# Improving Source Extraction with Diffusion and Consistency Models

**Tornike Karchkhadze** [†] [*]
University of California San Diego
tkarchkhadze@ucsd.edu

**Mohammad Rasool Izadi** [†]
Bose Corp.
russell_izadi@bose.com

**Shuo Zhang**
Bose Corp.
shuo_zhang@bose.com

## Abstract

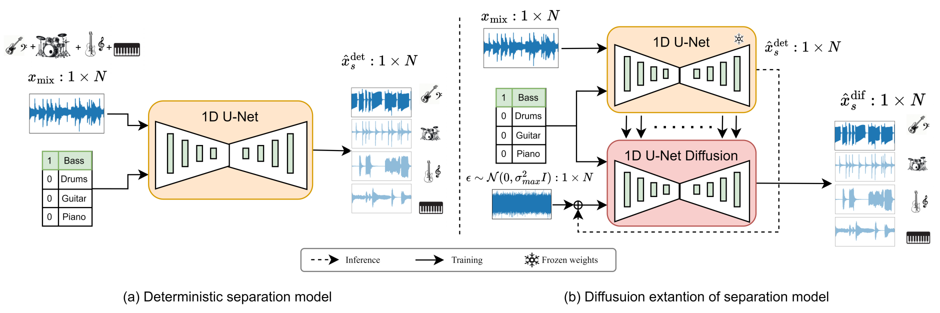
In this work, we integrate a score-matching diffusion model into a standard deterministic architecture for time-domain musical source extraction. To address the typically slow iterative sampling process of diffusion models, we apply consistency distillation and reduce the sampling process to a single step, achieving performance comparable to that of diffusion models, and with two or more steps, even surpassing them. Trained on the Slakh2100 dataset for four instruments (bass, drums, guitar, and piano), our model shows significant improvements across objective metrics compared to baseline methods. Sound examples are available at https://consistency-separation.github.io/.

## 1 Introduction

Audio source extraction and separation involve isolating individual sound elements from a mixture of sounds. This technique is crucial in various fields, particularly in music production, restoration, analysis, music education, transcription, etc. Recent advances in deep learning technology have significantly impacted the field of audio source separation, leading to substantial improvements in its quality. There are two primary approaches to audio source separation using machine learning. The first approach involves deterministic discriminative models [1–7], which typically use mixtures for conditioning in the training and inference process and learn how to derive one or more sources from mixtures. On the other hand, generative models [8–17] generally learn a prior distribution of sources and use the mixture during inference to generate separate sources. Recently, there have been several attempts to apply diffusion models [18, 19] to audio source separation and extraction tasks [20–26]. For a detailed account on related works, please refer to Appendix A.

We propose an extension of the deterministic mixture-conditional musical source extraction model by incorporating a generative diffusion method. First, we train a Deterministic model. Next, we introduce a denoising score-matching diffusion model [27] that enhances the extracted sources, providing a generative final touch. As is typical for diffusion models, this approach requires an iterative sampling procedure, increasing the inference time for the overall model. To speed up the generation process, we adapted methodologies presented in [28, 29] and applied Consistency Distillation (CD) to the diffusion model, reducing the number of denoising steps. Our CD model achieved the accelerated speed of single-step denoising without a loss of quality. Additionally, the consistency model provided the option to trade off between quality and speed, allowing for improved quality over the diffusion model with only 2-4 steps and a minimal increase in time compared to the Deterministic model. We trained our model on the Slakh2100 [30] dataset, focusing on four instruments: bass, drums, guitar, and piano. In our experiments, we compared our model with baseline models Demucs [6], Demucs+Gibbs [14], and MSDM [26] and demonstrated significant improvements in the objective metrics of music separation.

[*]The work was done during an internship at Bose Corporation.
[†]Authors with equal contribution.



$x_{\text{mix}}: 1 \times N$
1 Bass
0 Drums
0 Guitar
0 Piano
1D U-Net
$\hat{x}_s^{\text{det}}: 1 \times N$
1D U-Net Diffusion
$\epsilon \sim \mathcal{N}(0, \sigma_{max}^2 I): 1 \times N$
$\hat{x}_s^{\text{dif}}: 1 \times N$
Inference
Training
Frozen weights
(a) Deterministic separation model
(b) Diffusuion extantion of separation model

Figure 1: **Diagram illustrating our proposed method.** (a) We first train a mixture-conditional deterministic source extraction model. (b) We then introduce a denoising score-matching diffusion model, conditioned on features extracted by the Deterministic model, which further enhances extraction by adding and removing noise.

**Contributions** Our work presents three main contributions: **(i)** We bridge the gap between deterministic and generative source separation methods, showing they can complement each other and improve model performance. **(ii)** We introduce a consistency framework for raw audio and demonstrate student CD model outperforming teacher diffusion model. **(iii)** We achieve significant improvements in musical source extraction and set a new benchmark on the Slakh2100 dataset.

## 2 Method

Let $x_{\text{mix}}$ represent a time-domain audio mixture containing $S$ individual tracks $x_s \in 1 \times N$, where $N$ is the number of audio samples and $s \in \{1, \ldots, S\}$ identifies each source. The mixture is defined as $x_{\text{mix}} = \sum_{s=1}^{S} x_s$. The source extraction problem is to minimize some loss function $\mathcal{L}$ that measures the average error between the true source $x_s$ and its corresponding prediction $\hat{x}_s$. Having all $S$ sources extracted, the task becomes one of source separation.

### 2.1 Deterministic Model

We develop and train a deterministic source extraction model $f_\theta$ using a U-Net encoder-decoder architecture, as shown in the left side of Fig. 1. This network, consisting of 1D convolutional layers with skip connections, is a common choice for time-domain source separation [6, 14, 31]. The model has two inputs: one for the mixture signal $x_{\text{mix}}$ and another for the instrument of interest $s$. We train our Deterministic model $f_\theta$ with the following loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{s, x_{\text{mix}}} \|x_s - \hat{x}_s^{\text{det}}\|_2^2, \tag{1}$$

where $\hat{x}_s^{\text{det}} = f_\theta(x_{\text{mix}}, s)$ is the prediction for source $s$.

### 2.2 Diffusion Model

After training the Deterministic model $f_\theta$, we freeze its parameters and integrate it into a larger system that includes a score-matching diffusion model $g_\phi$, as depicted in right side of Fig. 1. The diffusion model $g_\phi$ has the same architecture as the Deterministic model but takes four inputs instead of two: (1) a noisy version of $x_s$, (2) the target label $s$, (3) the scale of noise $\sigma$ added to the $x_s$, and (4) the intermediate features $\bar{x}_s^{\text{det}}$extracted by the frozen Deterministic model. We train the diffusion model $g_\phi$ with the Denoising Score Matching (DSM) loss:

$$\mathcal{L}_{\text{DSM}}(\phi) = \mathbb{E}_{s, x_{\text{mix}}, t} \|x_s - g_\phi(x_s + \sigma_t \epsilon, s, \sigma_t, \bar{x}_s^{\text{det}})\|_2^2, \tag{2}$$

where $\epsilon \sim \mathcal{N}(0, I)$ is noise sampled from a standard Gaussian distribution, and $\sigma_t := \sigma(t)$ is a monotonically increasing function that defines the noise step and the scale of the added noise.

The inference of $g_\phi$ is an iterative process of solving a numerical Ordinary Differential Equation (ODE) over $T$ steps. At each step, the diffusion model $g_\phi$ polishes $\hat{x}_s^{\text{det}}$ using intermediate features $\bar{x}_s^{\text{det}}$ and appropriate amount of noise $\sigma_t \epsilon$ for $\sigma : [1, T] \rightarrow [\sigma_{\text{min}}, \sigma_{\text{max}}]$ where $\sigma_{\text{min}}$ and $\sigma_{\text{max}}$ show minimum and maximum noise levels, respectively. The iterative process begins with the maximum noise level $\sigma_{\text{max}}$, setting the initial prediction $\hat{x}^{\text{dif}}_{s,T-1} = \texttt{Solver}_1(\hat{x}^{\text{det}}_s + \sigma_{\text{max}}\epsilon, s, \sigma_T, \bar{x}^{\text{det}}_s; g_\phi)$ and continues with the following update rule:

$$\hat{x}^{\text{dif}}_{s,t-1} = \texttt{Solver}_1(\hat{x}^{\text{dif}}_{s,t}, s, \sigma_t, \bar{x}^{\text{det}}_s; g_\phi), \tag{3}$$

where $\texttt{Solver}_k(\dots; g_\phi)$ denotes $k$ steps of any ODE solver that uses $g_\phi$ score-based model for data denoising. This process continues until reaching clean $\hat{x}^{\text{dif}}_{s,0}$. For a background on score-based diffusion models, see Appendix B.1.

### 2.3 Consistency Model

To mitigate the latency introduced by the diffusion model in the inference process and make 1-2 steps generation possible, we adopt CD [28,29]. In this approach, our consistency model $g_\omega$ is designed as an exact replica of the diffusion model and is trained using a pretrained diffusion model $g_\phi$ as a teacher. Requiring inference of diffusion teacher model, CD is a designed as discrete process with $t \in [1, T]$, where $T$ denotes a total number of steps. We iteratively apply an ODE solver to predict progressively less noisy samples along the trajectory:

$$\hat{x}^{\text{dif}}_{s,t-h} = \texttt{Solver}_h(x_s + \sigma_t\epsilon, s, \sigma_t, \bar{x}^{\text{det}}_s; g_\phi), \tag{4}$$

where $h \in \{1, \dots, t\}$ is the number of ODE steps used in distilation process. This prediction is then used to calculate the target in our CD optimization, with the following loss:

$$\mathcal{L}_{\text{CD}}(\omega) = \mathbb{E}_{t,h} \| \underbrace{g_{\texttt{sg}(\omega)}(\hat{x}^{\text{dif}}_{s,t-h}, s, \sigma_{t-h}, \bar{x}^{\text{det}}_s)}_{\textit{target}} - \underbrace{g_\omega(x_s + \epsilon\sigma_t, s, \sigma_t, \bar{x}^{\text{det}}_s)}_{\textit{prediction}} \|^2_2, \tag{5}$$

where $\texttt{sg}(\omega)$ denotes the stop-gradient running EMA (Exponential Moving Average) of $\omega$ during optimization, updated as $\texttt{sg}(\omega) \leftarrow \texttt{stopgrad}(\mu\texttt{sg}(\omega) + (1-\mu)\omega)$, with $\mu$ denoting update rate.

Inspired by the success of introducing direct signals from data by use of the DSM loss in Consistency Trajectory Model (CTM) [29], we adopted this idea and applied the DSM loss from Eq. (2). Our final loss is formulated as:

$$\mathcal{L}(\omega) = \mathcal{L}_{\text{CD}}(\omega) + \lambda_{\text{DSM}}\mathcal{L}_{\text{DSM}}(\omega), \tag{6}$$

where $\lambda_{\text{DSM}}$ is a balancing term between two losses. A more detailed account to CD techniques is provided in Appendix B.2.

## 3 Experimental setup

For our training and evaluation experiments, we used the Slakh2100 dataset [30]. To enable direct comparison with our baselines, we followed [26] and [14], focusing on the four most prevalent instrument classes: Bass, Drums, Guitar, and Piano. Additionally, we used the MUSDB18 dataset [32] for evaluation, and we report the results in Appendix D.

All of our models—Deterministic, Diffusion, and Consistency—are based on a U-Net backbone and operate in the audio waveform domain. The diffusion model was trained following preconditioning and training procedures from the EDM framework [33]. For the Consistency model, we applied consistency distillation as described in the original work [28], but with some modifications. Specifically, we did not use schedule functions for $T$ or $\mu$, and instead kept these values fixed at $T = 18$ and $\mu = 0.999$ throughout the experiments. We used of ODE steps to $h \leq 17$ (compared to $h = 1$ in the original). We incorporated settings from the CTM framework [29] to balance losses in the Eq. (6). For a detailed account of the hyperparameters and training settings, please refer to Appendix C.

## 4 Results

We compared our Deterministic model, Diffusion model, and Consistency Distillation (CD) results on Slakh2100 with the baselines Demucs [6], Demucs + Gibbs [14], and two different methods from MSDM [26], including supervised and weakly-supervised models with the 'Dirac algorithm.' For direct comparison, we use the scale-invariant SDR improvement (SI-SDR$_\text{I}$) metric [34] and adopted the exact same evaluation procedure as implemented in MSDM and Demucs + Gibbs. We evaluate

Table 1: **SI-SDR$_\text{I}$ (dB — higher is better) results for source separation on the Slakh2100 test set.** We compare our Deterministic Model, Diffusion, and Consistency Distillation results with our baselines for all stem categories. 'All' reports the average across the four stems.

| **Model** | **Bass** | **Drums** | **Guitar** | **Piano** | **All** |
|---|---|---|---|---|---|
| Demucs [6, 14] | 15.77 | 19.44 | 15.30 | 13.92 | 16.11 |
| Demucs + Gibbs (512 steps) [14] | 17.16 | 19.61 | 17.82 | 16.32 | 17.73 |
| ISDM (Dirac with correction) [26] | 19.36 | 20.90 | 14.70 | 14.13 | 17.27 |
| MSDM (Dirac with correction) [26] | 17.12 | 18.68 | 15.38 | 14.73 | 16.48 |
| Deterministic Model | 20.04 | 20.88 | 23.82 | 20.82 | 21.39 |
| Deterministic Model double | 20.93 | 20.97 | 23.56 | 21.14 | 21.65 |
| Diffusion $(T = 5, \sigma_\text{max} = 0.01, R = 2)$ | 21.06 | 21.77 | 25.35 | 22.17 | 22.58 |
| CD onestep $(T = 1, \sigma_\text{max} = 0.01244)$ | 20.99 | 21.91 | 26.10 | 22.73 | 22.93 |
| CD multistep $(T = 2, \sigma_\text{max} = 0.2497)$ | 21.97 | 22.19 | 27.34 | 23.24 | 23.68 |
| CD multistep $(T = 4, \sigma_\text{max} = 0.2495)$ | **22.39** | **22.45** | **28.09** | **23.96** | **24.22** |

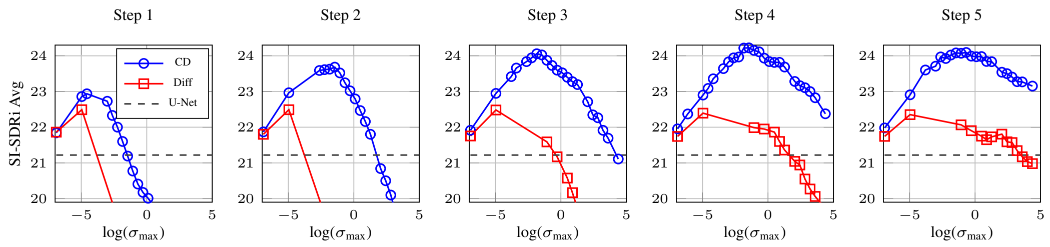


Figure 2: **SI-SDRi Avg. vs Log($\sigma_\text{max}$) for CD and Diffusion Models across 5 Steps**. Each subplot compares the performance of the CD model (blue) and the Diffusion model (red) on different numbers of denoising steps, with a dashed line representing the Deterministic Model's performance.

over the test set of Slakh2100, using a sliding window with chunks of 4 seconds in length and a 2-second overlap. Additionally, we filter out silent chunks and chunks consisting of only one source, due to the poor performance of SI-SDR$_\text{I}$ on such segments. Results are reported in Table 1 and show the following:

***(i)*** Our vanilla Deterministic model outperforms the baselines. We attribute this to the slightly larger size and improved architecture (self-attention instead of LSTM in Demucs).

***(ii)*** Adding a diffusion model further improved separation quality. We found that for inference using stochastic sampler presented in EDM [33] with $T = 5$ timesteps, $R = 2$ correction steps (10 actual steps) and a starting noise standard deviation of $\sigma_\text{max} = 0.01$ yielded optimal performance, boosting overall quality by approximately 1.2 dB. We also evaluated a double Deterministic model without diffusion to assess whether the improvement was due to increased model size or diffusion itself. As shown in the 5th row of the table, double Deterministic model offered only slight improvement over the single one, with the best performance around 150 epochs before overfitting occurred. This suggests that the improvement with diffusion models is not merely due to increased model size.

***(iii)*** Our CD method not only reduces denoising steps but also further improves quality. As observed in the last three rows of the table, CD with 1 step maintains the best performance of the Diffusion model. Furthermore, CD with 2 steps adds almost 1 dB improvement compared to the Diffusion model, demonstrating the "student beating the teacher" effect. To the best of our knowledge, we are the first to demonstrate a CD model outperforming diffusion without the use of GANs, which was demonstrated for image data in CTM [29]. Finally, the last row shows the best-performing CD model with $T = 4$ denoising steps, which provides an additional improvement over the 2-step CD, leading to a dramatic 3 dB improvement compared to Deterministic model and 6.5 dB improvement compared to the best baseline model, setting a new benchmark for music source separation on the Slakh2100 dataset.

**Diffusion vs CD** We conducted a hyperparameter search to evaluate the efficiency of CD model compared the diffusion at a lower number of steps. We performed sweeps with number of steps

Table 2: **Performance and efficiency comparison.** We compare the inference times for separating a single ∼12s mixture, number of parameters, and real-time factors (RTF) of our models with the baselines. The inference times of our models are multiplied by 4 as our model performs source extraction, requiring about this tabel4 iterations to generate all tracks from the mixture.

| Model | Inference Time (s) | # of parameters | RTF |
|---|---|---|---|
| Demucs (no shift trick) [6] | 0.1285 | 265.7M | 0.010 |
| Demucs + Gibbs (512 steps) [14] | $0.1285 \times 512 = 65.80$ | $\sim$ 265.7M | 5.529 |
| Demucs + Gibbs (256 steps) [14] | $0.1285 \times 256 = 32.9$ | $\sim$ 265.7M | 2.764 |
| ISDM (correction) [26] | $4.6 \times 4 = 18.4$ | 405M $\times 4$ | 1.546 |
| MSDM (correction) [26] | 4.6 | 405M | 0.386 |
| Deterministic Model | $0.0285 \times 4 = 0.114$ | 405M | 0.009 |
| Deterministic Model double | $0.0570 \times 4 = 0.228$ | 405M $\times 2$ | 0.019 |
| Diffusion $N \times R = 10$ steps | $0.3498 \times 4 = 1.399$ | 405M $\times 2$ | 0.117 |
| CD $T = 1$ steps | 0.0690 $\times 4 = 0.276$ | 405M $\times 2$ | 0.023 |
| CD $T = 2$ steps | 0.0963 $\times 4 = 0.385$ | 405M $\times 2$ | 0.032 |
| CD $T = 4$ steps | 0.1547 $\times 4 = 0.618$ | 405M $\times 2$ | 0.051 |

$T \in [1, 5]$ (Note: for the Diffusion model, the solver has correction steps and the number of actual denoising steps $T \times R$ is used) and starting noise level $\sigma_{\text{max}} \in [0.001, 80.0]$ for both models. Figure 2 presents the SI-SDRi Avg results for both models across these parameters, showing $\sigma_{\text{max}}$ on log scale. Although the Diffusion model shows some improvement over the Deterministic model benchmark (green dashed line at 21.22 dB), it does not perform optimally at higher noise levels and low step counts, particularly in the 1-2 step scenarios, peaking only at $T = 10$ actual steps (as reported in Table 1). In contrast, the CD model not only achieves performance equal to the Diffusion model's best with just one step, but also consistently outperforms the Diffusion model across all $\sigma_{\text{max}}$ values and steps.

**Speed and Model Efficiency** We compared our models with baselines in terms of speed and efficiency. Table 2 shows inference time, number of parameters, and real-time factor (RTF) for separating a 12-second mixture. Unlike Demucs, Demucs+Gibbs, and MSDM, which extract all 4 tracks simultaneously, our models extracts one source at a time, multiplying inference times by 4. Despite this, our Deterministic Model is the most efficient, with an inference time of 0.114 seconds and an RTF of 0.009, comparable to Demucs. We report the fastest configuration of Demucs without the shift trick, which slows down inference time as described in [6]. Our diffusion model, while slower with 1.399 seconds and an RTF of 0.117, still outperforms generative baselines like ISDM and MSDM, which are both parameter-heavy and slow. Demucs+Gibbs, requiring many inference steps, also shows slower performance. Our CD models, especially at $T = 1$, achieve an RTF of 0.023, only twice as slow as the discriminative models and comparable with double Deterministic model, while providing significant audio quality improvement. Notably, our best-performing CD model with 4 steps, though 5 times slower than the Deterministic model, remains still significantly faster than other diffusion-based models (including ours), far below the real-time generation benchmark.

## 5 Conclusion

In this work, we introduced a framework that integrates discriminative and generative models for time-domain musical source extraction by combining a score-matching diffusion model with a Deterministic model to enhance extraction quality. To overcome the typically slow sampling process of diffusion models, we applied Consistency Distillation, which accelerated sampling to speeds comparable to the Deterministic model without compromising quality, while offering significant improvements in exchange for a slight speed trade-off. Notably, this work not only represents the first application of consistency models in the audio waveform domain but also achieves significant improvements across objective metrics, establishing new state-of-the-art source extraction/separation results on the Slakh2100 dataset. These results underscore the strong potential of generative models for advancing the field of source separation.

## References

[1] Woosung Choi, Minseok Kim, Jaehwa Chung, and Soonyoung Jung. Lasaft: Latent source attentive frequency transformation for conditioned source separation. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 171–175. IEEE, 2021.

[2] Alexandre Défossez. Hybrid spectrogram and waveform source separation. In *Proceedings of the ISMIR 2021 Workshop on Music Source Separation*, 2021.

[3] Francesc Lluís, Jordi Pons, and Xavier Serra. End-to-end music source separation: Is it possible in the waveform domain? In *INTERSPEECH*, pages 4619–4623, 2019.

[4] 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.

[5] 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.

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

[7] Naoya Takahashi, Nabarun Goswami, and Yuki Mitsufuji. Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation. In *Proc. IWAENC*, pages 106–110, 2018.

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

[9] Y Cem Subakan and Paris Smaragdis. Generative adversarial source separation. In *Proc. ICASSP*, pages 26–30. IEEE, 2018.

[10] 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 *Proc. IJCAI*, page 2747–2753. AAAI Press, 2019.

[11] Vivek Narayanaswamy, Jayaraman J. Thiagarajan, Rushil Anirudh, and Andreas Spanias. Unsupervised audio source separation using generative priors. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2020-October:2657–2661, 2020.

[12] Vivek Jayaram and John Thickstun. Parallel and flexible sampling from autoregressive models via langevin dynamics. In *Proc. ICML*, pages 4807–4818. PMLR, 2021.

[13] Emilian Postolache, Giorgio Mariani, Michele Mancusi, Andrea Santilli, Luca Cosmo, and Emanuele Rodolà. Latent autoregressive source separation. In *Proc. AAAI*, AAAI Press, 2023.

[14] Ethan Manilow, Curtis Hawthorne, Cheng-Zhi Anna Huang, Bryan Pardo, and Jesse Engel. Improving source separation by explicitly modeling dependencies between sources. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 291–295. IEEE, 2022.

[15] Emilian Postolache, Jordi Pons, Santiago Pascual, and Joan Serrà. Adversarial permutation invariant training for universal sound separation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.

[16] Ilya Kavalerov, Scott Wisdom, Hakan Erdogan, Brian Patton, Kevin Wilson, Jonathan Le Roux, and John R Hershey. Universal sound separation. In *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 175–179. IEEE, 2019.

[17] Scott Wisdom, Efthymios Tzinis, Hakan Erdogan, Ron Weiss, Kevin Wilson, and John Hershey. Unsupervised sound separation using mixture invariant training. *Advances in Neural Information Processing Systems*, 33:3846–3857, 2020.

[18] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis R. Bach and David M. Blei, editors, *Proceedings ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2256–2265. JMLR.org, 2015.

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 6840–6851, 2020.

[20] 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)*. IEEE, 2023.

[21] Shahar Lutati, Eliya Nachmani, and Lior Wolf. Separate and diffuse: Using a pretrained diffusion model for better source separation. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

[22] Chaorui Huang, Susan Liang, Yapeng Tian, Anurag Kumar, and Chenliang Xu. Davis: High-quality audio-visual separation with generative diffusion models. arXiv:2308.00122, 2023.

[23] Chin-Yun Yu, Emilian Postolache, Emanuele Rodolà, and György Fazekas. Zero-shot duet singing voices separation with diffusion models. arXiv:2311.07345, 2023.

[24] Masato Hirano, Kazuki Shimada, Yuichiro Koyama, Shusuke Takahashi, and Yuki Mitsufuji. Diffusion-based signal refiner for speech separation. *arXiv preprint arXiv:2305.05857*, 2023.

[25] Genís Plaja-Roglans, Miron Marius, and Xavier Serra. A diffusion-inspired training strategy for singing voice extraction in the waveform domain. In *Proc. of the 23rd Int. Society for Music Information Retrieval*, 2022.

[26] 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, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

[27] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11895–11907, 2019.

[28] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 32211–32252. PMLR, 2023.

[29] Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

[30] 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 *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019.

[31] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, pages 334–340, 2018.

[32] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. The MUSDB18 corpus for music separation, December 2017.

[33] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, 2022.

[34] Jonathan Le Roux, Scott Wisdom, Hakan Erdogan, and John R. Hershey. Sdr – half-baked or well done? In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 626–630, 2019.

[35] Emad M. Grais, Mehmet Umut Sen, and Hakan Erdogan. Deep neural networks for single channel source separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*, pages 3734–3738. IEEE, 2014.

[36] Stefan Uhlich, Franck Giron, and Yuki Mitsufuji. Deep neural network based instrument extraction from music. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*, pages 2135–2139. IEEE, 2015.

[37] Stefan Uhlich, Marcello Porcu, Franck Giron, Michael Enenkl, Thomas Kemp, Naoya Takahashi, and Yuki Mitsufuji. Improving music source separation based on deep neural networks through data augmentation and network blending. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*, pages 261–265. IEEE, 2017.

[38] Jen-Yu Liu and Yi-Hsuan Yang. Denoising auto-encoder with recurrent skip connections and residual regression for music source separation. In M. Arif Wani, Mehmed M. Kantardzic, Moamar Sayed Mouchaweh, João Gama, and Edwin Lughofer, editors, *17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018, Orlando, FL, USA, December 17-20, 2018*, pages 773–778. IEEE, 2018.

[39] Naoya Takahashi and Yuki Mitsufuji. Multi-scale multi-band densenets for audio source separation. In *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA 2017, New Paltz, NY, USA, October 15-18, 2017*, pages 21–25. IEEE, 2017.

[40] Aditya Arie Nugraha, Antoine Liutkus, and Emmanuel Vincent. Multichannel music separation with deep neural networks. In *24th European Signal Processing Conference, EUSIPCO 2016, Budapest, Hungary, August 29 - September 2, 2016*, pages 1748–1752. IEEE, 2016.

[41] Naoya Takahashi and Yuki Mitsufuji. D3net: Densely connected multidilated densenet for music source separation. *arXiv preprint arXiv:2010.01733*, 2020.

[42] Romain Hennequin, Anis Khlif, Félix Voituret, and Manuel Moussallam. Spleeter: a fast and efficient music source separation tool with pre-trained models. *J. Open Source Softw.*, 5(56):2154, 2020.

[43] Aäron 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. In *Proc. 9th ISCA Workshop on Speech Synthesis Workshop (SSW 9)*, page 125, 2016.

[44] Hakan Erdogan, Scott Wisdom, Xuankai Chang, Zalán Borsos, Marco Tagliasacchi, Neil Zeghidour, and John R. Hershey. Tokensplit: Using discrete speech representations for direct, refined, and transcript-conditioned speech separation and recognition. In *24th Annual Conference of the International Speech Communication Association, Interspeech 2023, Dublin, Ireland, August 20-24, 2023*, pages 3462–3466. ISCA, 2023.

[45] Jean-Marie Lemercier, Julius Richter, Simon Welker, and Timo Gerkmann. Storm: A diffusion-based stochastic regeneration model for speech enhancement and dereverberation. *IEEE ACM Trans. Audio Speech Lang. Process.*, 31:2724–2737, 2023.

[46] Jean-Marie Lemercier, Joachim Thiemann, Raphael Koning, and Timo Gerkmann. Wind noise reduction with a diffusion-based stochastic regeneration model. *In Speech Communication; 15th ITG Conference*, pages 116–120, 2023.

[47] 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.

[48] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2020.

[49] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In *NeurIPS*, 2022.

[50] Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *ICLR*, 2024.

[51] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[52] Flavio Schneider, Ojasv Kamal, Zhijing Jin, and Bernhard Schölkopf. Moûsai: Efficient text-to-music diffusion models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 8050–8068. Association for Computational Linguistics, 2024.

[53] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *ICLR*, 2020.

## A Related Work

Audio source separation has seen significant advances, driven by the development of deep learning techniques and the availability of large-scale datasets. A significant body of work in source separation operates on time-frequency representations like the Short-Time Fourier Transform (STFT). The field of audio source separation initially focused on speech separation [35], and later expanded to include music [36–40]. State-of-the-art models for music source separation in the spectrogram domain include MMDenseLSTM [7], which combines convolutional and recurrent networks, D3Net [41], which uses dilated convolutions, and Spleeter [42], based on a U-Net architecture.

In recent years, there has been a shift in focus towards models that operate directly on raw waveforms. The pioneering waveform-based audio separation model [3] was built on the WaveNet architecture [43], followed by Wave-U-Net [31], which is based on the U-Net architecture. However, both models underperformed compared to spectrogram-based models at a time. Conv-TasNet [5], utilizing stacked dilated 1-D convolutional blocks, was the first waveform-based model to surpass spectrogram-based approaches in speech source separation. Demucs [6], inspired by music synthesis models, extended the U-Net architecture by incorporating bidirectional LSTM layers between the encoder and decoder. Building on Demucs, the source separation problem was reframed as an Orderless Neural Autoregressive Density Estimator (NADE) by Demucs+Gibbs [14], using a Gibbs sampling procedure to iteratively improve separation performance by conditioning each source estimate on those from previous steps. These latest two works serve as the primary baselines for our approach.

Similar to our work, MSDM [26] introduced a denoising score-matching [27] diffusion-based U-Net architecture for the musical sourse separation in the waveform domain. MSDM operates in a multichannel manner and is fully generative, proposing a novel inference-time conditioning scheme based on the Dirac delta function for posterior sampling. Unlike our approach and most separation models, MSDM is also capable of synthesizing music and creating arrangements, which makes it not a direct comparison to our work. Nevertheless, we include it in our baselines due to the architectural and methodological similarities.

A similar approach to ours, which combines discriminative and generative models, has recently been proposed in several works, primarily for speech separation, enhancement and generation [44–46].

## B Preliminary

### B.1 Score-based Diffusion Models

Score-based diffusion models [27], like other diffusion models [18, 19], are designed to learn data representations by introducing controlled noise and learning to iteratively remove it from the data. Score-based models are a class of generative models that rely on learning the gradient of the data distribution, known as the score function of the target distribution $p(x)$, namely $\nabla_x \log p(x)$.

Given a data point $x_0$ from the true data distribution $p(x_0)$, the model generates a sequence of noise-corrupted samples $x_t$ by adding Gaussian noise to $x_0$ as $x_t = x_0 + \sigma_t \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$ represents Gaussian noise, and $\sigma_t$ defines the amount of noise added at time $t$. In score-based diffusion, this process is typically continuous, and with the choice $\sigma_t = t$. Often, as in EDM [33], in training time $\sigma$ can be sampled from a distribution. As $t$ approaches $T$, where $T$ is a sufficiently large value, the process results in isotropic Gaussian noise, with $x_T \sim \mathcal{N}(0, I)$. As described by [47], the noise removal reverse process in diffusion can be formulated using a probability flow Ordinary Differential Equation (ODE) as follows:

$$\mathrm{d}x_t = \sigma_t \nabla_{x_t} \log p(x_t)\, \mathrm{d}t. \tag{7}$$

By training a neural network to approximate the score function, we can enable a generative process that generates data from noise. For this, the score function is often defined as $\nabla_{x_t} \log p(x_t) = \frac{g_\theta(x_t, \sigma_t) - x_t}{\sigma_t^2}$, where $g_\theta$ is the neural network. The model can be trained with the Denoising Score Matching (DSM) loss, given by:

$$\mathcal{L}(\theta) = \mathbb{E}_{t,x} \left[ \lambda(\sigma_t) \left\| g_\theta(x_t, \sigma_t) - x_0 \right\|_2^2 \right], \tag{8}$$

where $\lambda(\sigma_t)$ is a noise level dependent loss weighting function.

**Algorithm 1** Heun's 2nd order deterministic sampler.

```
1:  procedure HEUN SAMPLER(Solver$_k(x_t, \sigma_t; g_\phi)$)
2:      sample $x_T \sim \mathcal{N}(\mathbf{0},\ \sigma^2_{max}\ \mathbf{I})$      ▷ Generate initial sample at $T$ ($\sigma_T = \sigma_{\text{max}}$)
3:      for $t = T, \ldots, 1$ do                                  ▷ Solve over $T$ time steps
4:          $d \leftarrow \frac{x_t - g_\phi(x_t, \sigma_t)}{\sigma_t}$                          ▷ Evaluate derivative at $t$
5:          $x_{t-1} \leftarrow x_t + (\sigma_{t-1} - \sigma_t)d$                 ▷ Euler step from $t$ to $t-1$
6:          if $\sigma_{t-1} \neq 0$ then              ▷ Apply 2nd order correction unless next $\sigma$ is zero
7:              $d' \leftarrow \frac{x_{t-1} - g_\phi(x_{t-1}, \sigma_{t-1})}{\sigma_{t-1}}$                 ▷ Evaluate derivative at $t-1$
8:              $x_{t-1} \leftarrow x_t + (\sigma_{t-1} - \sigma_t)\frac{1}{2}(d + d')$      ▷ Explicit trapezoidal rule at $t-1$
9:          end if
10:     end for
11:     return $x_0$                      ▷ Return noise-free last sample at $t = 1$, gives $x_0$
12: end procedure
```

Additionally, for effective training of the network, it is advisable to keep input and output signal magnitudes fixed, for example, to unit variance. To avoid large variations in gradient magnitudes, a common practice is not to represent $g_\theta$ as a neural network directly, but to train a different network, $g'_\theta$, from which $g_\theta$ is derived. EDM describes a preconditioning setup defined as:

$$g_\theta(x_t; \sigma_t) = c_{\text{skip}}(\sigma_t)x_t + c_{\text{out}}(\sigma_t)g'_\theta\big(c_{\text{in}}(\sigma_t)x_t, c_{\text{noise}}(\sigma_t)\big), \tag{9}$$

where $g'_\theta$ is a neural network. The preconditioning parameters regulate the network's skip connection to $x_t$, outputs, inputs, and noise levels respectively as follows:

$$\begin{aligned} c_{\text{skip}}(\sigma_t) &= \frac{\sigma^2_{\text{data}}}{\sigma^2_t + \sigma^2_{\text{data}}}, & c_{\text{out}}(\sigma_t) &= \frac{\sigma_t \cdot \sigma_{\text{data}}}{\sqrt{\sigma^2_{\text{data}} + \sigma^2_t}}, \\ c_{\text{in}}(\sigma_t) &= \frac{1}{\sqrt{\sigma^2_t + \sigma^2_{\text{data}}}}, & c_{\text{noise}}(\sigma_t) &= \frac{1}{4}\ln(\sigma_t). \end{aligned} \tag{10}$$

where $\sigma_{\text{data}}$ is the expected standard deviation of the clean data.

In the time of inference, solving an ODE numerically requires approximating the true solution trajectory. In the reverse diffusion process, $\sigma_t$ is a discrete function with finite steps $t \in [1, T]$, such that $\sigma_T = \sigma_{\text{max}}$ and $\sigma_1 = \sigma_{\text{min}}$. (Note: often in the score-based diffusion literature, this parametrization is reversed, with $\sigma_1 = \sigma_{\text{max}}$ and $\sigma_T = \sigma_{\text{min}}$; however, to be consistent with consistency distillation, which also uses this noise scheduler, we adopt the reverse scheduling paradigm.) EDM [33] introduced an effective non-linear schedule for time discretization, given by:

$$\sigma_t = \left(\sigma_{\text{min}}^{\frac{1}{\rho}} + \frac{t-1}{T-1}\left(\sigma_{\text{max}}^{\frac{1}{\rho}} - \sigma_{\text{min}}^{\frac{1}{\rho}}\right)\right)^{\rho}, \tag{11}$$

where $\rho$ controls the curvature or "bend" of the noise schedule.

There are many ODE solvers in the score-based diffusion literature. ODE solvers can be categorized by their order of accuracy, deterministic vs. stochastic nature, etc. For instance, DDIM [48] and the Euler sampler [47] correspond to 1st-order deterministic solvers, while EDM [33] introduces a deterministic 2nd-order Heun solver. The DPM2 sampler [49] uses a 2nd-order deterministic solver, while Euler-Maruyama [47] represents a first-order stochastic solver.

Following CTM [29], we denote solvers with $\texttt{Solver}_k(x_t, \sigma_t, [c]; g_\phi)$, where $c$ is a placeholder for conditioning information, $x_t$ is the starting sample, $g_\phi$ is a neural network trained to approximate the score, $\sigma_t$ is the current noise level, and $k$ is the number of steps the solver performs (solver produces clean sample from pure noise $X_T$ when $k = T$). In the Algorithms 1 and 2 we show examples of deterministic and stochastic solvers used in our experiments.

In Algorithm 1, we present Heun's 2nd order deterministic sampler, which applies a two-step correction mechanism for sampling from a diffusion model. It begins by generating an initial sample at the maximum noise level, $\sigma_{\text{max}}$, and iteratively reduces noise over time using an Euler step followed

**Algorithm 2** EDM sampler with stochasticity and correction mechanism.

```
1:  procedure EDM SAMPLER($\texttt{Solver}_k(x_t, \sigma_t; g_\phi)$)
2:      sample $x = x_T \sim \mathcal{N}(\mathbf{0},\ \sigma^2_{max}\ \mathbf{I})$      ▷ Generate initial sample at $T$ ($\sigma_T = \sigma_{\text{max}}$)
3:      $\gamma \leftarrow \min(S_{\text{churn}}/T, \sqrt{2} - 1)$      ▷ Calculate $\gamma$ value that controls increased noise levels
4:      for $t = T, \ldots, 1$ do      ▷ Solve over $T$ time steps
5:          for $r = R - 1, \ldots, 0$ do      ▷ Iterate over $R$ correction steps
6:              $\hat{\sigma} \leftarrow \sigma_t \cdot (\gamma + 1)$      ▷ Calculate temporarily increased noise level $\hat{\sigma}$
7:              sample $\epsilon \sim \mathcal{N}(0, I)$      ▷ Sample noise
8:              $\hat{x} \leftarrow x + \sqrt{\hat{\sigma}^2 - \sigma_t^2}\epsilon$      ▷ Add noise to move to temporarily increased noise level
9:              $d \leftarrow \frac{\hat{x} - g_\phi(\hat{x}, \hat{\sigma})}{\hat{\sigma}}$      ▷ Evaluate derivative at $\hat{t}$
10:             $x \leftarrow \hat{x} + (\sigma_{t-1} - \hat{\sigma})d$      ▷ Euler step from $\hat{t}$ to $t - 1$
11:             if $r > 0$ then      ▷ If not last resample step
12:                 sample $\epsilon \sim \mathcal{N}(0, I)$      ▷ Sample noise
13:                 $x \leftarrow x + \sqrt{\sigma_t^2 - \sigma_{t-1}^2}\epsilon$      ▷ Renoise
14:             end if
15:         end for
16:     end for
17:     return $x$      ▷ Return noise-free sample after last step $t = 1$
18: end procedure
```

by a 2nd order correction, based on the trapezoidal rule. This approach improves the accuracy of the generated samples by accounting for the slope at both the current and the next time step, ensuring more precise sampling throughout the process.

Algorithm 2 describes the EDM sampler (also used in MSDM with a little variations). In this sampler, the stochasticity is controlled by the parameter $S_{\text{churn}}$ and it introduces a correction mechanism with $R$ steps for adjusting the trajectories of generated samples. A $\gamma$ value, calculated using $S_{\text{churn}}$, temporarily increases the noise at each step, followed by a resampling procedure that helps correct errors in the diffusion process. The EDM sampler includes $R$ iterations at each time step, resulting in $T \times R$ actual steps in the sampler.

### B.2 Consistency Models

Consistency models [28] are a novel class of generative models closely related to diffusion models, designed for few-step or even one-step generation. The core idea of consistency models is the concept of self-consistency. Consistency function $g$ directly connects any timestep point of the diffusion trajectory to the trajectory's starting point $g(x_t, t) \to x_\delta$, where $\delta \to 0^+$ represents an infinitesimally small positive value, indicating the very low noise step at the start of the trajectory. This property of the consistency function can also be written as $g(x_t, t) = g(x_{t'}, t')$, $\forall t, t' \in [\delta, T]$, indicating that any noise level on the given trajectory results in the same output.

The consistency function is typically learned with a deep neural network $g_\omega$ from the data, enforcing the self-consistency property across all timesteps. Consistency models are trained to perform single-step denoising from any time step $t$ to 0, either through Consistency Distillation (CD) [28], distilling pretrained teacher diffusion model, or from scratch [50]. Similar to inference of score-based models, the time horizon of CD is discretized into $T$ steps and noise levels $\sigma$ are defined following the function the Eq. 11 with boundaries $\sigma : [1, T] \to [\sigma_{\text{min}}, \sigma_{\text{max}}]$. For the consistency function to hold it's *boundary condition*, which is $g_\omega(x_1, \sigma_{\text{min}}) = x_1 = x_0 + \epsilon\sigma_{\text{min}}$ the consistency model $g_\omega$, is parameterized as follows:

$$g_\omega(x_t, \sigma_t) = c_{\text{skip}}(\sigma_t)x_t + c_{\text{out}}(\sigma_t)g'_\omega(x_t, \sigma_t), \tag{12}$$

where $g'_\omega$ is a deep neural network. The preconditioning $c_{\text{skip}}(\sigma_t)$ and $c_{\text{out}}(\sigma_t)$ are differentiable functions, and are usually chosen as:

$$c_{\text{skip}}(\sigma_t) = \frac{\sigma^2_{\text{data}}}{(\sigma_t - \sigma_{\text{min}})^2 + \sigma^2_{\text{data}}}, \quad c_{\text{out}}(t) = \frac{\sigma_{\text{data}}(\sigma_t - \sigma_{\text{min}})}{\sqrt{\sigma^2_{\text{data}} + t^2}}, \tag{13}$$

which satisfies $c_{\text{skip}}(\sigma_{\min}) = 1$ and $c_{\text{out}}(\sigma_{\min}) = 0$.

In the original CD [28], the optimization of the consistency model $g_\omega$ is done through minimizing discrepancies between successive estimates of the data at different timesteps. The process begins with a noisy data point $x_t$. The target for the loss is first calculated by taking a single denoising step with an ODE solver, which uses the pretrained score-matching diffusion model $g_\phi$, acting as a teacher, obtaining an estimate of the data at the $t-1$ step, $\hat{x}^{\phi}_{t-1} = \texttt{Solver}_1(x_t, \sigma_t; g_\phi)$. Then, the stop-gradient EMA updated copy of the student model performs a step from $t-1$ to 0. The estimate for the loss is calculated by the student model $g_\omega$, which directly jumps from $t$ to 0. The loss is then defined as follows:

$$\mathcal{L}_{\mathcal{CD}}(\omega) = \mathbb{E}_{x,t}\Big[d\big(\underbrace{g_{\texttt{sg}(\omega)}(\hat{x}^{\phi}_{t-1}, \sigma_{t-1})}_{\textit{target}}, \underbrace{g_\omega(x_t, \sigma_t)}_{\textit{estimate}}\big)\Big], \tag{14}$$

where $\texttt{sg}(\theta)$ denotes the stop-gradient running EMA updated $g_\omega$ during optimization, and $d(\cdot,\cdot)$ is any function used to measure the distance between the model's predictions.

The Consistency Trajectory Model (CTM) [29] presented a unified approach that integrates score-based and consistency distillation models by allowing both infinitesimally small and long-step jumps along the Probability Flow ODE trajectory. CTM trains a neural network $g_\omega(x_t, \sigma_t, \sigma_{t'})$ to predict the solution of the ODE from an initial time $t$ to a final time $t'$, with $t' < t$, enabling any-step-to-any-step predictions. The preconditioning for the network is as follows:

$$g_\omega(x_t, \sigma_t, \sigma_{t'}) = \frac{\sigma_{t'}}{\sigma_t} x_t + \Big(1 - \frac{\sigma_{t'}}{\sigma_t}\Big) g'_\omega(x_t, \sigma_t, \sigma_{t'}), \tag{15}$$

where $g'_\omega$ is a neural network, and thus $g_\omega$ automatically satisfies the initial boundary condition. Using same discrete noise scheduling, we have $g_\omega(x_1, \sigma_{\min}, \sigma_{\min}) = x_1 = x_0 + \epsilon\sigma_{\min}$.

For the distillation loss in CTM, the teacher model first takes steps from $t$ to $u$, where the noise level $u$ is randomly chosen between the start and end as $u \in [t', t)$. Thus, we obtain an estimate $\hat{x}^{\phi}_{u} = \texttt{Solver}_{t-u}(x_t, \sigma_t; g_\phi)$. Then, the stop-gradient student model proceeds and jumps from $u$ to $t'$, followed by a step to 0 to calculate the target. For the estimate, the student model directly transitions from $t$ to $t'$, and similarly proceeds to 0 using the stop-gradient model. The optimization of the consistency model $g_\omega$ is done through minimizing the loss, which is calculated as follows:

$$\mathcal{L}_{\text{CTM}}(\omega) = \mathbb{E}_{t,t',u}\Big[d\big(\underbrace{g_{\texttt{sg}(\omega)}(g_{\texttt{sg}(\omega)}(\hat{x}^{\phi}_{u}, \sigma_u, \sigma_{t'}), \sigma_{t'}, 0)}_{\textit{target}}, \underbrace{g_{\texttt{sg}(\omega)}(g_\omega(x_t, \sigma_t, \sigma_{t'}), \sigma_{t'}, 0)}_{\textit{estimate}}\big)\Big]. \tag{16}$$

Additionally, in the CTM framework, two auxiliary losses are used. First, the model is trained to approximate infinitesimally small neural jumps when $t' \to t$, using the DSM loss:

$$\mathcal{L}_{\text{DSM}}(\omega) = \mathbb{E}_t\left[\|x_0 - g_\omega(x_t, \sigma_t, \sigma_t)\|_2^2\right]. \tag{17}$$

The second auxiliary loss is introduced by incorporating an additional GAN component into the system, introducing the GAN loss, and making the final loss a weighted sum of these terms:

$$\mathcal{L}(\omega, \eta) = \mathcal{L}_{\text{CTM}}(\omega) + \lambda_{\text{DSM}}\mathcal{L}_{\text{DSM}}(\omega) + \lambda_{\text{GAN}}\mathcal{L}_{\text{GAN}}(\omega, \eta), \tag{18}$$

where $\lambda_{\text{DSM}}$ and $\lambda_{\text{GAN}}$ are adaptive weights to stabilize training by balancing the gradient scale of each term.

## C Additional Experimental Details

### C.1 Dataset

For our experiments, we used the Slakh2100 dataset, a widely recognized and extensively used benchmark for music source separation. Beyond its compatibility with baseline models and general quality, our choice was driven by the high data requirements of diffusion models, making Slakh an ideal candidate for training. Slakh consists of 2100 tracks, with 1500 allocated for training, 375 for validation, and 225 for testing. It is a synthetically generated multi-track audio waveform dataset created from MIDI files using virtual instruments. The dataset contains 31 instrument classes. We downsampled the audio to 22kHz and used a duration of approximately 11.9 seconds per audio file. We focused on four instruments—Bass, Drums, Guitar, and Piano—due to their prevalence in the dataset and to ensure direct comparability with baseline models. The presence of these instruments in the dataset is 94.7% for Bass, 99.3% for Drums, 100.0% for Guitar, and 99.3% for Piano.

In addition to Slakh2100, we also trained and evaluated our models using the MUSDB18 [32] dataset, which is another widely used data for music source separation. The dataset consists of 150 tracks, with 100 designated for training and 50 for testing, totaling approximately 10 hours of professional-grade audio. Each track is divided into stems: Bass, Drums, Vocals, and Other. For the results on MUSDB18 dataset please refer to Appendix D.

## C.2 Deterministic Model

For our Deterministic Model, we adopted U-Net architecture consisting of 1D CNNs with self-attention and enhanced convolutional blocks. The U-Net, originally introduced for biomedical imaging [51], consists of convolutional layers in both the encoder and decoder, connected by skip connections. Our design of U-Net operates in the audio waveform domain and follows the structure of generative U-Net models from Moûsai [52] and MSDM [26], with a minor modifications of adding a one-hot encoded instrument label as a conditioning input.

We adopted the hyperparameter setting used by MSDM [26], with the modification of using a single channel instead of four and using U-Net without a diffusion part. The model implementation is based on the publicly available repository `audio-diffusion-pytorch/v0.0.432`[†]. Our U-Net encoder consists of six layers, each containing two convolutional ResNet blocks. The first three layers do not incorporate attention, while the last three include multi-head attention with 8 heads and 128 attention features. The downsampling factor is 4 in the first three layers and 2 in the last three layers. The number of channels in the encoder is [256, 512, 1024, 1024, 1024, 1024]. The bottleneck includes a ResNet block, followed by a self-attention mechanism, and another ResNet block, all maintaining 1024 channels. The decoder mirrors the encoder, following a symmetric structure in reverse. We trained the Deterministic Model model using the Adam optimizer with a learning rate of $1 \times 10^{-4}$ for 170 epochs until convergence.

## C.3 Diffusion Model

For the diffusion, we used the same architecture and set of hyperparameters for the backbone U-Net as in the Deterministic Model, wrapping it within the diffusion model framework. The standard deviation of the data was set to $\sigma_{\text{data}} = 0.2$. During training, we sampled $\sigma$ from a log-normal distribution with $\ln(\sigma) \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}}^2)$, where $P_{\text{mean}} = -3$ and $P_{\text{std}} = 1$. The loss weighting was performed using $\lambda(\sigma_t) = \frac{1}{c_{\text{out}}(\sigma_t)^2}$, where $c_{\text{out}}(\sigma_t)$ is taken from Eq. (10). We trained the diffusion model conditioned with the Deterministic model features, using the Adam optimizer with a learning rate of $1 \times 10^{-4}$ for 280 epochs until convergence.

For inference, we found the EDM sampler, given in Algorithm 2, to be most effective for our diffusion model. In our experiments, we set the minimum noise parameter to $\sigma_{\min} = 0.0001$ and found that $\rho = 9$ and $S_{\text{churn}} = 20.0$ were optimal. The best results for our diffusion model were achieved with $T = 5$ sampling steps, $R = 2$ correction steps, and $\sigma_{\max} = 0.01$. Additionally, the parameter $\sigma_{\max}$ was varied between 0.001 and 80, and the impact of these values can be observed in Figure 2 in the results section.

## C.4 Consistency Distillation

We used the same architecture for the consistency models as in the diffusion models. For effective distillation, we initialized the consistency model with pre-trained diffusion model weights and trained it using the Rectified Adam optimizer [53], without applying learning rate decay, warm-up, or weight

[†]`https://github.com/archinetai/audio-diffusion-pytorch/tree/v0.0.43`

decay, and with a fixed learning rate of $1 \times 10^{-5}$. Additionally, we applied an Exponential Moving Average (EMA) with a value of 0.9999 to the weights of the student model. The $\sigma$-schedule for the DSM loss followed a half-lognormal distribution with $P_{\text{mean}} = -3$ and $P_{\text{std}} = 1$, where half of the batch in each iteration was sampled from this lognormal distribution, and the other half was sampled by uniformly selecting $t$ and calculating $\sigma_t$ from Eq. (11). Aslo, noise scheduling for the CD loss fully adhered to the discrete form in Eq. (11). We set $\sigma_{\text{min}} = 0.0001$, $\sigma_{\text{max}} = 10.0$ and $\rho = 9.0$ for the CD training. For learning trajectories, a Heun's 2nd-order deterministic ODE solver, shown in Algorithm 1, was used with the teacher model. The number of ODE steps was uniformly sampled from $h \sim U[1, 17]$. We experimented with weight scheduling for both the $\mathcal{L}_{\text{CD}}$ and $\mathcal{L}_{\text{DSM}}$ losses. We found that using uniform weighting for $\mathcal{L}_{\text{CD}}$ and applying $\lambda(\sigma_t) = \frac{1}{c_{\text{out}}(\sigma_t)^2}$ (as in the diffusion model from Eq. (10)) for $\mathcal{L}_{\text{DSM}}$ provided the best training stability and performance. Additionally, we tested adaptive balance weighting between these losses, as defined in Eq. (6), and found that setting $\lambda_{\text{DSM}} = 1$ was optimal for our CD training.

For inference, we employed the *onestep* and *multistep* algorithms, as described in [28]. However, as mentioned earlier, instead of initializing the diffusion process with pure noise $x_T \sim \mathcal{N}(0, \sigma_{\text{max}}^2 \mathbf{I})$, we start with a sum of noise and the output $\hat{x}_s^{\text{det}}$ of the Deterministic model as: $x_T = \hat{x}_s^{\text{det}} + \sigma_{\text{max}} \cdot \epsilon$, where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$. As evident from our results, small values of $\sigma_{\text{max}} \in [0.001, 0.5]$ resulted in the best performance during the inference process.

Table 3: **Comparison of results of out model with MSDM, Demucs v2 and Demucs + Gibbs on MUSDB18 test set [32].** We report the SI-SDR$_\text{I}$ values in dB (higher is better).

| Model | Trained on | Bass | Drums | Other | Vocals | All |
|---|---|---|---|---|---|---|
| Demucs v2 [26] | MUSDB | 13.28 | 11.53 | 8.59 | 16.80 | 12.55 |
| MSDM [26] | MUSDB | 4.87 | 3.28 | 1.97 | 6.83 | 4.24 |
| Demucs (as reported in [14]) | MUSDB | 6.80 | 5.25 | -1.30 | 3.90 | 3.66 |
| Demucs + Gibbs (as reported in [14]) | MUSDB | 9.00 | 7.15 | 1.00 | 11.30 | 7.61 |
| MSDM [26] | Slakh | -0.83 | -0.94 | - | - | -0.88 |
| Deterministic Model | Slakh | 3.55 | 5.19 | - | - | 4.37 |
| Diffusion | Slakh | 3.50 | 5.15 | - | - | 4.32 |
| CD 1 step | Slakh | 3.54 | 5.14 | - | - | 4.34 |
| CD 2 step | Slakh | 3.83 | 5.04 | - | - | 4.43 |
| CD 4 step | Slakh | 3.70 | 4.90 | - | - | 4.30 |
| Deterministic Model | MUSDB | 9.93 | 8.19 | 4.59 | 12.32 | 8.75 |
| Diffusion ($\sigma_{\text{max}} = 0.0001$) | MUSDB | 8.75 | 7.14 | 3.82 | 11.82 | 7.88 |

## D Results on MUSDB dataset

Reported in the Table 3, we compared the performance of our model with MSDM, Demucs v2, and Demucs + Gibbs on the MUSDB18 [32] test set, reporting SI-SDR$_\text{I}$ values (dB). First, we evaluated our model trained on Slakh2100 without any training or fine-tuning on MUSDB18, reporting results only for bass and drums, as our model is not trained on the other and vocals categories in this setting. We found that our model, when evaluated on the MUSDB18 test set, outperformed MSDM trained on Slakh2100 and even MSDM trained on MUSDB18 (as reported in their paper). However, we observed that the diffusion and CD extensions of our model did not show any additional improvement over the Deterministic model on the MUSDB dataset.

Next, we trained our Deterministic model on the MUSDB18 dataset and found that it outperformed both Demucs and Demucs + Gibbs as reported in their work [14]. However, our Deterministic model did not outperform Demucs v2, as reported in the MSDM paper. Furthermore, when we trained the diffusion model on the MUSDB dataset, we observed no improvement over the Deterministic model. Although, due to our architecture, infinitesimally small $\sigma_{\text{max}}$ values should at least allow the diffusion model to achieve performance comparable to the Deterministic model, our results revealed a 1 dB decrease in extraction quality even when using $\sigma_{\text{max}} = 0.0001$. This suggests that the diffusion-based approach does not work in the case of the MUSDB dataset.

Although our model performed somewhat adequately on MUSDB18 dataset, we attribute these results to the relatively small size of MUSDB18 (10 hours compared to 145 hours in the Slakh2100 dataset), which was insufficient to fully exploit the potential of our diffusion-based systems. We reserve the adaptation of our approach to smaller dataset size for future work.