-scale audio classification, 2017. URL  
249 <https://arxiv.org/abs/1609.09430>.
- 250 Qingqing Huang, Daniel S. Park, Tao Wang, Timo I. Denk, Andy Ly, Nanxin Chen, Zhengdong Zhang,  
251 Zhishuai Zhang, Jiahui Yu, Christian Frank, Jesse Engel, Quoc V. Le, William Chan, Zhifeng Chen,  
252 and Wei Han. Noise2music: Text-conditioned music generation with diffusion models, 2023a. URL  
253 <https://arxiv.org/abs/2302.03917>.
- 254 Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin  
255 Liu, Xiang Yin, and Zhou Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced  
256 diffusion models, 2023b. URL <https://arxiv.org/abs/2301.12661>.
- 257 Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of  
258 diffusion-based generative models, 2022. URL <https://arxiv.org/abs/2206.00364>.
- 259 Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fréchet au-  
260 dio distance: A metric for evaluating music enhancement algorithms, 2019. URL  
261 <https://arxiv.org/abs/1812.08466>.
- 262 Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022. URL  
263 <https://arxiv.org/abs/1312.6114>.

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

266 Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and  
267 Mark D. Plumbley. Audioldm: Text-to-audio generation with latent diffusion models, 2023. URL  
268 <https://arxiv.org/abs/2301.12503>.

269 Haohe Liu, Yi Yuan, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Qiao Tian, Yuping Wang, Wenwu  
270 Wang, Yuxuan Wang, and Mark D. Plumbley. Audioldm 2: Learning holistic audio generation with  
271 self-supervised pretraining, 2024. URL <https://arxiv.org/abs/2308.05734>.

272 Ethan Manilow, Gordon Wichern, Prem Seetharaman, and Jonathan Le Roux. Cutting music source  
273 separation some slakh: A dataset to study the impact of training data quality and quantity, 2019.  
274 URL <https://arxiv.org/abs/1909.08494>.

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

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

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

282 Julian D. Parker, Janne Spijkervet, Katerina Kosta, Furkan Yesiler, Boris Kuznetsov, Ju-Chiang Wang,  
283 Matt Avent, Jitong Chen, and Duc Le. Stemgen: A music generation model that listens, 2024. URL  
284 <https://arxiv.org/abs/2312.08723>.

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

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

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

293 Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Om-  
294 mer. High-resolution image synthesis with latent diffusion models, 2022. URL  
295 <https://arxiv.org/abs/2112.10752>.

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

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

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

304 Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning,  
305 2018. URL <https://arxiv.org/abs/1711.00937>.

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

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

## 311 A Model Architectures and Training Details

### 312 A.1 SourceVAE

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

### 320 A.2 Latent Diffusion and Unet Architecture

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

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

323 To accommodate the 1D-Unet architecture used in MSDM, we concatenate the  $K$  source latent channel-  
 324 wise, so the new channel dimension becomes  $KC$ , and the input to the 1D-Unet is  $Z'(t) \in \mathbb{R}^{KC \times \frac{N}{D}}$ ,  
 325 which is a reshaped version of  $Z(t)$ .  $KC$  is treated as the channel dimension and  $\frac{N}{D}$  is treated as the tem-  
 326 poral dimension for the Unet. In our setup,  $K = 4, C = 80, D = 320$ . We train our diffusion model on seg-  
 327 ments with  $N = 327672$  samples (about 15 seconds). For the 1D-Unet architecture, similar to MSDM,  
 328 we adapt the architecture used in Moûsai Schneider et al. [2023] but with some modifications. We set the  
 329 input channel dimension to be  $KC = 320$  and then we experiment on two different configurations. We  
 330 call one model MSLDM and one larger model MSLDM-Large. The MSLDM contains 6 nested U-Net  
 331 blocks with increasing channels [1024, 2048, 4096, 4096, 4096, 4096]. The downsampling factor for  
 332 the blocks is [1, 1, 2, 1, 1, 2]. The self-attention blocks are used for all the blocks except the first one. 12  
 333 Attention heads are used, and each head is 64 dimensional. For MSDLM-Large, the Unet contains 8  
 334 layers where the corresponding output channel dimensions are [1024, 2048, 4096, 4096, 4096, 4096,  
 335 4096, 4096]. The corresponding downsampling factors are [1, 1, 2, 1, 1, 2, 2, 2]. All blocks contain self-  
 336 attention blocks (except the first one) and each attention block contains 12 attention heads that are each  
 337 128 dimensional. The Unet and diffusion code setup are adapted from audio-diffusion-pytorch. Similar  
 338 to MSDM, we train the diffusion model with a batch size 16 and a learning rate of  $2e-5$  for 400k steps.  
 339 All the model training is performed on a single RTX A6000 GPU with 48GB of VRAM. Further details  
 340 can be found in our code.

## 341 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	<b>0.18</b>	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	<b>0.32</b>	1.22	0.81	0.64	1.00	0.92	<b>0.98</b>	1.44	1.57	1.48	1.43	0.98
MSLDM-Large	<b>0.14</b>	<b>0.51</b>	0.23	0.41	<b>0.56</b>	<b>0.49</b>	<b>0.61</b>	<b>0.59</b>	<b>0.66</b>	1.05	<b>0.81</b>	<b>1.01</b>	<b>1.40</b>	<b>1.25</b>	<b>0.70</b>

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



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