SAPIENZA
UNIVERSITÀ DI ROMA

Sapienza University of Rome

Department of Computer Science
PhD in Computer Science

Thesis For The Degree Of Doctor Of Philosophy

From Source Separation to Compositional Music Generation

Thesis Advisor
Prof. Emanuele Rodolà

Candidate
Emilian Postolache
1649271

Academic Year MMXX-MMXXIII (XXXVI cycle)

To my father, who believed no dream is unreachable.

Abstract

This thesis proposes a journey into sound processing through deep learning, particularly generative models, exploring the compositional structure of sound, which is layered in different sources that compose the final auditory experience. The first part of the text focuses on the problem of separating the sources from mixtures, initially using a deterministic separator trained via adversarial losses in a permutation invariant manner and then exploring the setting of Bayesian inference through the use of latent autoregressive models. In the second half of the thesis, we focus on the continuous musical setting (as opposed to symbolic), where the sources that compose the sound are interdependent. By modeling this interdependence probabilistically, we develop diffusion models that allow for the compositional processing of the different stems present in tracks, thus not only separating them but generating them in a conditioned manner (accompaniments). Subsequently, we generalize these models to text-conditioned diffusion models without requiring supervised data. We conclude the thesis by discussing possible developments in the compositional generation of audio.

Keywords: source separation, music generation, generative models, bayesian inference.

Contents

List of Figures

List of Tables

Nomenclature

$\mathbf{x}'$	General notation for both individual sources and mixtures
$\mathbf{x}, \mathbf{X}, \boldsymbol{\xi}$	Sources in time, STFT and latent domains
$\mathbf{y}, \mathbf{Y}, \boldsymbol{\upsilon}$	Mix in time, STFT and latent domains
$\mathbf{z}$	Conditioning vector (e.g., labels, text embeddings)
L, S	Audio length in time domain and latent domain
N, N'	Number of sources estimated by a model and present in a mix

Acronyms

CASA Computational Auditory Scene Analysis

FAD Fréchet Audio Distance

FID Fréchet Inception Distance

FUSS Free Universal Sound Separation

GAN Generative Adversarial Networks

GMSDI Generalized Multi-Source Diffusion Inference

ISDM Independent Source Diffusion Model

LASS Latent Autoregressive Source Separation

MSDM Multi-Source Diffusion Model

NMF Non-negative Matrix Factorization

PIT Permutation Invariant Training

PSNR Peak Signal to Noise Ratio

SDR Signal to Distortion Ratio

SI-SDR Scale-Invariant SDR

SI-SDR$_\mathbf{I}$ Scale-Invariant SDR Improvement

SI-SDR$_\mathbf{S}$ Scale-Invariant SDR Single

STFT Short Time Fourier Transform

TDCN++ (improved) Time-Dilated Convolutional Network

VQ-GAN Vector-Quantized GAN Variational Autoencoder

VQ-VAE Vector-Quantized Variational Autoencoder

Chapter 1

Introduction

Large-scale deep learning-based artificial intelligence has changed the way we perform everyday tasks. For example, OpenAI's GPT-4 [4] allows the generation of texts of any kind, being a versatile tool for composition and processing. More recently, the use of AI has also conquered the video domain. The Sora model [5] can generate high-resolution photorealistic videos up to a minute long by simply providing the model with a text prompt describing the video. A fundamental aspect of human experience is the perception of sound. Therefore, it is unsurprising that increasingly sophisticated models can improve the analysis [6, 7] and generation [8–10] of such signals.

The most spectacular application of these new models is music generation based on a prompt. For example, on the app `suno.ai`, we can enter any lyric, along with a semantic description of the track (e.g., experimental edm pop), to obtain a realistic song faithful to our requests. Models underlying apps like these can be trained on a large scale since they take as inputs full songs, along with some textual description and the lyrics information. However, despite having excellent quality, the result turns out to be quite generic. After all, the generative model has learned the probability distribution quite well and can provide samples similar to those in the training set. If, instead of generating a track outright, we want to allow artists to use the model to interactively manipulate their creations, propose suggestions for the different instruments, and be intelligent assistants rather than generic composers, things get complicated. This is because it is difficult for a model to extract or manipulate the components (called *sources*) of a song when its training signal is, in most cases, an opaque *mixture* of sounds.

As such, the problem underlying any attempt at structural processing of a general audio track is the problem of separating the sources that compose it [6, 11]. No matter how skilled a model is at generating a track, if it is incapable of decomposing the underlying sources, we cannot have total control over what we are generating, and, as such, we cannot guide generation to act on the sub-components of the track. In this thesis, titled "From Source Separation to Compositional Music Generation," we start precisely from the source separation problem and aim to exploit it from a generative perspective to obtain models for compositional audio processing.

As we will explore in greater detail, the principal challenge in facilitating source separation arises primarily from the disparity between the limited data available in a decomposed format, which could serve as a learning resource, and the abundant data available in a mixed form. Our objective is to identify the most effective methods for utilizing this data and to comprehend how it can optimally inform a generative assistant for composition.

1.1 Motivation

The main motivation behind this research is applied in nature. We envision a computer system where the users can upload complex sound, especially music, and the system can seamlessly identify and decompose its constituent sources. At the same time, the system should be able to iteratively generate new material based on the decomposed tracks, potentially via rich textual descriptions provided as input. Ideally, the system should provide granular control over such generations, like envelopes or notes, for a human in the loop experience. Such a system has been described in the literature [12] as a generative digital workstation (GAW), an evolved form of a classic digital workstation (DAW). While there already exist commercial GAWs such as `WavTool`[1], they focus more on the symbolic representations, which must be synthesized aposteriori. For example, it is difficult to synthesize guitar from MIDI [13], especially when the dynamical content of the sound depends on information present in the other instruments (how should I play a guitar given a drum track?). Aditionally, the system can separate only the classes present in the MUSDB [14] dataset (Bass, Drums, Vocals and Other). While symbolic interfaces are useful for granular control [15], we would like to obtain general waveform separation and compositional generation capabilities.

The difficulties in creating such a system stems from the inherent challenges in decomposing complex audio signals into their constituent sources and the subsequent task of creatively generating music that is not only coherent but also compositionally rich.

The advent of deep learning [16] and generative models [17], however, has opened new avenues for addressing these challenges, providing the tools necessary to model the intricate structures and patterns inherent in music. This thesis aims to: *(i)* improve performance on general source separation, *(ii)* find new ways in which we can perform source separation and *(iii)* develop new ways in which we can perform waveform music generation at the stem level. We do so through a series of studies that apply, in the audio domain, techniques from the principal classes of generative models, such as adversarial training [18], latent discrete variable models [9, 19, 20] and diffusion models [21–26].

1.2 Contributions

The research presented herein is structured around five papers published by the author. We present such papers in the following list, together with the contributions of the author when the author has equally contributed to a paper or is second author.

- **Adversarial Permutation Invariant Training for Universal Sound Separation [27] (Conference Paper, ICASSP 2023):** The author of the thesis is first author of the paper.
- **Latent Autoregressive Source Separation [28] (Conference Paper, AAAI 2023):** The author of the thesis is first author and equal contributor to the paper, having proposed and implemented the idea of the discrete Bayesian sampler, proposed the idea of discrete likelihoods, implemented them in their first version and written the most of the paper.
- **Multi-Source Diffusion Models for Simultaneous Music [29] (Conference Paper, ICLR 2024, Oral):** The author of the thesis is equal contributor to the paper, having

[1] `https://wavtool.com/`

proposed the idea of a source-joint separator, formalized the proof of the Dirac separator, proposed the sub-FAD metric, performed the first experiments for source imputation and written substantial parts of the paper.

- **Zero-Shot Duet Singing Voices Separation with Diffusion Models [30] (Workshop Paper, ISMIR 2023, Music Demixing Workshop):** The author of the thesis is second author of the paper, having proposed the architecture and the sampler and having performed initial experiments.
- **Generalized Multi-Source Inference for Text Conditioned Music Diffusion Models [31] (Conference Paper, ICASSP 2024):** The author of the thesis is first author of the paper.

1.3 Structure of the Thesis

Following this introduction, the thesis is organized into chapters that correspond to the aforementioned studies, each chapter dedicated to a different aspect of the research:

- **Chapter 2**: Introduces a new approach to universal sound separation that leverages adversarial training to improve separation quality. The chapter is based on the paper [27].
- **Chapter 3**: Delves into a novel Bayesian inference methodology for performing source separation based on latent autoregressive models. The chapter is based on the paper [28].
- **Chapter 4**: Defines the Multi-Source Diffusion Model (MSDM) for simultaneous music generation and separation together with a novel separator based on Dirac delta functions. We also apply the independent variant of the model on the task of singing voice separation. The chapter is based on the papers [29] and [30].
- **Chapter 5**: Examines the generalized approach of MSDM via text-conditioned diffusion models. The chapter is based on the paper [31].
- **Chapter 6**: Concludes the thesis with a summary of the contributions and proposes interesting research directions, together with possible solutions.

In the body of the thesis, we have maintained an essential and compact style, which nevertheless should be fully comprehensible for researchers in the field of deep learning applied to audio with a background in generative modeling. Where definitions might seem dry, we nonetheless refer the reader to the relevant articles through the thesis's rich bibliography. Definitions are inserted when needed, and we make many references between the different chapters of the thesis, given the substantial coherence of the themes presented. While the chapters closely follow their respective papers, we have also included new material, for example, Section 3.1 and the final discussion, with a unifying character, in order to connect the different topics better and have a more global vision than reading the individual papers separately.

Chapter 2

Adversarial Permutation Invariant Training for Universal Sound Separation

Audio source separation is the task of decomposing an audio mixture $\mathbf{y}$ of length L, into its constituent sources, $\mathbf{x}_1,\ldots,\mathbf{x}_{N'}$, where N' is the total (often variable) number of sources. Formally, we can express the observed mixed signal[1] as:

$$\mathbf{y} = \sum_{n=1}^{N'} \mathbf{x}_n \,. \tag{2.1}$$

Each source $\mathbf{x}_n$ represents an individual audio stream within the mix, and the goal of source separation is to estimate each $\mathbf{x}_n$ from $\mathbf{y}$ as accurately as possible.

In the broadest setting, audio source separation does not restrict the type or nature of the sources $\mathbf{x}_n$. This setting is called *universal source separation*, and unlike domain-specific separation tasks such as music source separation (separating vocals, bass, and drums from a music mix [35–37]) or speech source separation (separating various speakers talking simultaneously [38–40]), it encompasses any kind of audio, not limiting the type or nature of the sources $\mathbf{x}_n$. This general approach is well-suited for real-world applications where the audio scene can contain diverse sounds, from human voices to natural and mechanical noises.

While universal source separation encompasses a wide range of scenarios, possibly including those typical of music source separation, it's important to note the distinct characteristics of the latter. Music source separation often deals with sources that exhibit a high degree of dependency, such as harmonically related instruments playing in sync. This level of inter-source dependency necessitates specialized approaches, as discussed in Chapters 4 and 5.

A common approach to universal sound separation with deep learning [16, 41] models consists in using Permutation Invariant Training (PIT) [42–45] or variants of it (like mixture invariant training [46]). In this setting, with predefined labels for the sources missing, the model's estimations of these sources may occur in any sequence. Consequently, it becomes essential to permute these estimations to correctly align them with their corresponding ground truth sources, ensuring the process remains agnostic to the order of output.

In this chapter we argue that an adversarial framework [18] for permutation invariant training

[1] When more than one mixture channel is available (e.g., 4 channels [32, 33] or 5.1 audio [34]) spatial cues can be leveraged to perform sound separation. This thesis focuses on mono signals and does not treat spatial methods.

Table 2.1: Summary of differences between the presented method (bottom) and previous adversarial PIT works for speech source separation (top).

Previous works on (two speaker) speech source separation	*Discriminator type and input (Type → Input: real / fake)*	*Multiple D?*	*Input to D (domain)*	*With $\mathcal{L}_{PIT}$? (+$\mathcal{L}_{PIT}$ domain)*	*Adversarial loss*
CBLDNN [48]	$D^{\text{STFT}}_{\text{ctx},I=0} \rightarrow [\mathbf{y}, \mathbf{x}_1, \mathbf{x}_2]$ / $[\mathbf{y}, \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2]$ ($\mathbf{y}$-conditioned)	No	|STFT|	Yes, |STFT|	LSGAN
SSGAN-PIT [49]: variant (i)	$D^{\text{STFT}}_{\text{ctx},I=0} \rightarrow [\mathbf{y}, \mathbf{x}_1, \mathbf{x}_2]$ / $[\mathbf{y}, \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2]$ ($\mathbf{y}$-conditioned)	No	|STFT|	Yes, |STFT|	LSGAN
SSGAN-PIT [49]: variant (ii)	$D^{\text{STFT}}_{\text{ctx},I=0} \rightarrow [\mathbf{x}_1, \mathbf{x}_2]$ / $[\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2]$	No	|STFT|	Yes, |STFT|	LSGAN
SSGAN-PIT [49]: variant (iii)	$D^{\text{STFT}}_{\text{inst}} \rightarrow [\mathbf{x}_1]$ / $[\hat{\mathbf{x}}_1]$ and $[\mathbf{x}_2]$ / $[\hat{\mathbf{x}}_2]$	No	|STFT|	Yes, |STFT|	LSGAN
Furcax [50]	$D^{\text{wave}}_{\text{ctx},I=0} \rightarrow [\mathbf{x}_1, \mathbf{x}_2]$ / $[\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2]$	No	Waveform	Yes, waveform	LSGAN
Conv-TasSAN [51]	Metric predictor $\rightarrow [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_2]$ / $[\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \mathbf{x}_1, \mathbf{x}_2]$	No	Waveform	Yes, waveform	MetricGAN
Presented source agnostic method for universal sound separation	Any above except "metric predictor"; with more than two input sources; using $D_{\text{ctx},I>0}$ with I-replacement; and with source agnostic discriminators	Yes, for better quality	Any above, plus masks	Optional	Hinge loss

can outperform regression-based [47] training in the universal source separation task. A number of speech source separation works also complemented PIT with adversarial losses [48–51]. Yet, as shown in Sections 2.4 and 2.5, adversarial PIT formulations used in speech separation do not perform well for universal source separation. To improve upon that, in Section 2.3 we extend speech source separation works with a novel I-replacement context-based adversarial loss, by combining multiple discriminators, and generalize adversarial PIT such that it works for universal sound separation (with source agnostic discriminators dealing with more than two sources). Table 2.1 outlines how this approach compares with speech separation works, which, for completeness are described in detail in Appendix A.1.

2.1 Methods for Universal Sound Separation

Various techniques have been previously used for universal sound separation: computational auditory scene analysis during the 1990's; factorization methods during the early 2000's; and, currently, deep learning techniques are being developed.

2.1.1 Computational Auditory Scene Analysis

Computational Auditory Scene Analysis (CASA) was inspired by the cognitive ability of humans to perceive individual sound sources in an audio mix [52–55]. CASA relies on our understanding of human perception and is based on following association cues related to the perceptual organization of sound sources [53]: *(i)* spectral proximity as closeness in time or frequency; *(ii)* harmonic concordance; *(iii)* synchronous changes expressed as common onsets, common offsets, common amplitude modulations, common frequency modulations, or equidirectional movement in the spectrum; and *(iv)* spatial proximity. A common approach to CASA consists in analyzing the spectrogram of a sound, where association cues are picked out, and representations of the selected cues are organized into an abstract description of the initial input to extract the sources [52, 55]. Accordingly, CASA methods are based on human perception and do not necessarily rely on source-specific cues. As a result, CASA methods allow building source agnostic models suitable for universal sound separation. The main drawback of CASA methods is the limited understanding of human perception.

2.1.2 Factorization Methods

Factorization methods aim at decomposing the mix $\mathbf{y}$ linearly into bases $\mathbf{W}$ and activations $\mathbf{x}$, as follows: $\mathbf{y} \approx \mathbf{W}\mathbf{x}$. Factorization methods can decompose the mix in an unsupervised fashion with-

out using prior information about the sources (i.e., blind source separation). Hence, factorization methods can be source-agnostic and suitable for universal sound separation. Common factorization methods are independent component analysis [56–59] and Non-negative Matrix Factorization (NMF) [60]. When more sources are available than mixture channels, as in our case, overcomplete methods are used [61, 62] (i.e., $\mathbf{W}$ is a fat matrix). Such a setting is more challenging than the more studied complete case (square $\mathbf{W}$) and requires further assumptions such as sparsity [63].

Factorization methods have been used for speech source separation [64], music source separation [65, 66], and to separate and detect (non-music and non-speech) sound events [67–70]. The main drawback of linear factorization methods is their limited representation power.

2.1.3 Deep Learning Regressors

Deep learning methods are currently considered the best-performing ones because they overcome the main drawbacks of the methods above. Deep learning is based on (deep) non-linear neural networks capable of learning from data, with minimal human supervision [16, 41].

In this setting, a mixture is fed to a neural network that outputs the separated sources. Training is typically performed in a supervised manner by matching the estimated separations with the ground truth sources with a regression loss (e.g., L^1 or L^2) [47]. Two approaches are prevalent: the mask-based approach and the waveform approach. In the mask-based approach, the model performs separation by applying estimated masks on mixtures, typically in the Short Time Fourier Transform (STFT) domain [71–77]. In the waveform approach, the model outputs the estimated sources directly in the time domain to overcome phase estimation, which is required when transforming the signal from the STFT domain to the waveform domain [2, 36, 78]. Further research explores combining the two representation in hybrid models [11].

Deep learning models are more expressive than (linear) factorization models, and are not constrained by our understanding of human perception (like CASA) since deep learning automatically learns from data. Provided that deep learning methods can be trained to fit any arbitrary function or dataset, these can be trained to be source agnostic. Hence, deep learning methods are suitable for universal sound separation [34, 42–45]. The main drawback of such methods is that they require large datasets to learn a model that generalizes well with real-world examples [38, 79], and deep learning models tend to be computationally demanding because large models are required to fit the complex datasets at hand [2].

2.2 Permutation Invariant Training

Source separation methods based on deep learning employ neural networks f_θ to predict N sources $\hat{\mathbf{x}} = f_\theta(\mathbf{y})$, given an observable mixture $\mathbf{y}$ (Eq. (2.1)). Permutation Invariant Training optimizes the learnable parameters θ of f_θ by minimizing the following permutation invariant loss:

$$\mathcal{L}_{\text{PIT}} = \min_{\mathbf{P}} \sum_{n=1}^{N} \mathcal{L}\left(\mathbf{x}_n, [\mathbf{P}\hat{\mathbf{x}}]_n\right), \tag{2.2}$$

where we consider all permutation matrices $\mathbf{P}$, $\mathbf{P}^*$ is the optimal permutation matrix minimizing Eq. (2.2), and $\mathcal{L}$ can be any regression loss [47]. Since f_θ outputs N sources, in case a mix contains $N' < N$ sources, we set the target $\mathbf{x}_n = 0$ for $n > N'$. Note that a permutation invariant loss

is required to build source agnostic models, because the outputs of f_θ can be any source and in any order. As such, the model must not focus on predicting one source type per output, and any possible permutation of output sources must be equally correct [42, 80]. A common loss $\mathcal{L}$ for universal sound separation is the τ-thresholded logarithmic mean squared error [42, 81], which is unbounded when $\mathbf{x}_n = 0$. In that case, since $\mathbf{y} \neq 0$, one can use a different $\mathcal{L}$ based on thresholding with respect to the mixture [81]:

$$\mathcal{L}(\mathbf{x}_n, \hat{\mathbf{x}}_n) = \begin{cases} 10 \log_{10} \left(\|\hat{\mathbf{x}}_n\|^2 + \tau \|\mathbf{y}\|^2 \right) & \text{if } \mathbf{x}_n = 0 \\ 10 \log_{10} \left(\|\mathbf{x}_n - \hat{\mathbf{x}}_n\|^2 + \tau \|\mathbf{x}_n\|^2 \right) & \text{otherwise.} \end{cases} \tag{2.3}$$

2.3 Adversarial Permutation Invariant Training

Generative Adversarial Networks (GAN) [18, 82] introduced the idea of adversarial training, enabling high-dimensional generative modeling for the first time. In the context of source separation, it consists of simultaneously training two models: f_θ producing plausible separations $\hat{\mathbf{x}}$, and one (or multiple) discriminator(s) D assessing if separations $\hat{\mathbf{x}}$ are produced by f_θ (fake) or are ground-truth separations $\mathbf{x}$ (real). Under this setup, the goal of f_θ is to estimate (fake) separations that are as close as possible to the (real) ones from the dataset, such that D misclassifies $\hat{\mathbf{x}}$ as $\mathbf{x}$ [18, 83]. We propose combining variations of an instance-based discriminator D_{inst} with a novel I-replacement context-based discriminator $D_{\text{ctx},I}$. Each D has a different role and is applicable to various domains: waveforms, magnitude STFTs, or masks. Without loss of generality, we present D_{inst} and $D_{\text{ctx},I}$ in the waveform domain and then show how to combine multiple discriminators operating at various domains to train f_θ.

2.3.1 Instance-Based Adversarial Loss

The role of D_{inst} is to provide adversarial cues on the realness of the separated sources without context. That is, D_{inst} assesses the realness of each source individually:

$$[\mathbf{x}_1] \;/\; [\hat{\mathbf{x}}_1] \;\; \ldots \;\; [\mathbf{x}_N] \;/\; [\hat{\mathbf{x}}_N].$$

Throughout the chapter, we use brackets [ ] to denote the D's input and left / right for real / fake separations (not division). Hence, individual real / fake separations (instances) are input to D_{inst}, which learns to classify them as real / fake (Fig. 2.1). D_{inst} is trained to maximize

$$\mathcal{L}_{\text{inst}} = \frac{1}{N} \sum_{n=1}^{N} (\mathcal{L}_{\text{inst}}^{\text{real},n} + \mathcal{L}_{\text{inst}}^{\text{fake},n}),$$

where $\mathcal{L}_{\text{inst}}^{\text{real},n}$ and $\mathcal{L}_{\text{inst}}^{\text{fake},n}$ correspond to the hinge loss [84]:

$$\begin{aligned} \mathcal{L}_{\text{inst}}^{\text{real},n} &= \min\left(0, -1 + D_{\text{inst}}(\mathbf{x}_n)\right), \\ \mathcal{L}_{\text{inst}}^{\text{fake},n} &= \min\left(0, -1 - D_{\text{inst}}(\hat{\mathbf{x}}_n)\right). \end{aligned}$$

We use the hinge loss since it has proven to be more stable during training in practice [85] with respect to the vanilla GAN loss [18] and the Wesserstein GAN (WGAN) loss [86], the latter requiring

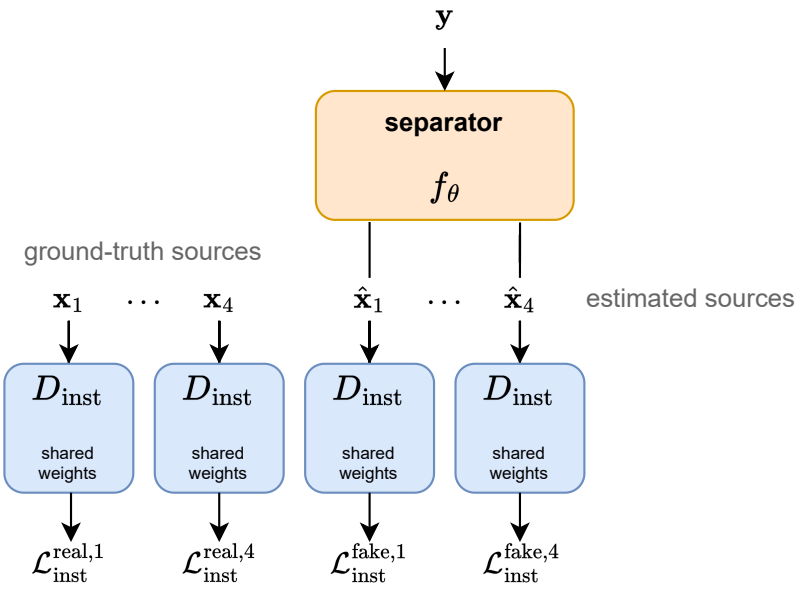

Figure 2.1: Instance-based adversarial loss ($N = 4$).

techniques such as spectral normalization [87] and weight clipping for stable training.

Previous works also explored using D_{inst}. However, such works used source-specific setups where each D_{inst} was specialized in a source type, e.g., for music source separation each D_{inst} was specialized in bass, drums, and vocals [35, 47], or for speech source separation D_{inst} was specialized in speech [49, 88]. Yet, each D_{inst} for universal sound separation is not specialized in any source type (are source agnostic) and assesses the realness of any audio, regardless of its source type.

2.3.2 I-Replacement Context-Based Adversarial Loss

The role of $D_{\text{ctx},I}$ is to provide adversarial cues on the realness of the separated sources considering all the sources in the mix (the context):

$$[\mathbf{x}_1, \dots, \mathbf{x}_N] \;/\; [\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_N].$$

Here, all the separations are input jointly to provide context to $D_{\text{ctx},I}$, which learns to classify them as real/fake. $D_{\text{ctx},I}$ can also be conditioned on the input mix $\mathbf{y}$:

$$[\mathbf{y}, \mathbf{x}_1, \dots, \mathbf{x}_N] \;/\; [\mathbf{y}, \bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_N].$$

Fake inputs contain entries $\bar{\mathbf{x}}_n$, obtained by randomly getting $I < N$ source indices without replacement $\Lambda_I \subset \{1, \dots, N\}$, and substituting the selected estimated sources $\hat{\mathbf{x}}_n^*$ with their ground-truth $\mathbf{x}_n$:

$$\bar{\mathbf{x}}_n = \begin{cases} \mathbf{x}_n & \text{if } n \in \Lambda_I, \\ \hat{\mathbf{x}}_n^* & \text{otherwise,} \end{cases} \tag{2.4}$$

where $\hat{\mathbf{x}}_n^* = [\mathbf{P}^* \hat{\mathbf{x}}]_n$ and $\mathbf{P}^*$ is the optimal permutation matrix minimizing Eq. (2.2) with $\mathcal{L}$ as in Eq. (2.3). Note that finding the right permutation $\mathbf{P}^*$ is important to replace the selected source $\hat{\mathbf{x}}_n^*$ with its corresponding ground-truth $\mathbf{x}_n$, because the (source agnostic) estimations do not necessarily

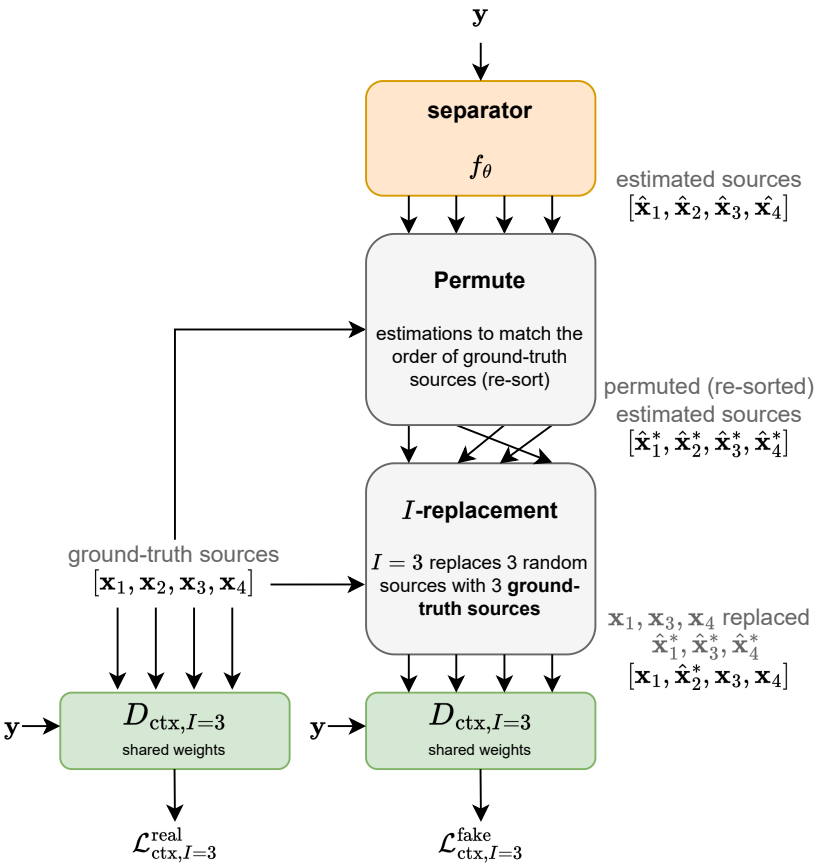

Figure 2.2: I-replacement context-based adversarial loss ($I = 3$, $N = 4$).

match the order of the ground-truth (see Fig. 2.2). Also, note that the $I = 0$ case corresponds to the standard context-based adversarial loss used for speech source separation (without I-replacement, see Table 2.1). Thus, we generalize adversarial PIT for universal sound separation by proposing a novel I-replacing schema that explicitly uses the ground-truth to guide the adversarial loss. $D_{\text{ctx},I}$ is trained to maximize

$$\mathcal{L}_{\text{ctx},I} = \mathcal{L}^{\text{real}}_{\text{ctx},I} + \mathcal{L}^{\text{fake}}_{\text{ctx},I},$$

where we again use the hinge loss [84]:

$$\mathcal{L}^{\text{real}}_{\text{ctx},I} = \min(0, -1 + D_{\text{ctx},I}(\mathbf{x}_1, \ldots, \mathbf{x}_N)),$$
$$\mathcal{L}^{\text{fake}}_{\text{ctx},I} = \min(0, -1 - D_{\text{ctx},I}(\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_N)).$$

2.3.3 Training with Multiple Discriminators

We have just presented D_{inst} and $D_{\text{ctx},I}$ in the waveform domain: $D^{\text{wave}}_{\text{inst}}$ and $D^{\text{wave}}_{\text{ctx},I}$. Next, we introduce them in the magnitude STFT ($D^{\text{STFT}}_{\text{inst}}, D^{\text{STFT}}_{\text{ctx},I}$) and mask ($D^{\text{mask}}_{\text{inst}}, D^{\text{mask}}_{\text{ctx},I}$) domains, and explain how to combine them. The STFT of a mix $\mathbf{y}$ is defined as $\mathbf{Y} = \text{STFT}(\mathbf{y})$. The magnitude STFT is then obtained by taking the absolute value of $\mathbf{Y}$ in the complex domain, namely $|\mathbf{Y}|$. Similarly, we denote as $|\mathbf{X}_n|$ and $|\hat{\mathbf{X}}_n|$ the magnitude STFT of the target and the estimated sources,

respectively. Ratio masks

$$\mathbf{R}_n = \frac{|\mathbf{X}_n|}{\sum_{n=1}^{N} |\mathbf{X}_n|}$$

are used to filter sources out from the mix by computing $\mathbf{X}_n = \mathbf{Y} \odot \mathbf{R}_n$, where $\odot$ denotes the element-wise product. Following the same notation as above, the input to the instance-based $D_{\text{inst}}^{\text{STFT}}$ and $D_{\text{inst}}^{\text{mask}}$ is

$$\begin{aligned} D_{\text{inst}}^{\text{STFT}}: &\quad [|\mathbf{X}_1|] \ / \ [|\hat{\mathbf{X}}_1|] \ \ldots \ [|\mathbf{X}_N|] \ / \ [|\hat{\mathbf{X}}_N|], \\ D_{\text{inst}}^{\text{mask}}: &\quad\quad [\mathbf{R}_1] \ / \ [\hat{\mathbf{R}}_1] \ \ldots \ [\mathbf{R}_N] \ / \ [\hat{\mathbf{R}}_N], \end{aligned}$$

and for the context-based $D_{\text{ctx},I}^{\text{STFT}}$ and $D_{\text{ctx},I}^{\text{mask}}$ (conditioned on $\mathbf{y}$) is

$$\begin{aligned} D_{\text{ctx},I}^{\text{STFT}}: &\quad [|\mathbf{Y}|, |\mathbf{X}_1|, \ldots, |\mathbf{X}_N|] \ / \ [|\mathbf{M}|, |\bar{\mathbf{X}}_1|, \ldots, |\bar{\mathbf{X}}_N|], \\ D_{\text{ctx},I}^{\text{mask}}: &\quad\quad [|\mathbf{Y}|, \mathbf{R}_1, \ldots, \mathbf{R}_N] \ / \ [|\mathbf{M}|, \bar{\mathbf{R}}_1, \ldots, \bar{\mathbf{R}}_N], \end{aligned}$$

where $|\bar{\mathbf{X}}_n|$ and $\bar{\mathbf{R}}_n$ entries follow the same I-replacement procedure as in Eq. (2.4). Here, for $D_{\text{ctx},I}^{\text{STFT}}$ and $D_{\text{ctx},I}^{\text{mask}}$, the optimal permutation matrix $\mathbf{P}^*$ required for re-sorting the fake examples is computed considering the L^1-loss between magnitude STFTs or masks. The motivation behind combining multiple discriminators is to facilitate a richer set of adversarial loss cues to train f_θ [47, 89, 90], such that both D_{inst} and $D_{\text{ctx},I}$ can provide different perspectives with respect to the same signal in various domains: waveform, magnitude STFT, and mask. Hence, in addition to train each D alone, one can train multiple combinations, e.g., $D_{\text{inst}}^{\text{wave}} + D_{\text{ctx},I}^{\text{wave}}$, $D_{\text{ctx},I}^{\text{wave}} + D_{\text{ctx},I}^{\text{STFT}} + D_{\text{ctx},I}^{\text{mask}}$, or any combination of the discriminators above. However, the more discriminators used, the more computationally expensive it is to run the loss, and the longer it takes to train f_θ (but does not affect inference time). To the best of our knowledge, training with multiple discriminators has never been considered for source separation before.

2.3.4 Separator Loss

In adversarial training, f_θ is trained such that its (fake) separations $\hat{\mathbf{x}}$ are misclassified by the discriminator(s) as ground-truth ones $\mathbf{x}$ (real). To do so, during every adversarial training step, we first update the discriminator(s) (without updating f_θ) based on $\mathcal{L}_{\text{inst}}$, $\mathcal{L}_{\text{ctx},I}$, or any combination of the losses above. Then, $\mathcal{L}_{\text{sep}}$ is minimized to train f_θ without updating the discriminator(s). For example, when using $D_{\text{inst}}^{\text{wave}}$ (with $D_{\text{inst}}^{\text{wave}}$ frozen) we minimize

$$\mathcal{L}_{\text{sep}} = -\frac{1}{N} \sum_{n=1}^{N} D_{\text{inst}}^{\text{wave}}(\hat{\mathbf{x}}_n),$$

when using $D_{\text{ctx},I}^{\text{STFT}}$ (with $D_{\text{ctx},I}^{\text{STFT}}$ frozen) we minimize

$$\mathcal{L}_{\text{sep}} = -D_{\text{ctx},I}^{\text{STFT}}(|\bar{\mathbf{X}}_1|, \ldots, |\bar{\mathbf{X}}_N|),$$

or when using $D_{\text{inst}}^{\text{STFT}}$ and $D_{\text{ctx},I}^{\text{STFT}}$ conditioned on $\mathbf{y}$ (with $D_{\text{inst}}^{\text{STFT}}$ and $D_{\text{ctx},I}^{\text{STFT}}$ frozen) we minimize

$$\mathcal{L}_{\text{sep}} = -\frac{1}{N} \sum_{n=1}^{N} D_{\text{inst}}^{\text{STFT}}(|\hat{\mathbf{X}}_n|) - D_{\text{ctx},I}^{\text{STFT}}(|\mathbf{Y}|, |\bar{\mathbf{X}}_1|, \ldots, |\bar{\mathbf{X}}_N|).$$

Again, note that we use the hinge loss [84]. While we are not presenting all possible loss combinations for brevity, from the above examples, one can infer all the combinations we experiment with in Section 2.5. Finally, we can also add an $\mathcal{L}_{\text{PIT}}$ term (as in Eqs. (2.2) and (2.3)) to adversarial PIT: $\mathcal{L}_{\text{sep}} + \lambda\mathcal{L}_{\text{PIT}}$, where λ scales $\mathcal{L}_{\text{PIT}}$ such that it is of the same magnitude as $\mathcal{L}_{\text{sep}}$ [85]. All previous adversarial PIT works for speech source separation used $\mathcal{L}_{\text{sep}} + \lambda\mathcal{L}_{\text{PIT}}$ (Table 2.1). Yet, in Section 2.5 we show that the proposed adversarial training setup allows dropping $\mathcal{L}_{\text{PIT}}$ while still obtaining competitive results, possibly because of the strong cues provided by $D_{\text{ctx},I}$ (with I-replacement) and the multiple discriminators. To the best of our knowledge, we are the first to report results similar to $\mathcal{L}_{\text{PIT}}$ with a purely adversarial setup (cf. [49]).

2.4 Experimental Setup

2.4.1 Dataset, Evaluation Metrics and Baseline

We use the reverberant Free Universal Sound Separation (FUSS) dataset, a common benchmark for universal sound separation with 20 k / 1 k / 1 k (train / val / test) mixes of 10 s with one to four sources [81, 91, 92]. Metrics rely on the Scale-Invariant SDR (Signal to Distortion Ratio) [81]:

$$\text{SI-SDR}(\mathbf{x}_n, \hat{\mathbf{x}}_n^*) = 10\log_{10}\frac{\|\alpha\mathbf{x}_n\|^2 + \epsilon}{\|\alpha\mathbf{x}_n - \hat{\mathbf{x}}_n^*\|^2 + \epsilon},$$

where $\alpha = \frac{\mathbf{x}_n{}^T\hat{\mathbf{x}}_n^* + \epsilon}{\|\mathbf{x}_n\|^2 + \epsilon}$, $\epsilon = 10^{-5}$, and $\hat{\mathbf{x}}_n^* = [\mathbf{P}^*\hat{\mathbf{x}}]_n$ with $\mathbf{P}^*$ being the optimal source-permutation matrix maximizing SI-SDR. Further, to account for inactive sources, estimate-target pairs that have silent target sources are discarded [46]. For mixes with one source, we compute $\text{SI-SDR}_\text{S} = \text{SI-SDR}(\mathbf{x}_n, \hat{\mathbf{x}}_n^*)$, which is equivalent to $\text{SI-SDR}(\mathbf{y}, \hat{\mathbf{x}}_n^*)$ since with one-source mixes the goal is to bypass the mix (the S sub-index stands for single-source[1]). For mixes with two to four sources, we report the average across sources of the $\text{SI-SDR}_\text{I} = \text{SI-SDR}(\mathbf{x}_n, \hat{\mathbf{x}}_n^*) - \text{SI-SDR}(\mathbf{x}_n, \mathbf{y})$ (the I sub-index stands for improvement[2]). Note that we are using the standard SI-SDR formulation as in [46] instead of the alternative (less common) SI-SDR in [44, 81], and that we use the reverberant FUSS dataset as in [81, 93] instead of its dry counterpart [44, 46] because it is more realistic. As such, the results in [44, 46, 81] are not comparable with ours because they either use a different SI-SDR formulation or a different dataset. To compare this work against a meaningful state-of-the-art baseline we use the DCASE model, an (improved) Time-Dilated Convolutional Network (TDCN++) [42, 78] predicting STFT masks [93]. It is trained on the reverberant FUSS dataset, and we evaluate it with the metrics based on the standard SI-SDR. Finally, we report SI-SDR_S for consistency [46, 81], but SI-SDR_I scores are more relevant for comparing models since most SI-SDR_S scores are already very close to the upper-bound of 39.9 dB (see Table 2.3).

2.4.2 Separator

The mix $\mathbf{y}$ of length $L = 160000$ (10 s at 16 kHz) is mapped to the STFT domain $\mathbf{Y}$, with windows of 32 ms and 25% overlap (256 frequency bins and 1250 frames). From $\mathbf{Y}$ we obtain its magnitude STFT $|\mathbf{Y}|$, that is input to a U-Net [94, 94] g_θ that predicts a ratio mask $\hat{\mathbf{R}}_n$. The mask is obtained using a softmax layer σ across the source dimension n: $\hat{\mathbf{R}} = \sigma(g_\theta(|\mathbf{Y}|))$, such that $\sum_n |\hat{\mathbf{X}}_n| = |\mathbf{Y}|$.

[2] SI-SDR_S is named as 1S or SS in [46, 81] andSI-SDR_I as MSi in [46, 81].

Then, we filter the estimated STFTs out of the mix with $\hat{\mathbf{X}}_n = \mathbf{Y} \odot \hat{\mathbf{R}}_n$, and use the inverse STFT to get the waveform estimates: $\hat{\mathbf{x}} = f_\theta(\mathbf{y}) = \text{iSTFT}(\hat{\mathbf{X}})$. Hence, our separator can be trained in the waveform domain (with $\mathcal{L}_{\text{PIT}}$, $D^{\text{wave}}_{\text{ctx},I}$, $D^{\text{wave}}_{\text{inst}}$), in the magnitude STFT domain (with $D^{\text{STFT}}_{\text{ctx},I}$, $D^{\text{STFT}}_{\text{inst}}$), and/or in the mask domain (with $D^{\text{mask}}_{\text{ctx},I}$, $D^{\text{mask}}_{\text{inst}}$). Our U-Net g_θ (of 46.9 M parameters) consists of an encoder, a bottleneck, and a decoder whose inputs include the corresponding encoder's block outputs. The encoder is built of 4 ResNet blocks [95], each followed by a downsampler that is a 1D-CNN [96] (kernel size=3, stride=2). The number of channels across encoder blocks is [256, 512, 512, 512]. The bottleneck consists of a ResNet block, self-attention, and another ResNet block (with all layers having 512 channels). The decoder block is built of 4 ResNet blocks, reversing the structure of the encoder, with upsamplers in place of downsamplers, reversing the number of channels of the encoder. The upsamplers perform linear interpolation followed by a 1D-CNN (kernel size=3, stride=1). A final linear layer adapts the output to predict the expected number of sources ($N = 4$).

2.4.3 Discriminators

Each D is of around 900 k parameters, are fully convolutional, and output one scalar. $D^{\text{wave}}_{\text{inst}}$ and $D^{\text{wave}}_{\text{ctx},I}$ rely on a similar model: 4 1D-CNN layers (kernel size=4, stride=3), interleaved by LeakyReLUs [97], with the following number of channels: [C, 128, 256, 256, 512], where $C = 1$ for $D^{\text{wave}}_{\text{inst}}$, and $C = 5$ or 4 for $D^{\text{wave}}_{\text{ctx},I}$, depending if it is $\mathbf{y}$-conditioned or not. Then, the 512 channels are projected to 1 with a 1D-CNN (kernel size=4, stride=1), and the final linear layer maps the remaining temporal dimension into a scalar. $D^{\text{STFT}}_{\text{inst}}$, $D^{\text{STFT}}_{\text{ctx},I}$, $D^{\text{mask}}_{\text{inst}}$, and $D^{\text{mask}}_{\text{ctx},I}$ are similar to $D^{\text{wave}}_{\text{inst}}$ and $D^{\text{wave}}_{\text{ctx},I}$, with the difference that 1D-CNNs are 2D (kernel size=4×4, stride=3×3) and the number of channels is [C, 64, 128, 128, 256].

2.4.4 Training and Evaluation Setup

Models are trained until convergence (around 500 k iterations) using the Adam optimizer [98], and the best model on the validation set is selected for evaluation. For training, we adjust the learning rate $\{10^{-5}, 10^{-4}, 10^{-3}\}$ and batch size $\{16, 32, 64, 96, 128\}$ such that all experiments, including ablations and baselines, get the best possible (validation) results. Finally, we use a mixture-consistency projection [99] at inference time (not during training) because it systematically improves SI-SDR$_\text{S}$ without degrading SI-SDR$_\text{I}$. The best model was trained for a month with 4 V100 GPUs with a learning rate of 10^{-4} and a batch size of 128.

2.5 Experiments and Discussion

First, in Table 2.2, we study various $D_{\text{ctx},I}$ configurations. We observe that the standard adversarial PIT ($I = 0$, as in speech source separation) consistently obtains the worst results for universal sound separation. In contrast, the models trained with I-replacement ($I > 0$) consistently obtain the best results. We hypothesize that with $I = 0$, f_θ does not separate much. Instead, it tends to approximate the naive solution of bypassing the mix. We can see this with the SI-SDR$_\text{S}$ scores, which tend to be closer (if not the same) to the SI-SDR$_\text{S}$ of the lower and upper bounds in Table 2.3. Overall, we note that the I-replacement context-based adversarial loss seems key to generalize ad-

Table 2.2: Study of various $D_{\text{ctx},I}$ configurations. $\mathbf{y}$ column: $D_{\text{ctx},I}$ is $\mathbf{y}$-conditioned or not. SI-SDR column: SI-SDR$_\text{I}$ / SI-SDR$_\text{S}$ in dB.

$\mathbf{y}$	$D^{\text{STFT}}_{\text{ctx},I}$	SI-SDR	$D^{\text{mask}}_{\text{ctx},I}$	SI-SDR	$D^{\text{wave}}_{\text{ctx},I}$	SI-SDR
Yes	I=0	4.9 / 35.7	I=0	5.9 / 35.8	I=0	8.5 / 39.9
Yes	I=1	11.0 / 33.2	I=1	9.3 / 33.6	I=1	11.3 / 35.5
Yes	I=2	10.7 / 33.1	I=2	9.6 / 34.3	I=2	11.9 / 36.3
Yes	I=3	11.7 / 33.4	I=3	8.6 / 32.6	I=3	12.1 / 35.9
No	I=0	5.7 / 35.0	I=0	3.8 / 36.8	I=0	8.5 / 39.9
No	I=1	9.9 / 32.8	I=1	6.7 / 34.1	I=1	10.3 / 37.6
No	I=2	10.4 / 32.0	I=2	9.0 / 33.6	I=2	11.0 / 36.0
No	I=3	10.2 / 31.0	I=3	9.7 / 33.5	I=3	11.4 / 35.3

Table 2.3: Comparison of adversarial PIT variants and baselines. SI-SDR column: SI-SDR$_\text{I}$ / SI-SDR$_\text{S}$ in dB. All $D_{\text{ctx},I}$ above are $\mathbf{y}$-conditioned with I=3, since it outperforms other setups (Table 2.2). All adversarial PIT ablations (rows 1-11 & Table 2.2) use the same f_θ.

$\mathcal{L}^{\text{STFT}}_{\text{ctx},I}$	$\mathcal{L}^{\text{wave}}_{\text{ctx},I}$	$\mathcal{L}^{\text{mask}}_{\text{ctx},I}$	$\mathcal{L}^{\text{STFT}}_{\text{inst}}$	$\mathcal{L}^{\text{wave}}_{\text{inst}}$	$\mathcal{L}^{\text{mask}}_{\text{inst}}$	$\mathcal{L}_{\text{PIT}}$	SI-SDR (↑)
✓	✓	✓	✓	✓	✓	✓	13.5 / 37.2
✓	✓	✓	✓	✓	✓	-	12.9 / 36.7
✓	✓	✓	-	-	-	-	12.5 / 37.3
✓	✓	-	✓	✓	-	✓	**13.8** / 35.3
✓	✓	-	✓	✓	-	-	12.0 / 38.3
✓	✓	-	-	-	-	-	11.6 / 35.3
✓	-	-	-	-	-	-	11.7 / 33.4
-	✓	-	-	-	-	-	12.1 / 35.9
-	-	✓	-	-	-	-	8.6 / 32.6
-	-	-	✓	-	-	-	8.2 / 27.0
-	-	-	-	✓	-	-	4.7 / 27.7
-	-	-	-	-	✓	-	4.5 / 36.8
-	-	-	-	-	-	✓	12.4 / 37.4
DCASE PIT baseline [93]							12.8 / 37.5
Lower-bound: return the input mix $\mathbf{y}$							0.0 / 39.9
Upper-bound: ideal ratio STFT masks [78]							25.3 / 39.9

versarial PIT for universal sound separation, where multiple heterogeneous sources are separated. This contrasts with adversarial PIT works for speech source separation, where two similar sources are separated. We argue that the universal sound separation case is more challenging, as speech separation discriminators can judge the realness of separations based on speech cues, but discriminators for universal sound separation cannot as sources can be of any kind. We hypothesize that the effectiveness of $D_{\text{ctx},I>0}$ can be attributed to:

- Replacing $\hat{\mathbf{x}}^*_n$ with $\mathbf{x}_n$ explicitly guides the adversarial loss to perform source separation. Note that $D_{\text{ctx},I=0}$ (and D_{inst}) focuses on assessing the realness of its input. Under this setup, a naive solution is always to bypass the mix, which looks as real as one-source mixes where the

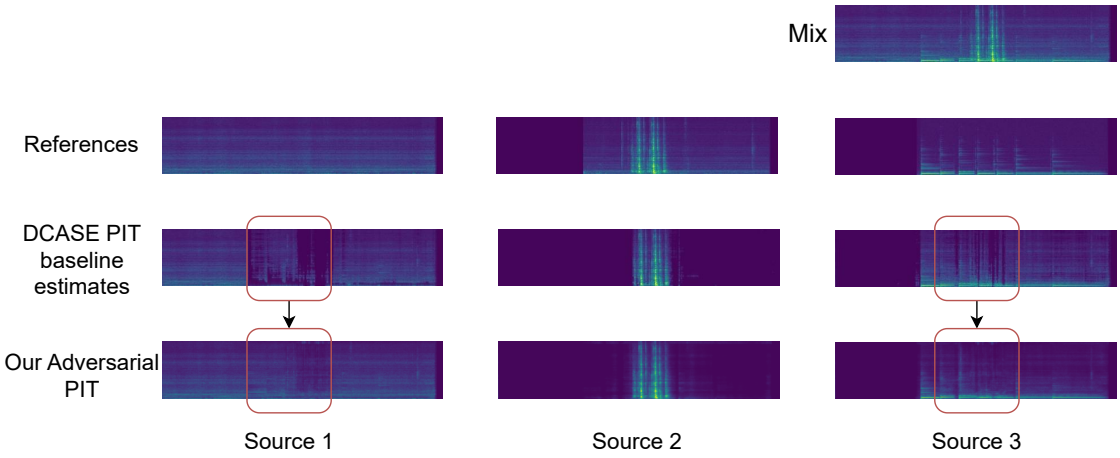

Figure 2.3: Qualitative results of separating the mixture at the top with our adversarial PIT baseline (using all discriminators, no regressor loss). We highlight the reduced spectral holes in the red squares using our method.

goal is to bypass the mix. To avoid this naive solution, some guidance like the I-replacement is required.

- It is more difficult for $D_{\text{ctx},I>0}$ to distinguish between real and fake separations, because fake ones contain replacements. Consequently, such replacements help defining a non-trivial task for the discriminator that results in a better adversarial loss to train f_θ.

Next, we study the discriminators introduced in Section 2.3 and their combination. In Table 2.2, we note that the $\mathbf{y}$-conditioned $D_{\text{ctx},I=3}$ generally outperform the rest. Hence, and for simplicity, in Table 2.3 we only experiment with this setup. We note the following trends:

- Our best result using adversarial PIT improves the state-of-the-art by 1 dB (from 12.8 to 13.8 dB) and improves the $\mathcal{L}_{\text{PIT}}$ baseline by 1.4 dB (from 12.4 to 13.8 dB). Informal listening also reveals that our best model separations more closely match the ground-truth sources and contain less spectral holes than the DCASE baseline (audio examples are available online[2]). Spectral holes are ubiquitous across mask-based source separation models, and are the unpleasant result of over-suppressing a source in a band where other sources are present. Adversarial training seems appropriate to tackle this issue since it improves the realness of the separations by avoiding spectral holes (which are not present in the training dataset). We showcase an example in Figure 2.3. We also compare our best model score (13.8 dB) against the lower and upper bounds (25.3 and 0 dB) to see that there is still room for improvement (also note this in our examples online[2] and follow the discussion in Appendix A.1).

- Our best results are obtained when combining multiple discriminators with $\mathcal{L}_{\text{PIT}}$ (over 13 dB). This shows that complementing the adversarial loss with $\mathcal{L}_{\text{PIT}}$ is beneficial, and confirms that using multiple discriminators in various domains can help to improve the separations' quality. We also note that $\mathcal{L}_{\text{PIT}}$ alone and the best adversarially trained models (without $\mathcal{L}_{\text{PIT}}$) obtain similar results (around 12.5 dB). Hence, purely adversarial models can obtain comparable results to $\mathcal{L}_{\text{PIT}}$ alone even without explicitly optimizing for $\mathcal{L}_{\text{PIT}}$, in Eq. (2.3), which is closely related to SI-SDR.

[2] `http://jordipons.me/apps/adversarialPIT/`

- When studying models trained with one D, we note that $D_{\text{ctx},I}$ alone tends to obtain better results than D_{inst} alone. We argue that the replacements in $D_{\text{ctx},I}$ explicitly guide the separator to perform source separation, while for D_{inst} this is not the case. In addition, $D^{\text{wave}}_{\text{ctx},I}$ alone obtains a competitive score of 12.1 dB, which can be improved up to 12.9 dB if combined with 5 additional discriminators. Hence, results can be improved by using multiple discriminators, but one can save computational resources by choosing the right D without dramatically compromising the results. Finally, even though D_{inst} alone under-performs the rest, we note that it can help improve the results when combined with $D_{\text{ctx},I}$.

2.6 Summary and Prospects

We adapted adversarial PIT for universal sound separation with a novel I-replacement context-based adversarial loss and by training with multiple discriminators. With that, we improve the separations by 1.4 dB SI-SDR$_\text{I}$ and reduce the unpleasant presence of spectral holes just by changing the loss without changing the model or dataset. Even with the improved results, the obtained separations can still be improved by an important margin.

Indeed, subsequent research [6] generalizing the presented work, leverages the enhanced architectures TDANet [100] (the original version with an increased number of parameters, which they call TDANet-Wav and a masked-based version operating on STFT magnitude spectrograms, similar to our separator, TDANet-STFT) and BSRNN [101], reaching 13.7 dB (TDANet-Wav) and 14.4 dB (TDANet-STFT, BSRNN) SI-SDR$_\text{I}$ on FUSS reverberant. Also, pre-trained versions of these architectures on an upstream large-scale dataset (15,499 hours, 3 orders of magnitude more data than FUSS) reach 16.4 dB (TDANet-STFT) and 16.0 dB (BSRNN) SI-SDR$_\text{I}$. Finally, by fine-tuning these pre-trained models on FUSS, they reach 18.6 dB (BSRNN) and 18.1 dB (TDANet-STFT) SI-SDR$_\text{I}$. We highlight that these improved results are obtained by training with a regressor loss alone. We hypothesize that the metrics could be scaled further by employing the adversarial method presented here with large-scale pre-training and improved architectures.

Chapter 3

Latent Autoregressive Source Separation

In the previous chapter, we introduced the general problem of source separation. Additionally, we used adversarial losses, characteristic of generative adversarial networks, remaining in a deterministic setting: given a mixture as input to the separator, the latter provides a single, best-fit explanation. The proposed method showcased superior performance compared to regression, keeping the neural architecture constant.

In this chapter, we fully embrace the generative setting [18, 23, 24, 102] in the context of source separation, where we can obtain multiple predictions given a mixture as input. As we will explore further, this paradigm shift proves beneficial for model reuse and allows us to make fewer assumptions about the data. At the same time, looking at the problem of source separation from a generative perspective, leads to a natural development of compositional music generation in the continuous domain (Chapters 4 and 5).

Autoregressive models have achieved impressive results in a plethora of domains ranging from natural language [4, 103] to densely-valued domains such as audio [8, 9, 104] and vision [85, 105, 106], including multimodal joint spaces [107–109]. In the dense setting, it is typical to train autoregressive models over discrete latent representations obtained through the quantization of continuous data, possibly using VQ-VAE autoencoders [19]. This way, generating higher-resolution samples while simultaneously reducing inference time is possible. The learned latent representations are also useful for downstream tasks [110]. However, in order to perform new non-trivial tasks, the standard practice is to fine-tune the model or, in the alternative, elicit prompting by scaling training [111, 112]. The former is usually the default option, but it requires additional optimization steps or modifications to the model. The latter is challenging on non-trivial tasks, especially in domains different from natural language [113, 114].

In this Chapter, we leverage existing vector-quantized autoregressive models without requiring any gradient-based optimization or architectural modifications to address the task of source separation for two source mixes.

We propose a generative approach to perform source separation via autoregressive prior distributions trained on a latent VQ-VAE domain (when class information is used, the approach is weakly supervised; otherwise, it is unsupervised). Performing separation in a latent domain has been explored in the context of discriminative models [115, 116], but here we explore it from a generative perspective, in order to re-use pre-trained components, as argued previously.

For our task, a non-parametric sparse likelihood function is learned by counting the occurrences

of latent mixed tokens with respect to the sources' tokens, obtained by mapping the data-domain sum signals and the relative addends via the VQ-VAE. This module is not invasive, neither for the VQ-VAE nor for the autoregressive priors, given that the representation space of the VQ-VAE does not change while learning the likelihood function. Finally, the likelihood function is combined with the estimations of the autoregressive priors at inference time via the Bayes formula, resulting in a posterior distribution. The separations are obtained from the posterior distributions via standard discrete samplers (e.g., ancestral, beam search). We call our method *Latent Autoregressive Source Separation (LASS)*.

Our contributions are summarized as follows:

- We introduce LASS as a Bayesian inference method for source separation that can leverage existing pre-trained autoregressive models in quantized latent domains.
- We experiment with LASS in the image domain and showcase competitive results at a significantly smaller cost in inference time with respect to competitors on MNIST and CelebA (32×32). We also showcase qualitative results on ImageNet (256×256) and CelebA-HQ (256×256), thanks to the scalability of LASS to pre-trained models. To the best of our knowledge, this is the first method to scale generative source separation to higher-resolution images.[1]
- We experiment with LASS in the music source separation task on the Slakh2100 dataset [119]. LASS obtains performance comparable to state-of-the-art supervised models, with a significantly smaller cost in inference and training time with respect to generative competitors.

3.1 Generative Source Separation

Before delving into the description of LASS, we want to outline a more abstract perspective on the possible generative source separation techniques, which will be useful in categorizing the method, along with the methods proposed in the following chapters.

Our treatment excludes (linear) unsupervised generative models that do not make assumptions on the sources via data (e.g., independent component analysis, see Subsection 2.1.2), given their limited representation power. Note that linearity is not the only limiting factor: recent research [120] proves that, more generally, disentanglement – in our case interpreted as source separation – is impossible without additional assumptions on the sources, even when working with more general deep (non-linear) models.

3.1.1 Conditional Generative Models

The most straightforward approach to developing a (deep) generative model for source separation involves configuring a neural separator (like the one described in Chapter 2) to rely on a random variable $\boldsymbol{\eta}$ (usually Gaussian or uniform in distribution). Since the model takes the mixture $\mathbf{y}$ as input, we talk about a *conditional generative model* (Figure 3.1, left), where we model the process of sampling from a conditional distribution $p(\mathbf{x} \mid \mathbf{y})$. The specific process in which we consume the

[1] Although not as prominent as its audio counterpart, image source separation has been addressed in literature [117, 118].

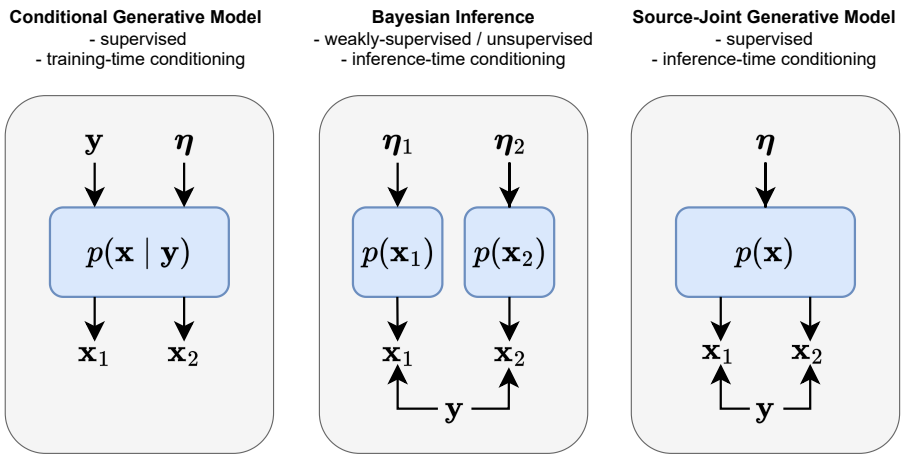

Figure 3.1: Families of generative techniques for source separation. We show the case where $\mathbf{y}$ is composed of two sources $\mathbf{x}_1$ and $\mathbf{x}_2$. *Left*: Conditional generative models. *Middle*: Bayesian inference. *Right*: Source-Joint generative models. In Bayesian inference and source-joint generative models, the mixture $\mathbf{y}$ is present only at inference time (it is not used as an input to the generative models).

variable $\boldsymbol{\eta}$ depends on the family of generative models we use. For example, a conditional GAN (cGAN) [121, 122] feds $\boldsymbol{\eta}$ to the generator together with the conditioning data ($\mathbf{y}$ in our case). Recently, several works have explored conditional generative modeling for speech source separation especially in the context of diffusion models [123–125].

The reader may already surmise that, at least in the case of the adversarial separator, obtaining a stochastic version is not difficult; it suffices to modify the input of the generator to accommodate a stochastic variable. The model, as in the previous situation relies on a supervised dataset and conditioning is performed during training. We can question what is the advantage of using a conditional generative separator over a regressor?

The main idea is that generative models can be pre-trained in an unsupervised fashion over large bodies of data [4, 7], learning representations that can be useful for downstream tasks such as source separation. The largest body of data related to audio signals is mixture data: [126] estimate that on YouTube alone there are more than 9 billion videos[2], each with an associated waveform, while supervised datasets containing clean sources are fairly scarce (or datasets containing single sources that can be combined independently), with [6], to the best of our knowledge, using the largest supervised audio source separation dataset in literature (15,499 hours; see Section 2.6). Modern fine-tuning techniques such as LoRA [127] or ControlNet [128] can turn a pre-trained generative model $p(\mathbf{x}')$ over generic waveform data $\mathbf{x}'$ (which can be both mixture or source data) into a conditional generative separator $p(\mathbf{x} \mid \mathbf{y})$, given an appropriate fine-tuning dataset. While we do not investigate this direction further in the body of the thesis, in Chapter 6 we see how such a direction is a convenient avenue for future research.

3.1.2 Bayesian Inference

A different technique that we can follow, and that we adopt in this chapter, is that of Bayesian inference (Figure 3.1, middle). In source separation, the peculiarity of Bayesian inference is the absence of mixtures $\mathbf{y}$ at training time, being the later present only at inference time. At training

[2]Here, of course, we skim over the relevant copyright issues that such a usage of data can entail and provide a maximum upper bound.

time we learn for each source $\mathbf{x}_n$, a generative model $p_n(\mathbf{x}_n)$, not conditioned on $\mathbf{y}$, that captures the probability distribution of the source alone. In this way, we only need information about individual sources as opposed to a set of contextual sources, so we do not have to rely on supervised datasets.

There are two types of training approaches. The first is an unsupervised approach, where we have no further information beyond the sources $\mathbf{x}_n$ (as in the case of universal source separation in the previous chapter). In such a case, we can use a single model $p(\mathbf{x}_n)$ to capture the distribution on the sources $\mathbf{x}_n$: by sampling from the generative model, we obtain a source, regardless of its type. The reader should not confuse this unsupervised approach with unsupervised blind source separation, i.e., in our case we have access to source data but we do not have class labels. A second approach is *weakly supervised*, where we have additional information $\mathbf{z}_n$ on the nature of the source $\mathbf{x}_n$, often in the form of class labels or text. The information $\mathbf{z}_n$ can condition the generative model explicitly as $p(\mathbf{x}_n \mid \mathbf{z}_n)$ or can be used implicitly, partitioning the data classes before training the model (i.e., train a model for bass $p_{\text{bass}}(\mathbf{x}_{\text{bass}})$ with bass sources, one for drums $p_{\text{drums}}(\mathbf{x}_{\text{drums}})$ with drums sources, etc.). The first approach is more convenient, given that we can train a single parametric model, and is made possible by the use of flexible conditioning mechanisms such as cross-attention [20], and will be employed in Chapter 5, while in this chapter we will use the class information implicitly.

At this point, however, we cannot do much if we do not know how to link the sources to the mixture $\mathbf{y}$ we want to separate at inference time. The missing ingredient is a *likelihood function,* or *observational model* $p(\mathbf{y} \mid \mathbf{x})$ which tell us if the sources $\mathbf{x}$ combine to the mixture. In a deterministic setting such distribution is the simple sum operation, but given that we work with probabilities, it can take different forms. In this and the following chapters we will model the likelihood function in various ways (categorical, Gaussian, Dirac delta), both to adapt to the choice of the generative model for the sources and to improve performance during inference.

Having all the required quantities, Bayesian inference estimates the sources using the well know Bayes formula. If $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ are two sources distributed according to the *prior distribution* $p_{\text{data}} = (p_{\text{data}_1}, p_{\text{data}_2})$, and $\mathbf{y} = (\mathbf{x}_1 + \mathbf{x}_2)/2$ is the observable mixture[3], we generate from the posterior:

$$p(\mathbf{x}_1, \mathbf{x}_2|\mathbf{y}) \propto p_{\text{data}}(\mathbf{x})p(\mathbf{y}|\mathbf{x}) \tag{3.1}$$

$$= p_{\text{data}_1}(\mathbf{x}_1)p_{\text{data}_2}(\mathbf{x}_2)p(\mathbf{y}|\mathbf{x}) \tag{3.2}$$

Bayesian inference, in source separation, takes an independence assumption on the sources, similar to Naive Bayes in classification [129, Section 8.2.2]: the joint distribution over the sources $p_{\text{data}}(\mathbf{x})$ factorizes in the product of the individual priors $p_{\text{data}}(\mathbf{x}_n)$. To be nitpicky, we should call the technique *independent Bayesian inference*, because one could also perform Bayesian inference without factorizing the prior (see the next section). Additionally, notice that the technique is automatically invariant to permutations, bypassing the need for permutation invariant training (Chapter 2). This renders the method applicable to the task of universal sound separation, when the prior is trained in an unsupervised manner.

Following early work on (deep) generative source separation using Bayesian inference based

[3]We write $\mathbf{y}$ as a sum with convex source gains 1/2 to be consistent with the material presented in the chapter (following the associated paper [28]); in following chapters we assume $\mathbf{y}$ is the direct sum, like in Eq. (2.1), where gains are embedded in the sources.

on GANs [40, 88, 130], Jayaram and Thickstun [118] propose the BASIS separation method in the image setting using score-based (diffusion) models [22] (BASIS-NCSN) and a noise-annealed version of flow-based models (BASIS-Glow). The inference procedure is performed in the image domain through Langevin dynamics [131], obtaining good quantitative and qualitative results. [132] explores Bayesian inference with normalizing flows. The authors in [133] extend the Langevin dynamics inference procedure to autoregressive models by re-training them with a noise schedule, introducing the Parallel and Flexible (PnF) method. Although innovative, mainly when used for tasks such as inpainting, this method cannot use pre-trained autoregressive models directly, requiring fine-tuning under different noise levels. Further, working directly on the data domain, it exhibits a high inference time and scales with difficulty to higher resolutions. In this chapter, we extend this line of research by proposing a separation procedure for latent autoregressive models that does not involve re-training, is scalable to arbitrary pre-trained checkpoints, and is compatible with standard discrete samplers.

We conclude the discussion around Bayesian inference by reasoning about its purpose. By not requiring supervised datasets, where often-case sources are inter-dependent (for example, in an environmental setting, it's more likely to find sounds of the same type in an office, or in music source separation datasets [119, 134], different tracks synchronize in time and harmonically) and are difficult to acquire, one might think that there are advantages in terms of the quantity of available data usable in this setting (after all, it's easier to obtain individual stems than to obtain sets of coherent stems). Yet a criticisms can be raised: One can randomly combine individual stems (as in the case of [6]) to obtain artificial mixtures and employ regressors, which can be trained easily and generally have much lower inference times (e.g., see Table 4.1). The main motivation for studying Bayesian techniques in a multi-source audio context, which will be partially justified in Chapter 5, is that pre-trained generative models on large amounts of data are able to parameterize individual sources in a few-shot manner, not having been trained to do so explicitly. Given this capability, only a likelihood model is needed to allow the use of such techniques at inference time, which can act as an additional regularizer for the problem. In this sense, Bayesian inference is a *guidance* technique (as it is called in recent literature, [135–137]) that further improves a base model (for example, see [138], where classifier guidance improves on a base conditional generative model).

3.1.3 Source-Joint Generative Models

Since independent Bayesian inference assumes independence between sources, it does not seem a satisfactory solution when sources are highly inter-dependent, especially in music source separation.

Typically, regressors [2] and conditional generative models for source separation like [123] estimate $K > 1$ number of sources from a mix, being required to capture inter-source dependencies during training. Also, recall from Table 2.3 the importance of the context in adversarial permutation invariant training. One question we can ask, is the following: given that sources have a high inter-dependency in music, maybe modeling the joint prior $p(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ in Eq. (3.1) can improve metrics in the setting of music source separation.

This third way, which uses a *source-joint generative model* (Figure 3.1, right), is a hybrid between the conditional generative model, given that it requires supervised coherent sources to be trained, and independent Bayesian inference, because separation is performed at inference-time. Chapter 4 is devoted at exploring this setting, in the context of diffusion models. Surprisingly,

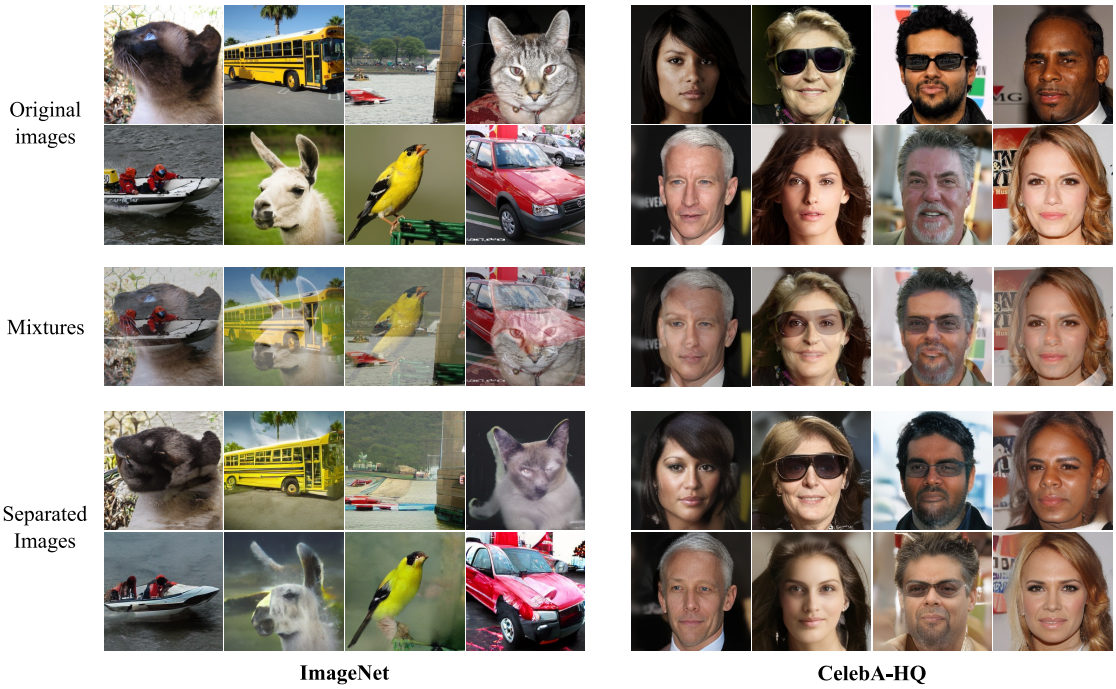

Figure 3.2: 256x256 separations obtained with LASS using pre-trained autoregressive models. Left: class-conditional ImageNet. Right: unconditional CelebA-HQ.

experimental evidence (Table 4.5) shows us that for source separation, modeling inter-dependecies does not improve over classic Bayesian inference in terms of separation quality! Nevertheless, such a model can be used both for separation and for novel music generation tasks, being a departure point for high-dimensional compositional music generation.

After this panoramic tour of generative source separation, we can return to the objective of this chapter, namely developing a Bayesian inference method for latent autoregressive separation.

3.2 Latent Autoregressive Models

This section briefly introduces the Vector-Quantized Variational Autoencoder (VQ-VAE) and autoregressive models, since they are core components of the separation procedure used in LASS.

3.2.1 VQ-VAE

A data point $\mathbf{x} \in \mathbb{R}^L$ (L is the total length of the data point, e.g., the length of the audio sequence or the number of pixel channels in an image) can be mapped to a discrete latent domain via a VQ-VAE [19]. First an encoder $E_\theta : \mathbb{R}^L \to \mathbb{R}^{S\times C}$ maps $\mathbf{x}$ to $E_\theta(\mathbf{x}) = (\mathbf{h}^1, \dots, \mathbf{h}^S)$, where C denotes the number of latent channels and S the length of the latent sequence. A bottleneck block $Q : \mathbb{R}^{S\times C} \to [K]^S$ casts the encoding into a discrete sequence $\boldsymbol{\xi} = (\xi^1, \dots, \xi^S)$ by mapping each $\mathbf{h}^s$ into the index (also called token) $\xi^s = B(\mathbf{h}^s)$ of the nearest neighbor $\mathbf{e}_{\xi^s}$ contained in an (ordered) set $\mathcal{C} = \{\mathbf{e}_k\}_{k=1}^K$ of learned vectors in $\mathbb{R}^C$ (called codes). A decoder $D_\psi : [K]^S \to \mathbb{R}^L$ maps the latent sequence back into the data domain, obtaining a reconstruction $\hat{\mathbf{x}} = D_\psi(\boldsymbol{\xi})$. VQ-GAN [85] is an enhanced version of the VQ-VAE, where the training loss is augmented with a discriminator and a perceptual loss, that improve reconstruction quality while increasing the compression rate of the

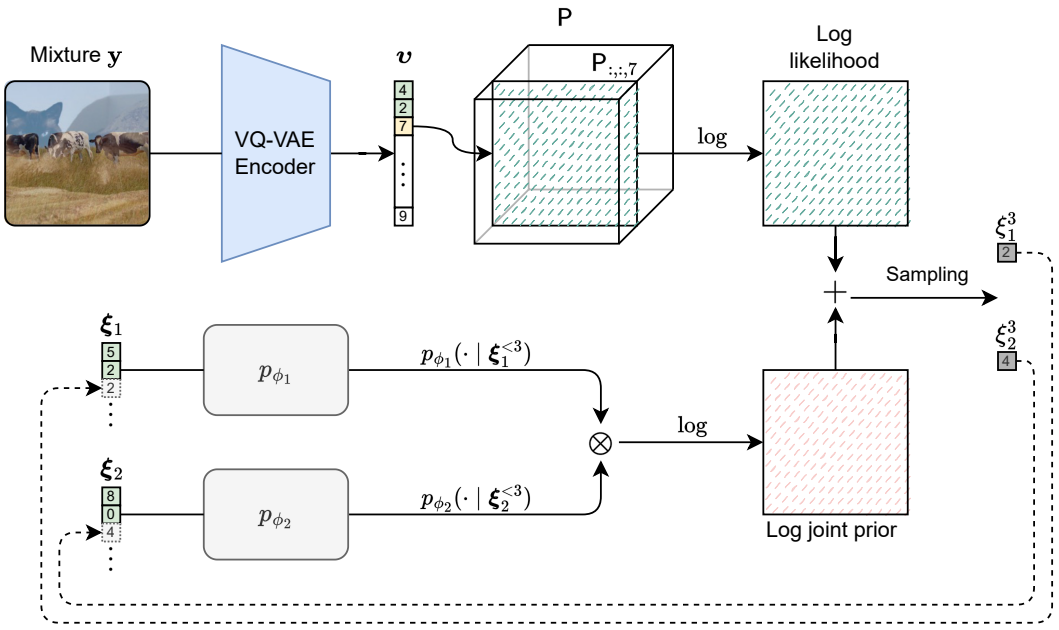

Figure 3.3: Schematic of the LASS separation procedure. The picture shows the separation procedure at $s = 3$ and is repeated until $s = S$. At the end of inference, we obtain $\mathbf{x}_1$ and $\mathbf{x}_2$ decoding $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$ via the VQ-VAE decoder (not depicted in the picture). We refer the reader to Algorithm 1 for more details.

autoencoder. We refer the reader to [19] and [85] for more details on VQ-VAE and VQ-GAN. In the remainder of the article, we will refer to both models as VQ-VAE and make distinctions when necessary.

3.2.2 Autoregressive Models

An autoregressive model learns a probability distribution over a discrete domain $[K]^S$ (in our case, the latent domain of the VQ-VAE). The joint probability of a sequence $\boldsymbol{\xi} = (\xi^1, \dots, \xi^S)$ is decomposed via the chain rule:

$$p_\phi(\boldsymbol{\xi}) = \prod_{s=1}^{S} p_\phi(\xi^s | \boldsymbol{\xi}^{<s}),$$

where $p_\phi(\cdot)$ is a learned parametric model, generally a neural network such as CNNs [139, 140] or transformers [20]. At inference time, samples can be obtained depending on the choice of a sampling procedure. Generally, ancestral sampling is used, where at each step, the token ξ^s is drawn stochastically from the conditional $p_\phi(\xi^s | \boldsymbol{\xi}^{<s})$, possibly employing top-k [141] filtering to increase the diversity of the generated data [142]. When the goal is instead to maximize the probability of the whole sequence (w.r.t. all the sequences), heuristics like beam search are used [143]. Beam search maintains B possible hypotheses (beams) $\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_B$ in parallel during inference. At each step s, it computes the conditional distributions $p_\phi(\xi_b^s | \boldsymbol{\xi}_b^{<s})$ for each beam b and selects the B new hypotheses that maximize the joint distributions $p_\phi(\boldsymbol{\xi}_b^{<s}) p_\phi(z^s | \boldsymbol{\xi}_b^{<s})$.

3.2.3 Autoregressive Models in the Audio Domain

Autoregressive models are firmly established in audio modeling [144]. The Jukebox model [104] leverages Scalable Transformers [20] to generate music tracks by utilizing hierarchical discrete representations. The hierarchy is composed of three code levels: top, middle and bottom, ranging from the most compressed (top) to the more detailed (bottom). The encoding at each level is learned via an independent VQ-VAE and the corresponding distribution is captured with a different autoregressive model. The top AR model is unconditional while the middle and bottom models are conditioned with codes at the "previous" level, e.g., middle is conditioned on top. Additionally, by incorporating a lyrics conditioner, Jukebox produces tracks with vocals that adhere to the specified text. Although Jukebox can model extended sequences in latent space, the resulting audio output exhibits quantization artifacts, especially at the top-most level. Recent latent autoregressive models for audio [9, 145, 146] are able to process longer contexts, their outputs are more cohesive and exhibit higher sound quality. They achieve improved sound quality by employing residual quantization [147, 148]. Residual quantization, differently from plain quantization, quantizes residuals $\mathbf{h}_r^s - \mathbf{e}_{\xi_r^s}$ in an iterative way using a set of codebooks $\{\mathcal{C}_r\}_{r=1,\dots,R}$, with $\mathbf{h}_1^s = E_\theta(\mathbf{x})^s$, $\mathbf{e}_{\xi_r^s} \in \mathcal{C}_r$ and R the total number of residuals. As such, if a plain VQ-VAE keeps only a single token for each step s, residual quantization keeps track of R increasingly finer codes. With residual quantization we can train a single autoregressive model (not requiring a cumbersome conditional hierarchy, as in Jukebox) and the decoded output does not suffer from quantization artifacts.

Cutting-edge latent autoregressive models in music, exemplified by MusicLM [8], facilitate generation by leveraging textual embeddings obtained via contrastive embedders [149, 150]. MusicLM can perform style transfer by taking a melody as input and modifying it using text prompting. In contrast, SingSong [151] pioneers vocal to accompaniment generation. The model we introduce in Chapter 4 also enables accompaniment creation but differentiates from the latter by enabling composition at the stem level, unlike the single accompaniment mixture produced by SingSong.

In this chapter we use Jukebox when working in the audio domain in Section 3.4.2, mainly because of its simplicity. A residual quantization version of the algorithm is left as future research.

3.3 Method

Working directly with Eq. (3.2) in the continuous data domain is inefficient. To overcome this problem, we first model p_{data} with autoregressive models in the latent space of a VQ-VAE. By changing the domain, we subsequentially redefine the likelihood function $p(\mathbf{y}|\mathbf{x}_1, \mathbf{x}_2)$ such that no gradient-based optimization or model re-training is required. We address the first issue in the following subsection and the second in the subsequent one. We then describe how to perform inference using LASS to separate data and propose a post-inference refinement procedure.

3.3.1 Latent Autoregressive Source Separation

This chapter explores the case in which p_{data} is estimated by a unique autoregressive model p_ϕ for all the sources (unsupervised) and the case in which we have two independent ones, $p_\phi = (p_{\phi_1}, p_{\phi_2})$, for each of the two sources (weakly supervised), either in terms of class-conditioned or independently trained models. We will focus on this latter case in the following, since the former can be generalized

setting $p_{\phi_1} = p_{\phi_2}$.

We denote the latent sources and mixtures, respectively, with $\boldsymbol{\xi} = (\boldsymbol{\xi}_1, \boldsymbol{\xi}_2) = Q(E_\theta(\boldsymbol{\xi}))$ and $\boldsymbol{\upsilon} = Q(E_\theta(\mathbf{y}))$. The posterior distribution in Eq. (3.2) can be locally expressed in the latent domain as:

$$p(\boldsymbol{\xi}^s|\boldsymbol{\xi}^{<s}, \boldsymbol{\upsilon}^{\leq s}) \propto p_\phi(\boldsymbol{\xi}^s|\boldsymbol{\xi}^{<s}) p(\boldsymbol{\upsilon}^{\leq s}|\boldsymbol{\xi}^{\leq s}), \tag{3.3}$$

for all $s = 1, \ldots, S$. The first factor is the (joint) Bayesian prior, modeled with independent autoregressive models. The second factor is the likelihood function, which quantifies the likelihood of the sequences $\boldsymbol{\xi}_1^{\leq s}, \boldsymbol{\xi}_2^{\leq s}$ to combine into $\boldsymbol{\upsilon}^{\leq s}$.

Since each code in the convolutional VQ-VAE describes a local portion of the data, and given that the mixing operation is point-wise in the data domain, the mixing relation between latent codes is local also in the latent domain. As such, we can drop the dependency on the previous context inside the likelihood function in Eq. (3.3), approximating it as:

$$p(\boldsymbol{\upsilon}^{\leq s}|\boldsymbol{\xi}^{\leq s}) \approx p(\upsilon^s|\boldsymbol{\xi}^s). \tag{3.4}$$

Notice that not depending on the global context and thus on the specific position in the sequence, we can drop the position index s:

$$p(\xi^s|\boldsymbol{\xi}^s) = p(\boldsymbol{\upsilon}^s|\xi_1^s, \xi_2^s) = p(\upsilon|\xi_1, \xi_2). \tag{3.5}$$

The following subsection describes how LASS models the likelihood function.

3.3.2 Discrete Likelihoods for Source Separation

Previous works [118, 133] model likelihood functions directly in the data domain, typically employing a σ-isotropic Gaussian term:

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|(\mathbf{x}_1 + \mathbf{x}_2)/2, \sigma^2\mathbf{I}).$$

In our setting, we cannot combine ξ_1^s and ξ_2^s (or the associate dense codes $\mathbf{e}_{\xi_1^s}$ and $\mathbf{e}_{\xi_2^s}$) with the canonical sum operation, given that the VQ-VAE does not impose an explicit arithmetic structure on the latent space.

To cope with this, we model the likelihood function in Eq. (3.5) using discrete conditionals, represented with rank-3 tensors[4] $\mathsf{P} \in \mathbb{R}^{K \times K \times K}$:

$$p(\cdot\,|\xi_1, \xi_2) = \mathsf{P}_{\xi_1, \xi_2, :}$$

In order to learn P, we perform frequency counts on latent mixed tokens given the latent sources' tokens, by iterating over a dataset X. We first initialize a null integer tensor $\mathsf{F}^0 \in \mathbb{N}^{K \times K \times K}$. Iterating over $\mathbf{x}_1, \mathbf{x}_2 \in X$, we compute $\mathbf{y} = (\mathbf{x}_1 + \mathbf{x}_2)/2$, then obtain the latent sequences $\boldsymbol{\xi}_1 = Q(E_\theta(\mathbf{x}_1)), \boldsymbol{\xi}_2 = Q(E_\theta(\mathbf{x}^2))$ and $\boldsymbol{\upsilon} = Q(E_\theta(\mathbf{y}))$. For each entry $(\xi_1^s, \xi_2^s, \upsilon^s) \in (\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \boldsymbol{\upsilon})$, at step i,

[4] We follow the notation for tensors as in Goodfellow et al. [41].

Algorithm 1 LASS inference

Input: $\mathbf{y}$
Output: $\mathbf{x}_1, \mathbf{x}_2$

$\boldsymbol{\upsilon} \leftarrow Q(E_\theta(\mathbf{y}))$
$\boldsymbol{\xi}_1 \leftarrow []$
$\boldsymbol{\xi}_2 \leftarrow []$
for $s = 1$ to S **do**
  prior $\leftarrow \log(p_{\phi_1}(\cdot\,|\boldsymbol{\xi}_1) \otimes p_{\phi_2}(\cdot\,|\boldsymbol{\xi}_2))$
  likelihood $\leftarrow \log(\mathsf{P}_{:,:,\upsilon^s})$
  posterior $\leftarrow$ prior $+ \lambda$ likelihood
  $(\xi_1^s, \xi_2^s) \leftarrow$ `Sampler`(posterior)
  $\boldsymbol{\xi}_1 \leftarrow$ `concat`$(\boldsymbol{\xi}_1, \xi_1^s)$
  $\boldsymbol{\xi}_2 \leftarrow$ `concat`$(\boldsymbol{\xi}_2, \xi_2^s)$
end for
$\mathbf{x}_1 \leftarrow D_\psi(\boldsymbol{\xi}_1)$
$\mathbf{x}_2 \leftarrow D_\psi(\boldsymbol{\xi}_2)$
return $\mathbf{x}_1$, $\mathbf{x}_2$

we simply increment the previous count by one:

$$\mathsf{F}^i_{\xi_1^s,\xi_2^s,\upsilon^s} = \mathsf{F}^{i-1}_{\xi_1^s,\xi_2^s,\upsilon^s} + 1\,,$$
$$\mathsf{F}^i_{\xi_2^s,\xi_1^s,\upsilon^s} = \mathsf{F}^{i-1}_{\xi_2^s,\xi_1^s,\upsilon^s} + 1\,.$$

We permute the order of the addends in order to enforce the commutative property of the sum. After performing the statistics, we can define P as:

$$\mathsf{P}_{\xi_1,\xi_2,:} = \frac{1}{\sum_{k=1}^{K} \mathsf{F}_{\xi_1,\xi_2,\upsilon=k}} \mathsf{F}_{\xi_1,\xi_2,:}. \tag{3.6}$$

At inference time, the likelihood function (parametric in ξ_1 and ξ_2, with υ fixed) can be obtained by slicing the tensor along υ, namely:

$$p(\upsilon|\cdot,\cdot) = \mathsf{P}_{:,:,\upsilon}.$$

At first glance, modeling the conditional distributions without parameters could seem memory inefficient, with a complexity of $O(K^3)$. In practice, the tensor P is *highly sparse*. We showcase this in Table 3.1 for all our experiments, where the density of P is defined as the percentage of nonzero elements in P.

Employing discrete likelihood functions for source separation in the latent domain of a VQ-VAE is a flexible approach; there is no need to change the VQ-VAE representation, the non-parametric learning procedure does not depend on hyperparameters, and the autoregressive priors do not require re-training.

3.3.3 Inference Procedure

Given an observable mixture $\mathbf{y}$, the autoregressive priors p_{ϕ_1}, p_{ϕ_2} and the learned likelihood tensor P, it is possible to perform inference and estimate $\mathbf{x}_1, \mathbf{x}_2$, as described in Algorithm 1 and depicted

in Figure 3.3.

We start by mapping $\mathbf{y}$ to the latent domain obtaining $\boldsymbol{\upsilon} = Q(E_\theta(\mathbf{y}))$ and initializing the estimates $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2$ with the empty sequences. The algorithm iterates over $s = 1, \ldots, S$.

At each step, the joint prior (a $K \times K$ matrix) is computed (Line 5) by taking the outer product of the two distributions predicted by the autoregressive models conditioned over the past context. We use the logarithms of the distributions for numerical stability. The log-likelihood function is computed next (Line 6), applying the logarithm on $\mathsf{P}_{:,:,\boldsymbol{\upsilon}^s}$. In our experiments, we can apply different scaling factors λ to the log-likelihood to balance it to the priors. The two matrices are then combined to form the posterior on Line 7.

Finally (Lines 8-10), different techniques can be employed to sample the best candidate tokens (ξ_1^s, ξ_2^s) from the posterior. In our experiments, we used ancestral sampling (with and without top-k filtering) and beam search. After the inference loop ends, the estimated sequences are mapped back to the data domain with the decoder of the VQ-VAE (Lines 12-13), obtaining $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$.

Post-inference Refinement

The quality of the separated images is limited by the quality of the images obtained via the VQ-VAE decoder. To enhance the separations we can adopt an additional refinement step by iteratively optimizing the VQ-VAE latent representations of the samples:

$$\mathbf{e}_1(i+1) = \mathbf{e}_1(i) + \alpha \nabla_{\mathbf{e}_1(i)} \| D_\psi(\mathbf{e}_1(i)) + D_\psi(\mathbf{e}_2(i)) - 2\mathbf{y} \|_2 \tag{3.7}$$

$$\mathbf{e}_2(i+1) = \mathbf{e}_2(i) + \alpha \nabla_{\mathbf{e}_2(i)} \| D_\psi(\mathbf{e}_1(i)) + D_\psi(\mathbf{e}_2(i)) - 2\mathbf{y} \|_2 \tag{3.8}$$

for $i = 1, \ldots, I$ and $\mathbf{e}_1(1) = E_\theta(\mathbf{x}_1)$, $\mathbf{e}_2(1) = E_\theta(\mathbf{x}_2)$. In simple words, we optimize for dense latent embeddings such that their decodings better sum to the mixture, initializing them to the output of Algorithm 1. We found this strategy particularly helpful on the MNIST datset, where we assess the quality of the separation through a pixel-wise metric (PSNR) and the VQ-VAE tends to produce smooth images.

3.4 Experiments

We perform quantitative and qualitative experiments on various datasets to demonstrate the efficacy and scalability of LASS. In the image domain, we evaluate on MNIST [152] and CelebA (32×32) [153] and present qualitative results on the higher resolution datasets CelebA-HQ (256×256) [154] and ImageNet (256×256) [155]. In the audio domain, we test on Slakh2100 [119], a large dataset for music source separation suitable for generative modeling. We conducted all our experiments on a single Nvidia RTX 3090 GPU with 24 GB of VRAM. Implementation details for all the models are provided on the companion website[5].

3.4.1 Image Source Separation

We choose the transformer architecture [20] as the autoregressive backbone for all image source separation experiments. With MNIST and CelebA, we first train a VQ-VAE, then train the au-

[5] github.com/gladia-research-group/latent-autoregressive-source-separation

Table 3.1: Statistics on likelihood functions over different datasets. K is the number of VQ-VAE (or VQ-GAN) latent codes. Density is the percentage of nonzero elements in the likelihood function.

Dataset	K	Density (%)
MNIST	256	1.49×10^{0}
CelebA	512	6.06×10^{0}
CelebA-HQ	1024	3.80×10^{-1}
ImageNet	16384	3.90×10^{-3}
Slakh2100 (Drum + Bass)	2048	7.60×10^{-2}

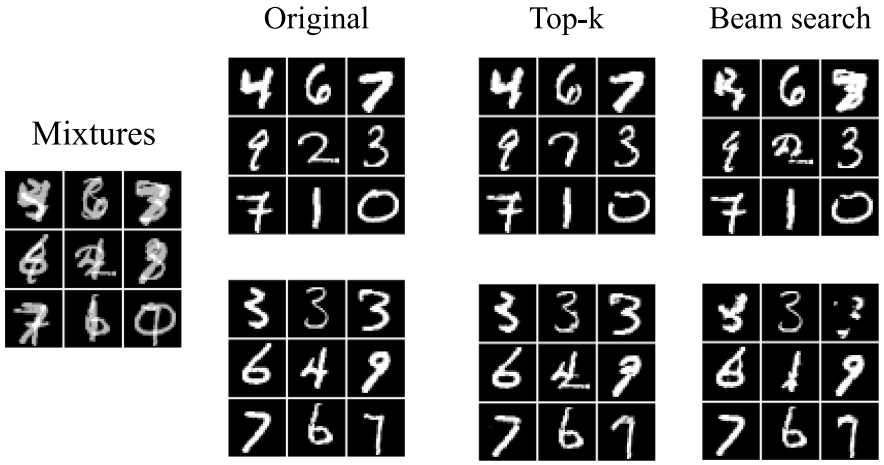

Figure 3.4: Results on MNIST with top-k sampling ($k = 32$) over a random batch of examples. Top-k sampling produces more defined digits, in agreement with the results in Table 3.3.

toregressive transformer on its latent space. We use $K = 256$ codes on MNIST and $K = 512$ on CelebA, given that CelebA presents more variability, requiring more information to reconstruct data. On CelebA-HQ and ImageNet, we leverage pre-trained VQ-GANs [85] alongside the pre-trained tansformers published by the authors[6] (`celebahq_transformer` checkpoint for CelebA-HQ and `cin_transformer` for ImageNet). Given the flexibility of LASS, they are employed inside the separation algorithm without modifications. On CelebA-HQ the VQ-GAN has $K = 1024$ codes, while on ImageNet has $K = 16384$. As a first step, in all image-based experiments we learn the P tensor using the procedure presented in the section "Method". As shown in Table 3.1, CelebA presents the lowest sparsity (highest density) while ImageNet has the highest. In all cases, density is below 7%, and the inference procedure is not affected by memory issues.

Quantitative Results

To assess the quality of image separations produced by LASS, we compare our method with different baselines on MNIST and CelebA.

On MNIST, we compare LASS with results reported for the two generative separation methods "BASIS NCSN" (score-based) and "BASIS Glow" (noise-annealed flow-based) from [118], the GAN-based "S-D" method [130], the fully supervised version of Neural Egg "NES" and the "Average" baseline, where separations are obtained directly from the mixture $\mathbf{x}_1 = \mathbf{x}_2 = \mathbf{y}/2$. In all these cases, the evaluation metric is the Peak Signal to Noise Ratio (PSNR) [156]. We follow the exper-

[6] `github.com/CompVis/taming-transformers`

Table 3.2: Comparison with other methods on MNIST and CelebA test set. Results are reported in PSNR (higher is better) and FID (lower is better).

Separation Method	MNIST (PSNR)	CelebA (FID)
Average	14.9	15.19
Non-negative Matrix Factorization (NMF)	9.4	-
S-D	18.5	-
BASIS Glow	22.7	-
BASIS NCSN	29.3	7.55
LASS (Ours)	24.2	8.96

imental procedure of [118] on MNIST and perform separation on a set of 6,000 mixtures obtained by combining 12,000 test sources. In order to choose the best sampler for this dataset, we validate the set of samplers in Table 3.3 on 1,000 mixtures constructed from the test split. We find that stochastic samplers perform best (PSNR $>$ 20 dB) while MAP methods do not reach a satisfactory performance. We hypothesize that beam search tends to fall into sub-optimal solutions by performing incorrect choices in early inference over sparse images such as MNIST digits. Top-k sampling with $k = 32$ performs best, so we choose it to perform the evaluation (a qualitative comparison is shown in Figure 3.4). For each mixture in the test set we sample a candidate batch of 512 separations, select the separation whose sum better matches the mixture (w.r.t. the L^2 distance), and finally perform the refinement procedure in Eqs. (3.7), (3.8) with $I = 500$ and $\alpha = 0.1$. Evaluation metrics on this experiment are shown in Table 3.2, while inference time is reported in Table 3.4. Our method achieves higher metrics than "NMF", "S-D" and "BASIS Glow" and is faster than "BASIS NCSN", thanks to the latent quantization. The higher PSNR achieved by the later method can be attributed to the fact that, in their case, the underlying generative models perform sampling directly in the image domain; in our case, the VQ-VAE compression can hinder the metrics.

We compare our method to "BASIS NCSN", using the pre-trained NCSN model [22] on CelebA. In this case, we evaluate against the FID metric [157] instead of PSNR, given that for datasets that feature more variability than MNIST, source separation can be an underdetermined task [118]: semantically good separations can receive a low PSNR score since the generative models may alter features such as color and cues (an effect amplified by a GAN decoder). The FID metric better quantifies if the separations belong to the distribution of the sources. We test on 10,000 mixtures computed from pair of images in the validation split using a top-k sampler with $k = 32$. We scale the likelihood term by multiplying it by $\lambda = 3$. It is known in the literature that score-based models outperform autoregressive models on FID metrics [158] on different datasets, yet our method paired with an autoregressive model shows competitive results with respect to "BASIS NCSN".

Qualitative Results

To demonstrate the flexibility of LASS in using existing models without any modification, we leverage pre-trained checkpoints on CelebA-HQ and ImageNet. In this case, only the likelihood tensor P is learned. We showcase a curated results list in Figure 3.2 and a more extensive list on the companion website. To the best of our knowledge, our method is the first to scale up to 256×256 resolutions and can be used with more powerful latent autoregressive models without re-training

Table 3.3: Performance of LASS with different sampling methods. On MNIST, the reported score is PSNR (dB) (higher is better), while on Slakh2100 is SDR (dB) (higher is better). When stochastic samplers are used (ancestral or top-k), the selected solution in the batch is the one whose sum minimizes the L^2 distance to the input mixture.

Sampling Method	MNIST (PSNR)	Slakh2100 (SDR)
Greedy	17.36 ± 5.90	1.23 ± 2.33
Beam Search	16.96 ± 5.78	5.01 ± 2.39
Ancestral Sampl.	24.03 ± 6.37	4.23 ± 2.29
Top-k ($k = 16$)	23.74 ± 6.55	3.13 ± 2.53
Top-k ($k = 32$)	24.23 ± 6.23	2.93 ± 2.20
Top-k ($k = 64$)	23.85 ± 6.13	3.24 ± 3.29

Table 3.4: Inference speed comparisons for computing one separation. To estimate variance, we repeat inference 10 times on MNIST and 3 times on Slakh2100. We consider 3-second-long mixtures on Slakh2100.

	Method	Time
MNIST	**LASS (Ours)**	4.49 s ± 0.27 s
	BASIS NCSN	53.34 s ± 0.51 s
Slakh2100	**LASS (Ours)**	1.33 min ± 0.87 s
	PnF	42.29 min ± 1.08 s

(which is cumbersome for very large models). As such, end-users can perform generative separation without having access to extensive computational resources for training these large models.

3.4.2 Music Source Separation

We perform experiments on the Slakh2100 dataset [119] for the music source separation task. This dataset contains 2100 songs with separated sources belonging to 34 instrument categories, for a total of 145 hours of mixtures. We focus on the "Drums" and "Bass" data classes, with tracks sampled at 22kHz. We use the public checkpoint of Dhariwal et al. [104] for the VQ-VAE model, taking advantage of its expressivity in modeling audio data over a quantized domain. Given that such a model is trained at 44kHz, we upsample input data linearly, then downsample the output back at 22kHz. For the two autoregressive priors, we train two transformer models, one for "Drums" and another for "Bass" and learn the likelihood function over the VQ-VAE (statistics are reported in Table 3.1). We compare LASS to a set of unsupervised blind source separation methods -"rPCA" [72], "ICA" [159], "HPSS" [160], "FT2D" [161] - and to two supervised baselines Demucs [2] and Conv-Tasnet [78] using the SDR (dB) evaluation metric computed with the `museval` library [162]. To evaluate the methods, we select 900 music chunks of 3 seconds from the test splits of the "Drums" and "Bass" classes, combining them to form 450 mixtures. The validation dataset is constructed similarly (with different music chunks). As a sampling strategy, we use beam search since it shows the best results on a validation of 50 mixtures (Table 3.3), using $B = 100$ beams. Evaluation results are reported in Table 3.5: LASS clearly performs better than all the blind unsupervised baselines and is comparable with the results obtained by methods that use supervision. Furthermore, we compare the time performance of LASS against the generative source separation method "PnF"

Table 3.5: Comparison with other source separation methods on Slakh2100 ("Drums" and "Bass" classes). Results are reported in SDR (dB) (higher is better). Lower part of the table shows supervised methods. With "Avg" we refer to the mean between the results over the two classes.

Separation Method	Avg	Drums	Bass
rPCA	0.82	0.60	1.05
ICA	-1.26	-0.99	-1.53
HPSS	-0.45	-0.56	-0.33
REPET	1.04	0.53	1.54
FT2D	0.95	0.59	1.31
LASS (Ours)	4.86	4.73	4.98
Demucs	5.39	5.42	5.36
Conv-Tasnet	5.47	5.51	5.43

[133] by evaluating the time required to separate a mixture of 3 seconds sampled at 22 kHz (piano vs. voice on "PnF"). Results in Table 3.4 show that LASS is significantly faster, and as such, it can be adopted in more realistic inference scenarios.

3.5 Limitations

In this chapter we limit our analysis to the separation of two sources. Even if this is a common setup especially in image separation [117, 133], dealing with multiple sources is a possible line of future work. Under our framework, this would require to increase the dimensions of the discrete distributions (both the priors and the likelihood function). To alleviate this problem, techniques such as recursive separation may be employed [163].

Another limitation of the proposed method is the locality assumption taken in Eq.(3.4). Different tasks such as colorization and super-resolution would require a larger conditioning context, and newer quantization schemes to aggregate latent codes on global contexts (using self-attention in the encoder and the decoder of the VQ-VAE) [164]. Adopting a VQ-VAE quantized with respect to the latent channels [165] combined with a parametric likelihood function could be a way to solve this limitation, while still maintaining the flexible separation between VQ-VAE, priors, and likelihoods presented in the chapter.

Lastly, state-of-the art autoregressive models [9, 145, 146] work with residual quantization [147], where differences between embedding vectors and codebook elements are quantized progressively, leading to higher perceptual quality. A residual version of the presented algorithm is to be sought.

3.6 Summary and Prospects

In this chapter, we proposed LASS as a source separation method for latent autoregressive models that does not modify the structure of the priors. We have tested our method on different datasets and have shown results comparable to state-of-the-art methods while being more scalable and faster at inference time. Additionally, we have shown qualitative results at a higher resolution than those proposed by the competitors. We believe our method will benefit from the improved quality of newer

autoregressive models [9, 145, 146], improving both the quantitative metrics and the perceptive results.

Chapter 4

Multi-Source Diffusion Models for Simultaneous Music Generation and Separation

In the previous chapter, we explored how to perform source separation using independent Bayesian inference with latent autoregressive models. We also saw that the method can be generally applied to any audio domain, similarly to how we proceeded in Chapter 2, although we tested the latter in the musical domain. In this chapter, we focus specifically on the musical domain, investigating its intrinsic peculiarities.

In contrast to other areas within the audio domain, such as speech, the sources found in musical compositions (stems) are intrinsically linked, sharing a unified *context* due to their strong interdependence. For example, a guitar solo in a song plays in sync with the drums and follows the accompaniment of the bass line. As argued in Section 3.1.3, the joint distribution of musical sources $p(\mathbf{x}_1, \dots, \mathbf{x}_N)$ does *not* factorizes into the product of individual source distributions $\{p_n(\mathbf{x}_n)\}_{n=1,\dots,N}$, which is the assumption of independence in Bayesian inference (Section 3.1.2). Also notice that we can obtain a sample from the distribution of the mixture $p(\mathbf{y})$ if we have knowledge of a sample from the joint $p(\mathbf{x}_1, \dots, \mathbf{x}_N)$, by simply applying the sum operation. Obtaining the inverse, namely a sample from the joint $p(\mathbf{x}_1, \dots, \mathbf{x}_N)$ given $p(\mathbf{y})$ is difficult, being equivalent to source-separation.

Humans possess the remarkable ability to simultaneously manage multiple sound sources, both in synthesis (for instance, in creating music or generating sounds) and in analysis (such as in separating distinct sound sources). Specifically, composers have the skill to craft several sources, by first composing the musical structure then playing, singing or synthesizing the sounds $\mathbf{x}_1, \dots, \mathbf{x}_N$ that blend into a coherent mix $\mathbf{y}$ and can discern and extract details about each individual source $\mathbf{x}_1, \dots, \mathbf{x}_N$ trough listening the combined sound $\mathbf{y}$. The capacity to both construct and deconstruct sound plays a pivotal role in generative music modeling. A model that aims to facilitate music composition needs to have the ability to identify and separate individual sources within a mix, enabling targeted manipulations on each isolated source. This functionality grants composers the utmost flexibility to determine what elements to alter and what to preserve within a piece. Thus, we posit that the endeavor of generating compositional music in the continuous domain (i.e., as opposed to generating music in symbolic domains like MIDI) is deeply intertwined with the process

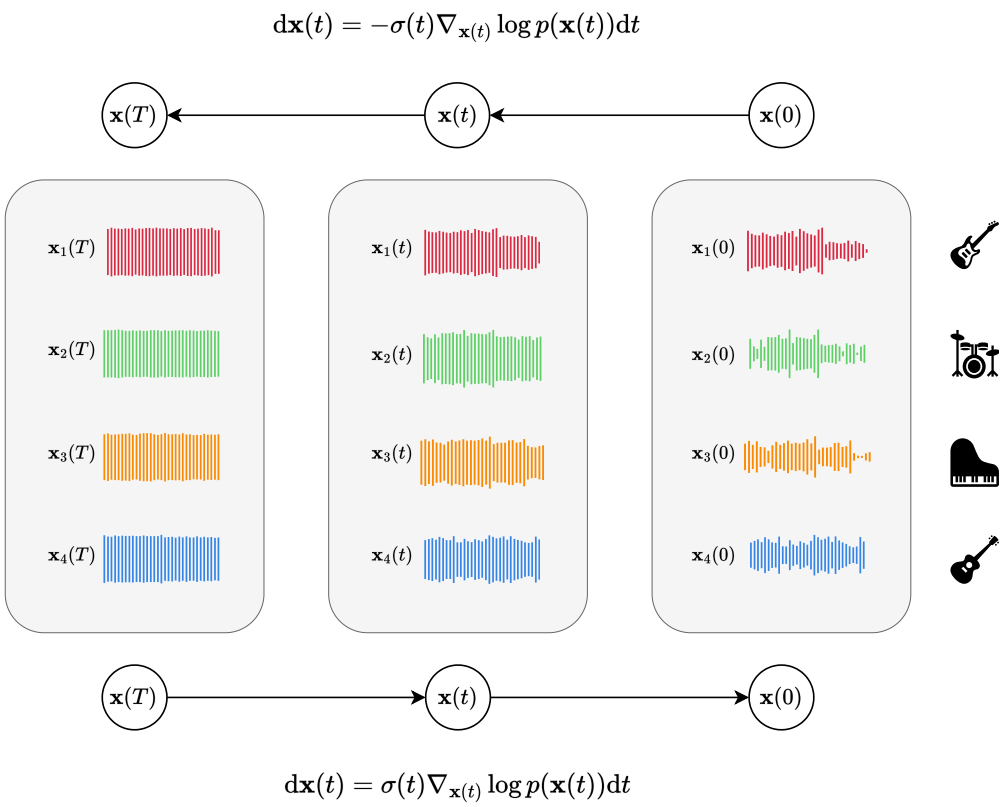

Figure 4.1: An overview of our proposed methodology. We use a forward Gaussian process (illustrated from right to left) to learn the score across contextual sets of instrumental sources (depicted as waveforms within larger rectangles) through various time steps t. In the inference phase, this process is inverted (shown from left to right), enabling operations like total generation, partial generation, and source separation, further elaborated in Figure 4.2.

of separating music sources.

Models aimed at the task of generation are trained to approximate the distribution $p(\mathbf{y})$ across mixtures (or to be more precise, on general audio $\mathbf{x}'$ that can represent both individual sources or mixtures of sources, see Section 3.1.1), and are not able to perform the separation task directly. In this case, we achieve precise modeling of mixtures but we do not have information about the individual sources. Note that methods which model the distribution of mixtures based on textual descriptions [8, 26], face the same challenges without additional analysis[1]. In this chapter we focus on models which are not conditioned by additional information $\mathbf{z}$. Contrarily, source separation models for music [2] focus on either modeling the deterministic counterpart of a conditional distribution $p(\mathbf{x}_1, \ldots, \mathbf{x}_N|\mathbf{y})$ (Chapter 2), which is conditioned on the mix, or on developing a distinct model $p_n(\mathbf{x}_n)$ for the distribution of each source (utilizing a weakly-supervised approach, given that in music we have labels for the different stems), with the mixture serving as the condition during the inference phase, as we discussed thoroughly in the previous chapter. For both scenarios, the creation of mixtures is unfeasible. In the initial scenario, since the model inputs a mixture architecturally, it obstructs the capability for unconditional modeling due to the lack of access to $p(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ (or equivalently, to $p(\mathbf{y})$). In the latter scenario, although we can model each source independently with

[1]We will see in Chapter 5 that by leveraging textual parametrization, it is possible to adapt such models for the tasks outlined in this chapter in a zero-shot fashion.

high precision, the information regarding their inter-dependence is lost, rendering the generation of cohesive mixtures unattainable.

In this chapter, we present the following advancements:

- Initially, we establish a connection between source separation and music generation by training a source-joint generative model (see Section 3.1.3) $p(\mathbf{x}_1, \ldots, \mathbf{x}_N)$, over sources that are contextually related (namely, those that are part of the same musical composition). To achieve this objective, we employ the denoising score-matching framework [22–24] to train a *Multi-Source Diffusion Model (MSDM)*. By training this unified model, we can execute both source separation and music generation tasks during the inference stage. Generation is accomplished by drawing samples from the prior distribution. In contrast, separation is conducted by first conditioning the prior on the given mixture and sampling from the ensuing posterior distribution.

- This novel framework enables a new task in the music generation domain, namely *source imputation*. This approach allows us to create musical accompaniments by generating specific subsets of sources that complement the ones given as input. For example, we can generate a piano piece that harmonizes with a given drum track.

- To achieve results on par with regressor models in source separation [1] on the Slakh2100 dataset [119], we introduce a novel methodology for estimating the posterior score utilizing *Dirac delta functions*. This method leverages the inherent functional dependency between the sources and the mixture.

4.1 Related Methods

As outlined in Section 3.1.2, the approach to source separation through (independent) Bayesian inference involves training a distinct prior model for each source, approximating the distributions $p_n(\mathbf{x}_n)_{n=1,\ldots,N}$. The mixture is observed only at inference time, with a likelihood function establishing the relationship between it and its source components.

The method most closely aligned with MSDM is the NCSN-BASIS algorithm [118], as discussed in the preceding chapter. The reader should recall that it utilizes Langevin Dynamics with an NCSN score-based model to separate the mixtures at inference time and is based on Gaussian likelihood functions. As our experimental results will demonstrate (see Table 4.5), Gaussian likelihoods do not perform as well as the proposed Dirac-based likelihood functions with score-based models. Our approach distinguishes itself from other source separation techniques grounded in (independent) Bayesian inference, including NCSN-BASIS, by its ability to model the entire joint distribution. This capability enables our single model to not only separate sources but also to generate mixtures or subsets of stems.

Contextual relationships among sources are explicitly taken into account in the work by Manilow et al. [1] and in the adversarial permutation invariant training approach we introduced in Chapter 2. The first work explicitly models the source interdependencies via a orderless NADE estimator [166] conditioned on the mixture. The model is trained to predict a subset of sources given the complementary batch. We recall that in our adversarial permutation invariant training method, utilizing a

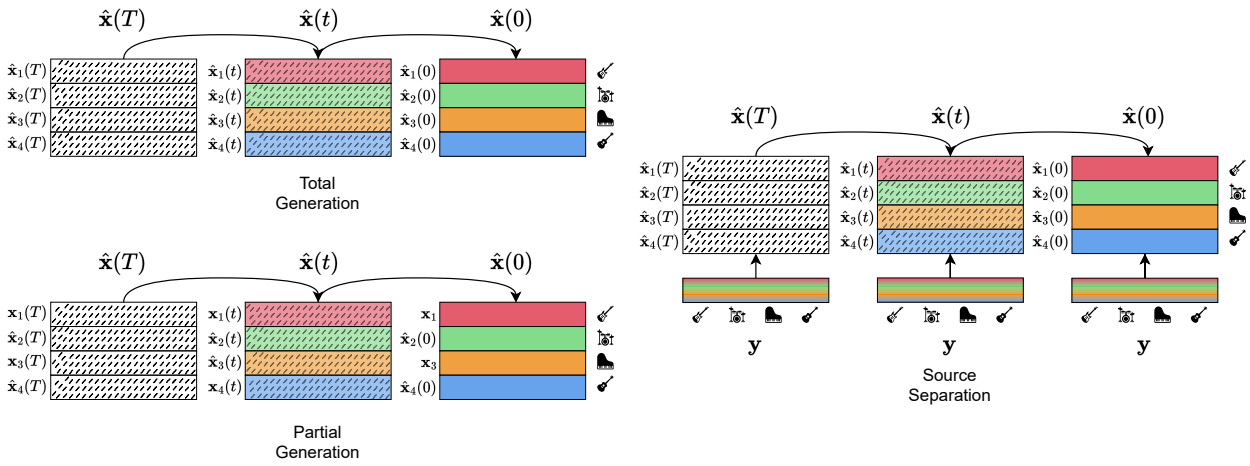

Figure 4.2: Inference modalities with MSDM. The presence of noise within the signal is symbolized by slanted dashes, which reduce progressively from left to right, reaching their peak level of noise at time T, the point at which the sampling process initiates. *Top-left:* The process involves generating every stem within a mixture, leading to a complete generation of all components. *Bottom-left:* Partial generation, or source imputation, is carried out. Given the sources $\mathbf{x}_1$ (Bass) and $\mathbf{x}_3$ (Piano) as input, the remaining sources, $\hat{\mathbf{x}}_2(0)$ (Drums) and $\hat{\mathbf{x}}_4(0)$ (Guitar), are synthesized. The noisy stems derived from $\mathbf{x}_1$ and $\mathbf{x}_3$ are represented by $\mathbf{x}_1(t)$ and $\mathbf{x}_3(t)$, respectively, produced through the perturbation kernel as specified in Eq. (4.1). *Right:* Source separation is achieved by conditioning the prior with a mixture $\mathbf{y}$, as detailed in Algorithm 2.

context-based discriminator we can model the relationship between sources, while instance discrimination alone does not lead to acceptable results. The two techniques operate deterministically and are architecturally designed to depend on the mixtures. A similar architectural restriction is encountered in source separation methods that utilize diffusion-based conditional generation [123, 167] (see Subsection 3.1.1) or deterministic approaches influenced by denoising diffusion [168]. The presented approach uniquely introduces a model not architecturally limited by a mixture conditioner, allowing us to achieve unconditional generation.

4.2 Score-Based Diffusion Models

The foundation of our model lies in estimating the joint distribution of the sources $p(\mathbf{x}_1, \ldots, \mathbf{x}_N)$. Our approach is generative because we model an unconditional distribution (the prior). The different tasks are then solved at inference time, exploiting the prior.

We employ a diffusion-based [21, 24] generative model trained via denoising score-matching [22] to learn the prior. Specifically, we present our formalism by utilizing the notation and assumptions established in [169]. The central idea of score-matching [56, 170, 171] is to approximate the "score" function of the target distribution $p(\mathbf{x})$, namely $\nabla_{\mathbf{x}} \log p(\mathbf{x})$, rather than the distribution itself. To effectively approximate the score in sparse data regions, denoising diffusion methods introduce controlled noise to the data and learn to remove it. Formally, the data distribution is perturbed with a Gaussian perturbation kernel:

$$p(\mathbf{x}(t) \mid \mathbf{x}(0)) = \mathcal{N}(\mathbf{x}(t); \mathbf{x}(0), \sigma^2(t)\mathbf{I}) \,, \tag{4.1}$$

where the parameter $\sigma(t)$ regulates the degree of noise added to the data. Following the authors in [169], we consider an optimal schedule given by $\sigma(t) = t$. With that choice of $\sigma(t)$, the forward

evolution of a data point $\mathbf{x}(t)$ in time is described by a probability flow ODE [23]:

$$\mathrm{d}\mathbf{x}(t) = -\sigma(t)\nabla_{\mathbf{x}(t)} \log p(\mathbf{x}(t))\,\mathrm{d}t\,. \tag{4.2}$$

For $t = T >> 0$, a data point $\mathbf{x}(T)$ is approximately distributed according to a Gaussian distribution $\mathcal{N}(\mathbf{x}(t); \mathbf{0}, \sigma^2(T)\mathbf{I})$, from which sampling is straightforward. Eq. (4.2) can be inverted in time, resulting in the following backward ODE that describes the denoising process:

$$\mathrm{d}\mathbf{x}(t) = \sigma(t)\nabla_{\mathbf{x}(t)} \log p(\mathbf{x}(t))\,\mathrm{d}t\,. \tag{4.3}$$

Sampling can be performed from the data distribution integrating Eq. (4.3) with a standard ODE solver, starting from an initial (noisy) sample drawn from $\mathcal{N}(\mathbf{x}(t); \mathbf{0}, \sigma^2(T)\mathbf{I})$. The score function, represented by a neural network $S_\theta(\mathbf{x}(t), \sigma(t))$, is approximated by minimizing the following score-matching loss:

$$\mathbb{E}_{t\sim\mathcal{U}([0,T])}\mathbb{E}_{\mathbf{x}(0)\sim p(\mathbf{x}(0))}\mathbb{E}_{\mathbf{x}(t)\sim p(\mathbf{x}(t)|\mathbf{x}(0))} \left\| S_\theta(\mathbf{x}(t), \sigma(t)) - \nabla_{\mathbf{x}(t)} \log p\left(\mathbf{x}(t) \mid \mathbf{x}(0)\right)\right\|_2^2 .$$

By expanding $p(\mathbf{x}(t) \mid \mathbf{x}(0))$ with Eq. (4.1), the score-matching loss simplifies to:

$$\mathbb{E}_{t\sim\mathcal{U}([0,T])}\mathbb{E}_{\mathbf{x}(0)\sim p(\mathbf{x}(0))}\mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\sigma^2(t)\mathbf{I})} \left\| D_\theta(\mathbf{x}(0) + \epsilon, \sigma(t)) - \mathbf{x}(0)\right\|_2^2 ,$$

where we compute $S_\theta(\mathbf{x}(t), \sigma(t))$ based on the prediction $D_\theta(\mathbf{x}(t), \sigma(t))$ at step 0:

$$S_\theta(\mathbf{x}(t), \sigma(t)) =: (D_\theta(\mathbf{x}(t), \sigma(t)) - \mathbf{x}(t))/\sigma^2(t)\,. \tag{4.4}$$

4.2.1 Score-Based Diffusion Models for Audio

DiffWave [172] and WaveGrad [173] were the first diffusion (score) based generative models in audio, tackling speech synthesis. Many subsequent models followed these preliminary works, mainly conditioned to solve particular tasks such as speech enhancement [174–177], audio upsampling [178, 179], MIDI-to-waveform [180, 181], or spectrogram-to-MIDI generation [182]. The first work in source-specific generation with diffusion models is CRASH [183]. [184–186] proposed text-conditioned diffusion models to generate general sounds, not focusing on restricted classes such as speech or music. Closer to our work, diffusion models targeting the musical domain are Riffusion [187] and Moûsai [26]. Riffusion fine-tunes Stable Diffusion [106], a large pre-trained text-conditioned vision diffusion model, over STFT magnitude spectrograms. Moûsai performs generation in a latent domain, resulting in context lengths that surpass the minute. Our score network follows the design of the U-Net proposed in Moûsai, albeit using the waveform data representation.

4.3 Multi-Source Audio Diffusion Models

In our setup, we have N distinct source waveforms $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with $\mathbf{x}_n \in \mathbb{R}^D$ for each n. The sources coherently sum to a mixture $\mathbf{y} = \sum_{n=1}^{N} \mathbf{x}_n$. We sometimes use the aggregated form $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \in \mathbb{R}^{N\times D}$.

In this setting, multiple tasks can be performed: one may generate a consistent mixture $\mathbf{y}$ or

separate the individual sources $\mathbf{x}$ from a given mixture $\mathbf{y}$. We refer to the first task as *generation* and the second as *source separation*. A subset of sources can also be fixed in the generation task, and the others can be generated consistently. We call this task *partial generation* or *source imputation*. Our key contribution is the ability to perform all these tasks simultaneously by training a single multi-source diffusion model (MSDM), capturing the prior $p(\mathbf{x}_1, \ldots, \mathbf{x}_N)$. The model, illustrated in Figure 4.1, approximates the noisy score function:

$$\nabla_{\mathbf{x}(t)} \log p(\mathbf{x}(t)) = \nabla_{(\mathbf{x}_1(t),\ldots,\mathbf{x}_N(t))} \log p(\mathbf{x}_1(t), \ldots, \mathbf{x}_N(t)) \,,$$

with a neural network:

$$S_\theta(\mathbf{x}(t), \sigma(t)) : \mathbb{R}^{N\times D} \times \mathbb{R} \to \mathbb{R}^{N\times D} \,, \tag{4.5}$$

where $\mathbf{x}(t) = (\mathbf{x}_1(t), \ldots, \mathbf{x}_N(t))$ denotes the sources perturbed with the Gaussian kernel in Eq. (4.1). We describe the three tasks (illustrated in Figure 4.2) using the prior distribution:

- *Total Generation.* This task requires generating a plausible mixture $\mathbf{y}$. It can be achieved by sampling the sources $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$ from the prior distribution and summing them to obtain the mixture $\mathbf{y}$.
- *Partial Generation.* Given a subset of sources, this task requires generating a plausible accompaniment. We define the subset of fixed sources as $\mathbf{x}_\mathcal{I}$ and generate the remaining sources $\mathbf{x}_{\overline{\mathcal{I}}}$ by sampling from the conditional distribution $p(\mathbf{x}_{\overline{\mathcal{I}}} \mid \mathbf{x}_\mathcal{I})$.
- *Source Separation.* Given a mixture $\mathbf{y}$, this task requires isolating the individual sources that compose it. It can be achieved by sampling from the posterior distribution $p(\mathbf{x} \mid \mathbf{y})$.

The three tasks of our method are solved during inference by discretizing the backward Eq. (4.3). Although different tasks require distinct score functions, they all originate directly from the prior score function in Eq. (4.5). We analyze each of these score functions in detail. For more details on the discretization method, refer to Section 4.3.4.

4.3.1 Total Generation

The total generation task is performed by sampling from Eq. (4.3) using the score function in Eq. (4.5). The mixture is then obtained by summing over all the generated sources.

4.3.2 Partial Generation

In the partial generation task, we fix a subset of source indices $\mathcal{I} \subset \{1, \ldots, N\}$ and the corresponding sources $\mathbf{x}_\mathcal{I} := \{\mathbf{x}_n\}_{n\in\mathcal{I}}$. The goal is to generate the remaining sources $\mathbf{x}_{\overline{\mathcal{I}}} := \{\mathbf{x}_n\}_{n\in\overline{\mathcal{I}}}$ consistently, where $\overline{\mathcal{I}} = \{1, \ldots, N\} - \mathcal{I}$. To do so, we estimate the gradient of the conditional distribution:

$$\nabla_{\mathbf{x}_{\overline{\mathcal{I}}}(t)} \log p(\mathbf{x}_{\overline{\mathcal{I}}}(t) \mid \mathbf{x}_\mathcal{I}(t)). \tag{4.6}$$

This falls into the setting of imputation or, as it is more widely known in the image domain, inpainting. We approach imputation using the method in [23]. The gradient in Eq. (4.6) is

Algorithm 2 "MSDM Dirac" sampler for source separation.

Require: I number of discretization steps for the ODE, R number of corrector steps, $\{\sigma_i\}_{i\in\{0,\dots,I\}}$ noise schedule, S_{churn}

Initialize $\hat{\mathbf{x}} \sim \mathcal{N}(0, \sigma_I^2\mathbf{I})$
$\alpha \leftarrow \min(S_{\text{churn}}/I, \sqrt{2}-1)$
for $i \leftarrow I$ **to** 1 **do**
  for $r \leftarrow R$ **to** 0 **do**
    $\hat{\sigma} \leftarrow \sigma_i \cdot (\alpha + 1)$
    $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
    $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} + \sqrt{\hat{\sigma}^2 - \sigma_i^2}\epsilon$
    $\mathbf{z} \leftarrow [\hat{\mathbf{x}}_{1:N-1}, \mathbf{y} - \sum_{n=1}^{N-1} \hat{\mathbf{x}}_n]$
    for $n \leftarrow 1$ **to** $N-1$ **do**
      $\mathbf{g}_n \leftarrow S_{\theta,n}(\mathbf{z}, \hat{\sigma}) - S_{\theta,N}(\mathbf{z}, \hat{\sigma})$
    end for
    $\mathbf{g} \leftarrow [\mathbf{g}_1, \dots, \mathbf{g}_{N-1}]$
    $\hat{\mathbf{x}}_{1:N-1} \leftarrow \hat{\mathbf{x}}_{1:N-1} + (\sigma_{i-1} - \hat{\sigma})\mathbf{g}$
    $\hat{\mathbf{x}} \leftarrow [\hat{\mathbf{x}}_{1:N-1}, \mathbf{y} - \sum_{n=1}^{N-1} \hat{\mathbf{x}}_n]$
    if $r > 0$ **then**
      $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
      $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2}\epsilon$
    end if
  end for
end for
return $\hat{\mathbf{x}}$

approximated as follows:

$$\nabla_{\mathbf{x}_{\overline{\mathcal{I}}}(t)} \log p([\mathbf{x}_{\overline{\mathcal{I}}}(t), \hat{\mathbf{x}}_{\mathcal{I}}(t)])\,,$$

where $\hat{\mathbf{x}}_{\mathcal{I}}$ is a sample from the forward process: $\hat{\mathbf{x}}_{\mathcal{I}}(t) \sim \mathcal{N}(\mathbf{x}_{\mathcal{I}}(t); \mathbf{x}_{\mathcal{I}}(0), \sigma(t)^2\mathbf{I})$. The square bracket operator denotes concatenation. Approximating the score function, we write:

$$\nabla_{\mathbf{x}_{\overline{\mathcal{I}}}(t)} \log p(\mathbf{x}_{\overline{\mathcal{I}}}(t) \mid \mathbf{x}_{\mathcal{I}}(t)) \approx S_{\theta,\overline{\mathcal{I}}}([\mathbf{x}_{\overline{\mathcal{I}}}(t), \hat{\mathbf{x}}_{\mathcal{I}}(t)], \sigma(t))\,,$$

where $S_{\theta,\overline{\mathcal{I}}}$ denotes the entries of the score network corresponding to the sources indexed by $\overline{\mathcal{I}}$.

4.3.3 Source Separation

We view source separation as a specific instance of conditional generation, where we condition the generation process on the given mixture $\mathbf{y} = \mathbf{y}(0)$. This requires computing the score function of the posterior distribution:

$$\nabla_{\mathbf{x}(t)} \log p(\mathbf{x}(t) \mid \mathbf{y}(0))\,. \tag{4.7}$$

Standard methods for implementing conditional generation for diffusion models involve directly estimating the posterior score in Eq. (4.7) at training time (i.e., Classifier Free Guidance, as described in [25]) or estimating the likelihood function $p(\mathbf{y}(0) \mid \mathbf{x}(t))$ and using the Bayes formula to derive the posterior. The second approach typically involves training a separate model, often a classifier, for the score of the likelihood function as in Classifier Guided conditioning, outlined in

[188].

In diffusion-based generative source separation, learning a likelihood model is typically unnecessary because the relationship between $\mathbf{x}(t)$ and $\mathbf{y}(t)$ is represented by a simple function, namely the sum. A natural approach is to model the likelihood function based on such functional dependency. This is the approach taken by [118], where they use a Gaussian likelihood function:

$$p(\mathbf{y}(t) \mid \mathbf{x}(t)) = \mathcal{N}(\mathbf{y}(t) \mid \sum_{n=1}^{N} \mathbf{x}_n(t), \gamma^2(t)\mathbf{I}), \tag{4.8}$$

with the standard deviation given by a hyperparameter $\gamma(t)$. The authors argue that aligning the $\gamma(t)$ value to be proportionate to $\sigma(t)$ optimizes the outcomes of their NCSN-BASIS separator.

We present a novel approximation of the posterior score function in Eq. (4.7) by modeling $p(\mathbf{y}(t) \mid \mathbf{x}(t))$ as a Dirac delta function centered in $\sum_{n=1}^{N} \mathbf{x}_n(t)$:

$$p(\mathbf{y}(t) \mid \mathbf{x}(t)) = \mathbb{1}_{\mathbf{y}(t)=\sum_{n=1}^{N} \mathbf{x}_n(t)}\,. \tag{4.9}$$

The complete derivation can be found in Appendix B.1, and we present only the final formulation, which we call "MSDM Dirac". The method constrains a source, without loss of generality $\mathbf{x}_N$, by setting $\mathbf{x}_N(t) = \mathbf{y}(0) - \sum_{n=1}^{N-1} \mathbf{x}_n(t)$ and estimates:

$$\begin{aligned}\nabla_{\mathbf{x}_m(t)} \log p(\mathbf{x}(t) \mid \mathbf{y}(0)) \approx\ & S_{\theta,m}((\mathbf{x}_1(t), \ldots, \mathbf{x}_{N-1}(t), \mathbf{y}(0) - \sum_{n=1}^{N-1} \mathbf{x}_n(t)), \sigma(t)) \\ & - S_{\theta,N}((\mathbf{x}_1(t), \ldots, \mathbf{x}_{N-1}(t), \mathbf{y}(0) - \sum_{n=1}^{N-1} \mathbf{x}_n(t)), \sigma(t))\,,\end{aligned} \tag{4.10}$$

where $1 \leq m \leq N-1$ and $S_{\theta,m}, S_{\theta,N}$ denote the entries of the score network corresponding to the m-th and N-th sources. Our approach models the limiting case wherein $\gamma(t) \to 0$ in the Gaussian likelihood function. This represents a scenario where the dependence between $\mathbf{x}(t)$ and $\mathbf{y}(t)$ becomes increasingly tight, sharpening the conditioning on the given mixture during the generation process.

The separation procedure can be additionally employed in the weakly-supervised source separation scenario, typically encountered in (independent) Bayesian source separation [28, 118, 132] (Section 3.1.2). We remind the reader that this scenario pertains to cases where we know that specific audio data belongs to a particular instrument class, but we do not have access to sets of sources that share a context. To adapt to this scenario, we assume independence between sources $p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \prod_{n=1}^{N} p_n(\mathbf{x}_n)$ and train a separate model for each source class. We call the resulting model "Independent Source Diffusion Model (ISDM) with Dirac Likelihood" or "ISDM Dirac". We derive its formula and formulas for the Gaussian likelihood versions "MSDM Gaussian" and "ISDM Gaussian" in Appendix B.2. While the ISDM method lacks generative capabilities, in the sense that it cannot produce a full mix, it enables us to demonstrate the effectiveness of generative source separation when combined with Dirac likelihood.

4.3.4 The Sampler

We use a first-order ODE integrator based on the Euler method and introduce stochasticity following [169]. The amount of stochasticity is controlled by the parameter S_{churn}. As shown in Table 4.4

Table 4.1: Inference time for a 12-second separation, and number of parameters for each model in Table 4.5. Demucs + Gibbs (256 steps) was added because 256 is the minimum number of steps that makes the SI-SDR$_\text{I}$ over all instruments (17.59) greater than the one of ISDM. While ISDM and MSDM are not time-competitive to Demucs, they are more time-efficient compared to Demucs + Gibbs (256 and 512 steps).

Model	Inference Time (s)	# of parameters
Demucs	$0.111_{\pm 0.071}$	40M
Demucs + Gibbs (512 steps)	$0.111_{\pm 0.071} \times 512 = 56.832_{\pm 36.352}$	$\sim$ 40M
Demucs + Gibbs (256 steps)	$0.111_{\pm 0.071} \times 256 = 28.416_{\pm 18.176}$	$\sim$ 40M
ISDM (correction)	$4.6_{\pm 0.345} \times 4 = 18.4_{\pm 1.38}$	405M $\times 4$
MSDM (correction)	$4.6_{\pm 0.345}$	405M

and explained in detail in [169], stochasticity significantly improves sample quality.

We implemented a correction mechanism [23, 118] iterating for R steps after each prediction step i, adding additional noise and re-optimizing with the score network fixed at σ_i. This correction procedure entails injecting additional noise and then re-denoising at each denoising step i employing the score network fixed at σ_i. This process is repeated R times for each denoising step i. The pseudocode for the "MSDM Dirac" source separation sampler is outlined in Algorithm 2.

As per [169], we adopt a non-linear schedule for time discretization that gives more importance to lower noise levels. It is defined as:

$$t_i = \sigma_i = \sigma_{\max}^{\frac{1}{\rho}} + \frac{i}{I-1}(\sigma_{\min}^{\frac{1}{\rho}} - \sigma_{\max}^{\frac{1}{\rho}})^{\rho}\,,$$

where $0 \leq i < I$, with I the number of discretization steps. We set $\sigma_{\min} = 10^{-4}$, $\sigma_{\max} = 1, \rho = 7$.

4.4 Experimental Setup

4.4.1 Datasets

Our experiments are conducted on the Slakh2100 dataset [119], which was also utilized in the preceding chapter. In order to maintain a fair comparison with the work of [1], to whom we compare our results, we restrict our usage to the dataset's four most frequently occurring classes: Bass, Drums, Guitar, and Piano. These instruments feature prominently in the vast majority of the tracks, with their occurrence rates being 94.7% for Bass, 99.3% for Drums, 100.0% for Guitar, and 99.3% for Piano.

MUSDB18-HQ [134] serves as the uncompressed variant (in WAV format) of the MUSDB18 dataset [14], a benchmark dataset for the music source separation task. It contains 150 tracks, with 100 allocated for training and 50 for testing, amounting to roughly 10 hours of professional-grade audio. Each piece within the dataset is separated into the stems: Bass, Drums, Vocals, and Other, with the latter encompassing any elements not included in the categories above.

We recall from Section 3.4.2 that Slakh2100 contains $\sim$150h of audio, more than 10$\times$ the quantity of data in MUSDB18, although not reaching the same level of quality of the latter, given that the tracks are synthesized from MIDI. The volume of data is critical in generative modeling, which positions Slakh2100 as a more suitable dataset for our research. Despite this, we also extend our analysis to include MUSDB18-HQ, as detailed in Table 4.6.

Table 4.2: Comparison between total generation capabilities of MSDM (Slakh2100) and an equivalent architecture trained on Slakh2100 mixtures. Both subjective (quality and coherence, higher is better) and objective (FAD, lower is better) evaluations are shown. The quality and coherence columns refer to the average scores of the listening tests, with respective variances.

Model	**FAD** ↓	**Quality** ↑	**Coherence** ↑
MSDM	6.55	6.44 ± 2.12	6.34 ± 2.37
Mixture Model	6.67	6.04 ± 2.48	5.63 ± 2.65

Table 4.3: Quantitative and qualitative results for the partial generation task on Slakh2100. We use both subjective (quality and density, higher is better) and objective (sub-FAD, lower is better) evaluation metrics. The sub-FAD metric is reported for all combinations of generated sources (**B**: Bass, **D**: Drums, **G**: Guitar, **P**: Piano). The quality and density columns refer to the average scores of the listening tests, with respective variances.

Slakh2100	**B**	**D**	**G**	**P**	**BD**	**BG**	**BP**	**DG**	**DP**	**GP**	**BDG**	**BDP**	**BGP**	**DGP**	**Quality**	**Density**
MSDM	0.45	1.09	0.11	0.76	2.09	1.00	2.32	1.45	1.82	1.65	2.93	3.30	4.90	3.10	6.2 ± 2.6	6.1 ± 2.6

4.4.2 Architectures and Training

The implementation of the score network is based on a time domain (non-latent) unconditional version of Moûsai [26]. We used the publicly available repository `audio-diffusion-pytorch/v0.0.432`[2]. The score network is a U-Net [189] comprised of encoder, bottleneck, and decoder with skip connections between the encoder and the decoder. The encoder has six layers comprising two convolutional ResNet blocks, followed by multi-head attention in the final three layers. The signal sequence is downsampled in each layer by a factor of 4. The number of channels in the encoder layers is [256, 512, 1024, 1024, 1024, 1024]. The bottleneck consists of a ResNet block, followed by self-attention, and another ResNet block (all with 1024 channel layers). The decoder follows a reverse symmetric structure with respect to the encoder. We employ `audio-diffusion-pytorch-trainer`[3] for training. We downsample data to 22kHz and train the score network with four stacked mono channels for MSDM (i.e., one for each stem) and one mono channel for each model in ISDM, using a context length of $\sim$ 12 seconds. All our models were trained until convergence on an NVIDIA RTX 6000 GPU with 24 GB of VRAM. We trained all our models using Adam [98], with a learning rate of 10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.99$ and a batch size of 16. Inference times and number of parameters are reported in Table 4.1.

4.5 Experimental Results

4.5.1 Music Generation

The performance of MSDM on the generative tasks is tested through subjective and objective evaluation.

Subjective evaluation is done through listening tests, whose form format is reported in Figure 4.3. Concisely, we produce two forms, one for total generation and one for partial generation. In

[2] `https://github.com/archinetai/audio-diffusion-pytorch/tree/v0.0.43`
[3] `https://github.com/archinetai/audio-diffusion-pytorch-trainer/tree/79229912`

the first, subjects are asked to rate, from 1 to 10, the *quality* and instrument *coherence* (i.e., how the instruments sound plausible together) of 30 generated chunks, of which 15 are generated by MSDM and 15 by a model trained on mixtures (using the same diffusion architecture as MSDM). In the second one, knowing the fixed instruments, subjects are asked to rate, from 1 to 10, the *quality* and the *density* of the generated accompaniment. Namely, 'quality' tests how the full chunk sounds plausible with respect to the ground truth data, and 'density' tests how much the generated instruments are present in the chunk. We also provide examples of mixture and accompaniment generation[4].

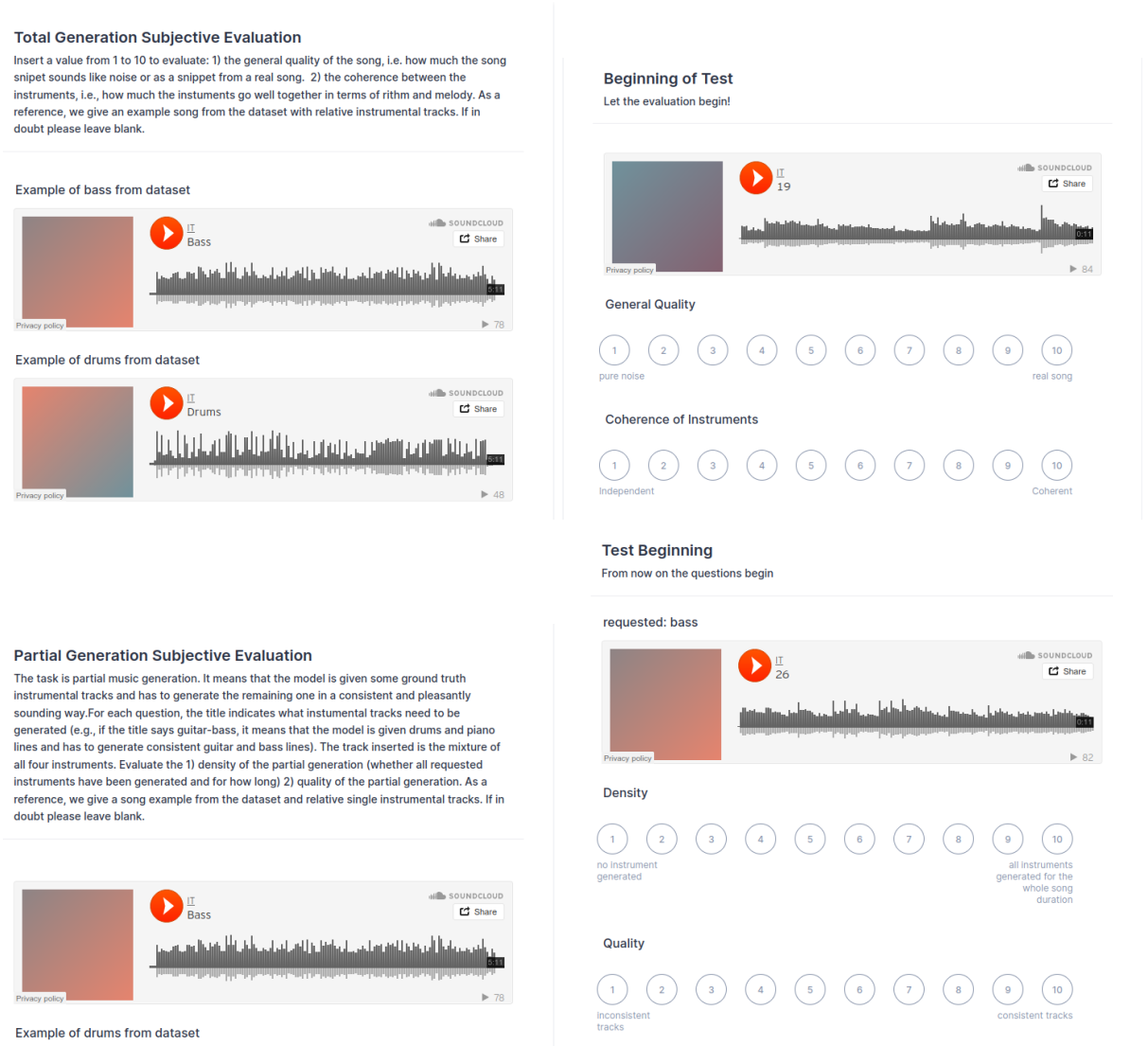

Figure 4.3: Snippets from the subjective evaluation form. The first row is relative to total generation, there people were asked to evaluate 30 songs, of which 15 were from the mixture model and 15 from MSDM. 45 people answered the survey. The second row is relative to partial generation. Subjects were asked to evaluate 15 songs. For each song, a random subset of sources is fixed and the other are generated by MSDM. The requested sources are explicitly stated above the song (e.g., in the snippet, the model has to generate only the bass). 25 subjects answered.

As for the objective evaluation of the generative tasks, we use the Fréchet Audio Distance (FAD) [190] metric with VGGish embeddings [191]. The recent study [192] shows that using using VGGish

[4] `https://gladia-research-group.github.io/multi-source-diffusion-models/`

Table 4.4: Hyperparameter search for source separation using "MSDM Dirac" (top-left), "ISDM Dirac" (bottom-left), "MSDM Gaussian" (top-right) and "ISDM Gaussian" (bottom-right) posteriors. We report the SI-SDR$_\text{I}$ values in dB (higher is better) averaged over all instruments (Bass, Drums, Piano, Guitar).

		Dirac Likelihood				**Gaussian Likelihood**						
	S_churn	Constrained Source				$\gamma(t)$						
		Bass	Drums	Guitar	Piano	$0.25\sigma(t)$	$0.5\sigma(t)$	$0.75\sigma(t)$	$1\sigma(t)$	$1.25\sigma(t)$	$1.5\sigma(t)$	$2\sigma(t)$
MSDM	0	4.41	5.05	3.28	2.87	-41.54	6.37	6.05	5.67	5.729	5.13	4.33
	1	7.90	8.18	7.03	7.05	-47.24	6.79	6.51	6.15	6.19	5.66	4.45
	20	**14.29**	12.99	12.19	11.69	-47.17	11.07	10.51	9.43	10.19	9.18	7.58
	40	14.28	13.02	5.51	4.78	-47.17	-36.92	**12.48**	11.25	11.87	10.80	9.03
ISDM	0	5.05	3.69	-2.50	6.93	-45.46	7.12	6.50	5.78	5.02	4.49	3.69
	1	9.23	8.57	7.28	9.20	-47.54	7.57	7.20	6.32	5.35	4.82	3.83
	20	15.35	15.08	13.20	15.36	-46.86	12.89	12.21	10.87	9.32	8.32	6.47
	40	**17.26**	15.77	15.30	14.98	-46.86	-35.97	**14.09**	12.82	10.85	10.02	8.26
	60	16.21	15.57	15.51	14.20	-46.80	-46.85	14.06	12.57	11.83	10.81	9.24

embeddings is not as reliable as CLAP [193] or EnCodec [148] embeddings. We opted for VGGish embeddings due to their widespread adoption within the research community. Additionally, the foundational work for this chapter was completed prior to the release of [192], which highlighted potential limitations of VGGish embeddings for computing the FAD metric.

We generalize the evaluation protocol in [151] to our total generation task and to partial generation with more than one source. Given D_real a dataset of ground truth mixtures chunks and $\mathcal{I}$ a set indexing conditioning sources ($\emptyset$ for total generation), we build a dataset D_gen whose elements are the sum between conditioning sources (indexed by $\mathcal{I}$) an the respective generated sources. We define the *sub-FAD* as FAD($D_\text{real}, D_\text{gen}$). Our method is the first able to generate any combination of partial sources, and as such, we do not have a competitor baseline. We thus report the sub-FAD results of our method as baseline metrics for future research, together with listening test results.

Results for total and partial generations are reported in Tables 4.2 and 4.3 respectively, both for subjective and objective evaluations. Results in Table 4.2 show a minimal difference between the model trained on mixtures and MSDM. This suggests that, given the same dataset and architecture, the generative power of MSDM is the same as the model trained on mixtures while being able to perform separation and partial generation. Table 4.3 shows via the subjective results that the task of partial generation can be performed with non-trivial quality. Our method being the first able to generate any combination of partial sources, does not have a competitor baseline for the objective metrics. We thus report the sub-FAD results of our method as baseline metrics for future research.

4.5.2 Source Separation

In order to evaluate source separation, we use the Scale-Invariant SDR Improvement (SI-SDR$_\text{I}$) metric [194]. We recall from Chapter 2 that the SI-SDR between a ground-truth source $\mathbf{x}_n$ and an estimate $\hat{\mathbf{x}}_n$ is defined as:

$$\text{SI-SDR}(\mathbf{x}_n, \hat{\mathbf{x}}_n) = 10 \log_{10} \frac{\|\alpha \mathbf{x}_n\|^2 + \epsilon}{\|\alpha \mathbf{x}_n - \hat{\mathbf{x}}_n\|^2 + \epsilon},$$

where $\alpha = \frac{\mathbf{x}_n^\top \hat{\mathbf{x}}_n + \epsilon}{\|\mathbf{x}_n\|^2 + \epsilon}$. We set $\epsilon = 10^{-8}$. Also recall that the improvement with respect to the mixture baseline is defined as SI-SDR$_\text{I}$ = SI-SDR$(\mathbf{x}_n, \hat{\mathbf{x}}_n)$ − SI-SDR$(\mathbf{x}_n, \mathbf{y})$.

Table 4.5: Quantitative results for source separation on the Slakh2100 test set. We use the SI-SDR$_\text{I}$ as our evaluation metric (dB – higher is better). We present both the supervised ("MSDM Dirac", "MSDM Gaussian") and weakly-supervised ("ISDM Dirac", "ISDM Gaussian") separators and specify if a correction step is used. "All" reports the average over the four stems. The results show that: (i) Dirac likelihood improves overall results, even outperforming the state of the art when applied to ISDM (ii) adding a correction step is beneficial (iii) MSDM with Dirac likelihood and one step of correction gives results comparable with the state of the art and superior to the Demucs model trained in [1] overall. We stress again that while the baselines are trained on the separation task alone, MSDM is able to perform also generative tasks.

Model	**Bass**	**Drums**	**Guitar**	**Piano**	**All**
Demucs [1, 2]	15.77	19.44	15.30	13.92	16.11
Demucs + Gibbs (512 steps) [1]	17.16	19.61	**17.82**	**16.32**	**17.73**
Dirac Likelihood					
ISDM	18.44	20.19	13.34	13.25	16.30
ISDM (correction)	**19.36**	**20.90**	14.70	14.13	17.27
MSDM	16.21	17.47	12.71	13.29	14.92
MSDM (correction)	17.12	18.68	15.38	14.73	16.48
Gaussian Likelihood [118]					
ISDM	13.48	18.09	11.93	11.17	13.67
ISDM (correction)	14.27	19.10	12.74	12.20	14.58
MSDM	12.53	16.82	12.98	9.29	12.90
MSDM (correction)	13.93	17.92	14.19	12.11	14.54

Slakh2100

On Slakh2100, we compare our supervised MSDM and weakly-supervised MSDM with the "Demucs" [2] and "Demucs + Gibbs (512 steps)" regressor baselines from [1], the state-of-the-art for supervised music source separation on Slakh2100, aligning with the evaluation procedure of [1]. We evaluate over the test set of Slakh2100, using chunks of 4 seconds in length (with an overlap of two seconds) and filtering out silent chunks and chunks consisting of only one source, given the poor performance of SI-SDR$_\text{I}$ on such segments.

We conduct a hyperparameter search over S_churn to evaluate the importance of stochasticity in source separation over a fixed subset of 100 chunks of the Slakh2100 test set, each spanning 12 seconds (selected randomly). To provide a fair comparison between the Dirac ("MSDM Dirac", "ISDM Dirac") and Gaussian ("MSDM Gaussian", "ISDM Gaussian") posterior scores, we execute a search over their specific hyperparameters, namely the constrained source for the Dirac separators and the $\gamma(t)$ coefficient for the Gaussian separators. Results are illustrated in Table 4.4. We observe that: *(i)* stochasticity proves beneficial for all separators, given that the highest values of SI-SDR$_\text{I}$ are achieved with $S_\text{churn} = 20$ and $S_\text{churn} = 40$, *(ii)* using the Dirac likelihood we obtain higher values of SI-SDR$_\text{I}$ with respect to the Gaussian likelihood, both with the MSDM and ISDM separators, and *(iii)* the ISDM separators perform better than the contextual MSDM separators (at the expense of not being able to perform total and partial generation).

We report results comparing our Dirac score posterior with the Gaussian score posterior of [118], using the best parameters of the ablations in Table 4.4 and 150 inference steps. Results are illustrated and discussed in Table 4.5. Concisely, MSDM proves to be very close to the state of the art. Moreover, the newly defined sampling procedure, when used in the weakly supervised

Table 4.6: Comparison of results of MSDM and Demucs v2 [2]. We report the SI-SDR$_I$ values in dB (higher is better). The network is the same as the one trained on Slakh2100 but the sampling rate is 44kHz and it is trained on chunks of length 6 seconds.

	Tested on MUSDB18-HQ			Finetuned on MUSDB18-HQ			Trained on MUSDB				
Model	Bass	Drums	All	Bass	Drums	All	Bass	Drums	Other	Vocals	All
Demucs v2	-	-	-	-	-	-	13.28	11.53	8.59	16.80	12.55
MSDM	-0.83	-0.94	-0.88	3.46	5.03	4.25	4.87	3.28	1.97	6.83	4.24

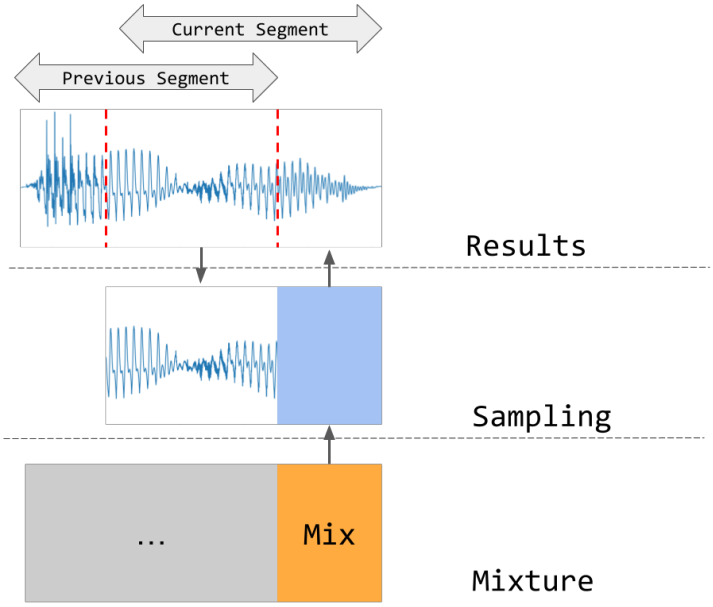

Figure 4.4: Autoregressive-sampling for source separation with score-based diffusion.

flavor, yields results that are better than the state of the art for some stems. Surprisingly, it performs better than contextual MSDM. As such we invalidate the hypothesis in Section 3.1.3, which nevertheless proved useful for the development of this new line of research. However, the reader should notice that 4 independent separators have 4 times more parameters than the base MSDM. In future research we plan to investigate more thoroughly the importance of modeling the context in music source separation.

MUSDB18-HQ

We report in Table 4.6 the results of MSDM and Demucs v2 [2] on the MUSDB18-HQ test set. We try three different strategies, we first check the out-of-distribution ability of the model trained on Slakh2100 by testing directly on MUSDB18-HQ. Then, we tried finetuning the model trained on Slakh2100 on MUSDB18-HQ, and finally, we trained directly on MUSDB18-HQ. Since the only stems that MUSDB18-HQ and Slakh2100 share are "Bass" and "Drums", the first and second strategies could be tested only on these two stems.

Table 4.7: Results on duet singing voices separation evaluated on MedleyVox [3].

Methods	SI-SDR$_\text{I}$	SDR$_\text{I}$
iSRNet [3]	15.10	14.20
NMF	5.12	5.97
Naive	6.61 ± 0.25	7.60 ± 0.21
Segmented	11.14 ± 0.48	11.77 ± 0.47
AR (proposed)	**11.24** ± 0.40	**11.89** ± 0.34
AR w/ TF	11.75 ± 0.38	12.34 ± 0.39

4.6 Applications to Singing Voice Separation

In the Sound Demixing Workshop 2023 paper [30], we use ISDM with the Dirac separator (Eq. (4.10)) for the task of separating duet singing voices. We leverage the model's capacity for zero-shot learning in handling audio sources of similar timbre, such as singing voices, without necessitating vast amounts of paired training data. We trained our model over 8 public singing voice datasets comprising over more than 104h of audio data [195, 196]. By dividing the audio mixture into overlapping segments and employing an auto-regressive sampling approach conditioned on preceding segments (AR), the model demonstrates improved consistency in singer identity across separated audio tracks (see Figure 4.4). We evaluated this sampling strategy on the MedleyVox dataset [3], as shown in Table 4.7 confirming its superior performance over blind methods such as NMF and showing that it improves substantially over the direct application of Eq. 4.10 on long audio segments (Naive). The naive application of the sampler causes voices to swap over longer time spans, decreasing performance. Also performing separation on short non-overlapping chunks (Segmented) improves metrics. Nevertheless, the method is bounded by the supervised baseline iSRNet [3].

The complete details, including source code and pre-trained models, are accessible at the repository `duet-svs-diffusion`[5].

4.7 Summary and Prospects

We have presented a general method, based on denoising score-matching, for source separation, mixture generation, and accompaniment generation in the musical domain. Our approach utilizes a single neural network trained once, with tasks differentiated during inference. Moreover, we have defined a new sampling method for source separation. We quantitatively tested the model on source separation, obtaining results comparable to state-of-the-art regressor models. We qualitatively and quantitaively tested the model on total and partial generation. For the first one we showed the model has the same generative power of the same model trained on mixtures. For the latter, we showed the accompaniment generated are plausible and nontrivial.

Our model's ability to handle both total and partial generation and source separation positions it as a significant step toward the development of general audio models for music. This flexibility paves the way for more advanced music composition tools, where users can easily control and manipulate individual sources within a mixture.

[5] `https://github.com/yoyololicon/duet-svs-diffusion`

Additionally, it would be intriguing to explore the possibility of extending our method to situations where the sub-signals are not related by addition but rather by a known but different function.

The amount of available supervised data constrains the performance of our model. To address this, pre-separating mixtures and training on the separations, as demonstrated in [151], can prove beneficial. In the following chapter we ask how we can relax the data assumptions on the model, in order to not depend on supervised datasets, neither pre-separating. In a certain way, we want to return to the independent Bayesian setting, where we can process single instances, but at the same time we want to perform the same tasks as delineated in this chapter: total generation, partial generation and source separation. Source separation is easier to frame in such setting, given that we can apply the ISDM method. But, for example, how can we perform accompaniment generation with independent Bayesian inference? In such a case the context is missing, making, at least at first glance, the task intractable.

Chapter 5

Generalized Multi-Source Inference for Text Conditioned Music Diffusion Models

We have already discussed in the previous chapter that the task of musical generation has seen significant advancements recently, thanks to developments in generative models. The families of generative models showcasing state-of-the-art results are latent language models [19] (Section 3.2) and (score-based) diffusion models [22–24] (Section 4.2). Despite differences between these generative models, they typically share some mechanisms for conditioning on rich textual embeddings, obtained either using text-only encoders [197] or audio-text contrastive encoders [149, 193, 198]. Such a mechanism allows generating a musical track following a natural language prompt.

Generative models for music typically output only a final mixture. As such, generating the constituent sources is challenging. This implies that musical generative models are hard to employ in music production tasks, where the subsequent manipulation of sub-tracks, creation of accompaniments, and source separation is often required. Two approaches aim to address this issue. The first approach, Multi-Source Diffusion Model (MSDM), was presented in the previous chapter. We recall that we trained a diffusion model in time domain on (supervised) sets of coherent sources viewed as different channels, without conditioning on textual information. Such a model allowed for generating a set of coherent sources, creating accompaniments, and performing source separation. Despite being a versatile compositional model for music, MSDM has three limitations: *(i)* It requires knowledge of separated coherent sources, which are hard to acquire. *(ii)* It architecturally assumes a fixed number of sources and their respective class type (e.g., Bass, Drums, Guitar, Piano). *(iii)* It is impossible to condition the sources on rich semantic information, as commonly done with text-conditioned music models. The second approach, based on supervised instruction prompting [199, 200], fine-tunes a latent diffusion model with instructions that allow adding, removing, and extracting sources present in a musical track. Although this approach addresses the issues *(ii)* and *(iii)* of MSDM, it does not solve the problem *(i)*, necessitating pre-separated supervised data. A strategy for scaling both models is training with data obtained by separating sources from mixtures using a pre-trained separator [151]. This approach, though, is not flexible because such separated data contains artifacts, and we are limited to the number and type of sources the separator can handle.

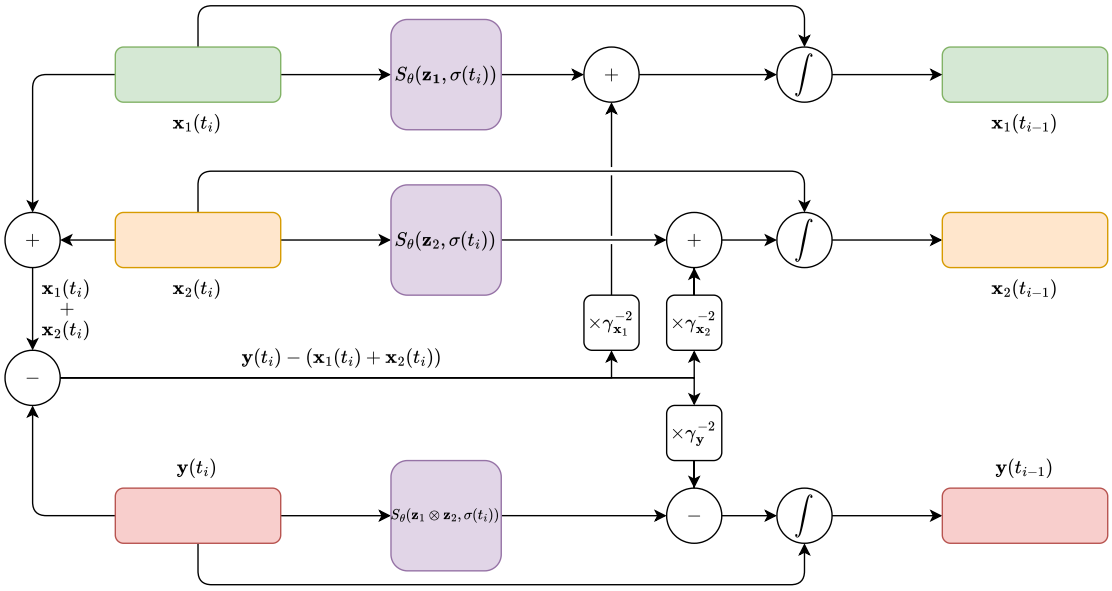

Figure 5.1: Diagram for unconditional generation procedure with GMSDI, sampling two coherent sources.

In this chapter we develop a novel inference procedure for the task, called *Generalized Multi-Source Diffusion Inference (GMSDI)*, that can be used in combination with *any* text-conditioned (time-domain) diffusion model for music. Such a method: *(i)* Requires only mixture data for training, resulting in an unsupervised[1] algorithm when paired with a contrastive encoder. *(ii)* Parameterizes an arbitrary number and type of sources. *(iii)* Allows for rich semantic control. To our knowledge, this is the first general algorithm for unsupervised compositional music generation. After developing the required background notions in Section 5.1, we develop the inference techniques in Section 5.2. We detail the experimental setup in Section 5.3 and show empirical results in Section 5.4. We conclude the chapter in Section 5.5.

5.1 Background

5.1.1 Text Embeddings

While we typically do not have direct access to the audio constituents $\{\mathbf{x}_n\}_{n\in[N']}$ (see Eq. 2.1) in a general mixture $\mathbf{y}$, we are usually equipped with text embeddings $\mathbf{z}_n$ which provide information about sources (weakly-supervised setting). We can obtain $\mathbf{z} = E_\phi^{\text{text}}(\mathbf{q})$ by encoding a text description $\mathbf{q}$ with a text-only encoder E_ϕ^{text}, or use a pre-trained contrastive [149, 150, 193] audio-text encoder E_ϕ^{contr} to extract embeddings both from the audio mixtures $\mathbf{z} = E_\phi^{\text{contr}}(\mathbf{y})$ and from text descriptions $\mathbf{z} = E_\phi^{\text{contr}}(\mathbf{q})$. Notice that, differently from Chapter 3, where we had only descriptions of sources in the form of their labels, here we can describe more generally mixtures and their constituent sources, also via semantic attributes (e.g., a guitar playing slowly together with a fast drum track).

[1] See the end of Section 5.1.2 for the precise meaning of the term unsupervised used in this chapter.

5.1.2 Text-conditioned Score-based Diffusion Models

We work with continuous-time score-based [23] diffusion models like in the previous chapter. A text-conditioned score-based diffusion model S_θ parameterizes the logarithm of the perturbed audio mixture density (see Eq. (4.2)), conditioned on the textual embedding:

$$\nabla_{\mathbf{y}(t)} \log p(\mathbf{y}(t) \mid \mathbf{z}) \approx S_\theta(\mathbf{y}(t), \mathbf{z}, \sigma(t)) \,. \tag{5.1}$$

At inference time, differently from the previous chapter, we use classifier-free guidance [25], integrating

$$S^*_\theta(\mathbf{y}(t), \mathbf{z}, \sigma(t)) = S_\theta(\mathbf{y}(t), \mathbf{z}, \sigma(t)) + w(S_\theta(\mathbf{y}(t), \mathbf{z}, \sigma(t)) - S_\theta(\mathbf{y}(t), \mathbf{z}^*, \sigma(t))) \,,$$

where $\mathbf{z}^*$ is a fixed learned embedding modeling the unconditional $\nabla_{\mathbf{y}(t)} \log p(\mathbf{y}(t))$, and $w \in \mathbb{R}$ is the embedding scale hyper-parameter. We can use a *negative embedding* [201] instead of $\mathbf{z}^*$ to better guide inference, by providing a direction that has to be avoided during sampling. With an abuse of notation, we will refer to S^*_θ as S_θ.

In this chapter, we give (yet a) different meaning to the word unsupervised, when the embeddings $\mathbf{z}$ are provided to the diffusion model by a pre-trained contrastive embedder E^{contr}_ϕ. In such a case, during the training of the generative model, it is not necessary to use textual data for training: we input the mixture $\mathbf{y}$ to E^{contr}_ϕ and condition the diffusion model with $E^{\text{contr}}_\phi(\mathbf{y})$. This is the approach taken, for example, by [8] using the MuLan [150] contrastive embedder and by us with the model trained on MTG-Jamendo (see Section 5.4) using CLAP [193] embeddings. At inference time, we use the embedders in the same way to parameterize the sources. In such a way, we can say that the training of the model is unsupervised, because it does not depend on external conditioning information during training. Such a terminology is also used in other works in literature: [202] uses a pre-trained tagger to steer the results during inference calling the method unsupervised. In reality, the contrastive encoder has been trained on tuples of the form $(\mathbf{y}, \mathbf{z})$ and we are implicitly in a weakly-supervised data setting. We can see this as a parametric pre-partitioning of the data during the training of the generative model, an evolved version of the manual partitioning performed in Chapter 3 using labels. While we are not in a proper unsupervised learning setting, given that we encapsulate the labeling information in an external (black-box) model we refer to this setting as unsupervised learning in the remaining of the thesis.

5.2 Generalized Multi-Source Diffusion Inference

We train (or use) a text-conditioned diffusion model (Eq. (5.1)) $S_\theta(\mathbf{y}(t), \mathbf{z}, \sigma(t))$, with audio mixtures $\mathbf{y}(t)$ and associated text embeddings $\mathbf{z}$, containing information about the sources present in the mixture. We assume that each text embedding $\mathbf{z}$ is of the form $\mathbf{z}_1 \otimes \cdots \otimes \mathbf{z}_N$ (more compactly $\bigotimes_{n=1}^N \mathbf{z}_n$), where each $\mathbf{z}_n$ describes a source $\mathbf{x}_n$ present in $\mathbf{y}$ and $\otimes$ denotes an encoding of concatenated textual information (e.g., $\mathbf{z}_1 \otimes \cdots \otimes \mathbf{z}_N = E^{\text{text}}_\phi(\mathbf{q}_1, \ldots, \mathbf{q}_N)$, with $E^{\text{text}}_\phi(\mathbf{q}_n) = \mathbf{z}_n$). The idea is to leverage such text embeddings for parameterizing the individual source score functions:

$$\nabla_{\mathbf{x}_n(t)} \log p(\mathbf{x}_n(t) \mid \mathbf{z}_n) \approx S_\theta(\mathbf{x}_n(t), \mathbf{z}_n, \sigma(t)) \,, \tag{5.2}$$

even if the model is trained only on mixtures. We devise a set of inference procedures for S_θ, called *Generalized Multi-Source Diffusion Inference*, able to solve the tasks of S_θ^{MSDM} (see Eq. (4.5)) in the relaxed data setting.

5.2.1 Total generation

In order to generate a coherent set of sources $\{\mathbf{x}_n\}_{n\in[N]}$, described by text embeddings $\{\mathbf{z}_n\}_{n\in[N]}$, we can sample from the conditionals $p(\mathbf{x}_n(t) \mid \mathbf{x}_{\bar{n}}(t), \mathbf{y}(t), \mathbf{z}_1, \dots, \mathbf{z}_N, \mathbf{z}_1 \otimes \cdots \otimes \mathbf{z}_N)$:

$$\frac{p(\mathbf{x}(t), \mathbf{y}(t) \mid \mathbf{z}_1, \dots, \mathbf{z}_N, \mathbf{z}_1 \otimes \cdots \otimes \mathbf{z}_N)}{p(\mathbf{x}_{\bar{n}}(t), \mathbf{y}(t) \mid \mathbf{z}_{\bar{n}}, \mathbf{z}_1 \otimes \cdots \otimes \mathbf{z}_N)}, \tag{5.3}$$

where $\mathbf{x}_{\bar{n}} = \{\mathbf{x}_m\}_{m\in[N]} - \{\mathbf{x}_n\}$ is the the complementary set of stems, fixing a source $\mathbf{x}_n$. First, we develop the numerator in Eq. (5.3) using the chain rule:

$$\begin{aligned}
& p(\mathbf{x}(t), \mathbf{y}(t) \mid \mathbf{z}_1, \dots, \mathbf{z}_N, \mathbf{z}_1 \otimes \cdots \otimes \mathbf{z}_N) \\
= \;& p(\mathbf{x}_n(t) \mid \mathbf{z}_n) p(\mathbf{y}(t), \mathbf{x}_{\bar{n}}(t) \mid \mathbf{x}_n(t), \mathbf{z}_{\bar{n}}, \mathbf{z}_1 \otimes \cdots \otimes \mathbf{z}_N) \\
= \;& p(\mathbf{x}_n(t) \mid \mathbf{z}_n) p(\mathbf{y}(t) \mid \mathbf{x}(t)) p(\mathbf{x}_{\bar{n}}(t) \mid \mathbf{x}_n(t), \mathbf{z}_{\bar{n}}) \\
\approx \;& p(\mathbf{x}_n(t) \mid \mathbf{z}_n) p(\mathbf{y}(t) \mid \mathbf{x}(t)) \,.
\end{aligned} \tag{5.4}$$

We assume independence of the likelihood $p(\mathbf{y}(t) \mid \mathbf{x}(t))$ from embeddings and approximate the last equality dropping the unknown term $p(\mathbf{x}_{\bar{n}}(t) \mid \mathbf{x}(t), \mathbf{z}_{\bar{n}})$. We substitute Eq. (5.4) in Eq. (5.3), take the gradient of the logarithm with respect to $\mathbf{x}_n(t)$ and model the likelihood with isotropic Gaussians [118] depending on a variance $\gamma^2_{\mathbf{x}_n}$ (the denominator cancels being constant w.r.t. $\mathbf{x}_n(t)$):

$$\begin{aligned}
& \nabla_{\mathbf{x}_n(t)} \frac{\log p(\mathbf{x}_n(t) \mid \mathbf{z}_n) p(\mathbf{y}(t) \mid \mathbf{x}(t))}{\log p(\mathbf{x}_{\bar{n}}(t), \mathbf{y}(t) \mid \mathbf{z}_{\bar{n}}, \mathbf{z}_1 \otimes \cdots \otimes \mathbf{z}_N)} \\
=& \nabla_{\mathbf{x}_n(t)} \log p(\mathbf{x}_n(t) \mid \mathbf{z}_n) + \nabla_{\mathbf{x}_n(t)} \log p(\mathbf{y}(t) \mid \mathbf{x}(t)) \\
=& \nabla_{\mathbf{x}_n(t)} \log p(\mathbf{x}_n(t) \mid \mathbf{z}_n) + \nabla_{\mathbf{x}_n(t)} \log \mathcal{N}(\mathbf{y}(t) \mid \sum_{m=1}^{N} \mathbf{x}_m(t), \gamma^2_{\mathbf{x}_n} \mathbf{I}) \\
=& \nabla_{\mathbf{x}_n(t)} \log p(\mathbf{x}_n(t) \mid \mathbf{z}_n) + \frac{1}{\gamma^2_{\mathbf{x}_n}} (\mathbf{y}(t) - \sum_{m=1}^{N} \mathbf{x}_m(t)) \,.
\end{aligned} \tag{5.5}$$

Applying similar steps we obtain the score of the density on $\mathbf{y}(t)$ conditioned on $\mathbf{x}(t)$ (notice the opposite likelihood gradient):

$$\begin{aligned}
& p(\mathbf{y}(t) \mid \mathbf{x}(t), \mathbf{z}_1, \dots, \mathbf{z}_N, \mathbf{z}_1 \otimes \cdots \otimes \mathbf{z}_N) \\
\approx \;& \nabla_{\mathbf{y}(t)} \log p(\mathbf{y}(t) \mid \bigotimes_{m=1}^{N} \mathbf{z}_m) + \frac{1}{\gamma^2_{\mathbf{y}}} (\sum_{m=1}^{N} \mathbf{x}_m(t) - \mathbf{y}(t)) \,.
\end{aligned} \tag{5.6}$$

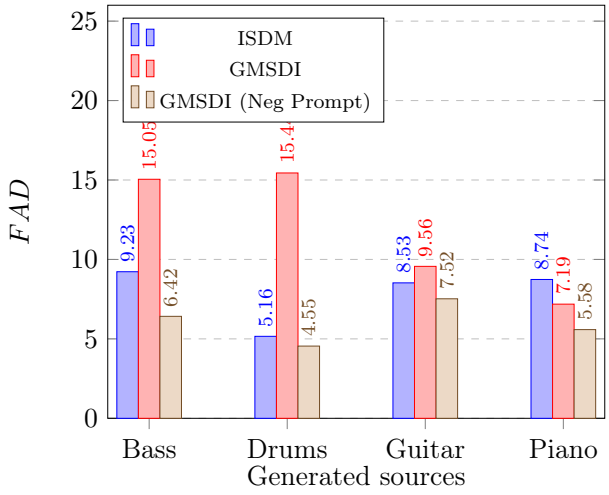

Figure 5.2: FAD (lower is better) between generated sources and Slakh100 test data (200 chunks, ∼12s each). Neg Prompt indicates the presence of negative prompting.

During inference, we sample from Eqs. (5.5) and (5.6) in *parallel*, replacing the gradients of the log-densities with score models (Eq. (5.2)):

$$\begin{cases} S_\theta(\mathbf{x}_n(t), \mathbf{z}_n, \sigma(t)) + \frac{1}{\gamma^2_{\mathbf{x}_n}}(\mathbf{y}(t) - \sum_{m=1}^N \mathbf{x}_m(t)) \\ S_\theta(\mathbf{y}(t), \bigotimes_{m=1}^N \mathbf{z}_m, \sigma(t)) + \frac{1}{\gamma^2_{\mathbf{y}}}(\sum_{m=1}^N \mathbf{x}_m(t) - \mathbf{y}(t)) \,. \end{cases} \tag{5.7}$$

A diagram of the method is illustrated in Figure 5.1. Given a partition $\{\mathcal{J}_m\}_{m\in[M]}$ of $[N]$ containing M subsets (i.e., $\cup_{m\in[M]}\mathcal{J}_m = [N]$), we can perform inference more generally with:

$$\begin{cases} S_\theta(\sum_{j\in\mathcal{J}_m} \mathbf{x}_j(t), \bigotimes_{j\in\mathcal{J}_m} \mathbf{z}_j, \sigma(t)) + \frac{1}{\gamma^2_{\mathcal{J}_m}}(\mathbf{y}(t) - \sum_{m=1}^N \mathbf{x}_m(t)) \\ S_\theta(\mathbf{y}(t), \bigotimes_{m=1}^N \mathbf{z}_m, \sigma(t)) + \frac{1}{\gamma^2_{\mathbf{y}}}(\sum_{m=1}^N \mathbf{x}_m(t) - \mathbf{y}(t)). \end{cases} \tag{5.8}$$

As we did in the previous chapter for source separation, it is also possible to model the likelihood function with Delta functions, parameterizing the mixture with the sources. Such a choice, however, resulted in unsatisfactory outcomes in our tests. We hypothesize that the conditioning with Dirac imposes too strong of a constraint, especially in the initial phase of the sampling procedure, and therefore leads to poor solutions.

5.2.2 Partial generation

We can generate accompaniments $\mathbf{x}_\mathcal{J}$ for a given set of sources $\mathbf{x}_\mathcal{I}$, described by $\{\mathbf{z}_i\}_{i\in\mathcal{I}}$, by selecting a set of accompaniment text embeddings $\{\mathbf{z}_j\}_{j\in\mathcal{J}}$. We integrate Eqs. (5.7) for $j \in \mathcal{J}$:

$$\begin{cases} S_\theta(\mathbf{x}_j(t), \mathbf{z}_j(t), \sigma(t)) + \frac{1}{\gamma^2_{\mathbf{x}_j}}\left[\mathbf{y}(t) - \left(\alpha \sum_{i\in\mathcal{I}} \mathbf{x}_i(t) + \beta \sum_{m\in\mathcal{J}} \mathbf{x}_m(t)\right)\right] \\ S_\theta(\mathbf{y}(t), \bigotimes_{m=1}^N \mathbf{z}_m, \sigma(t)) + \frac{1}{\gamma^2_{\mathbf{y}}}\left[\left(\alpha \sum_{i\in\mathcal{I}} \mathbf{x}_i(t) + \beta \sum_{m\in\mathcal{J}} \mathbf{x}_m(t)\right) - \mathbf{y}(t)\right] , \end{cases} \tag{5.9}$$

with $\mathbf{x}_i(t)$ $(i \in \mathcal{I})$ sampled from the perturbation kernel in Eq. (4.1) conditioned on $\mathbf{x}_i$ and $\alpha, \beta \in \mathbb{R}$ scaling factors. Using Eq. (5.8), we can generate the accompaniment mixtures $\sum_{j\in\mathcal{J}} \mathbf{x}_j$ directly.

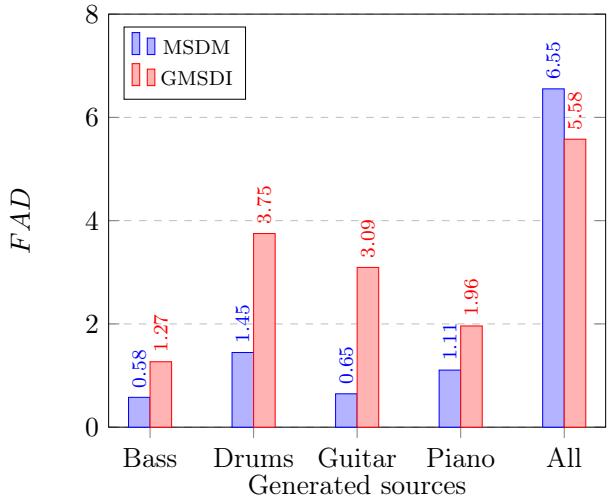

Figure 5.3: FAD (lower is better) results on total and partial generation, with respect to Slakh2100 test mixtures (200 chunks, ∼12s each). Results for MSDM in the partial setting are slightly different to those in Tables 4.3 because we enforce non-silent results with MSDM in this case (leading to slightly higher values of the FAD).

5.2.3 Source separation

Source separation can be performed by adapting Eq. (4.10) to the text-conditioned model. Let an observable mixture $\mathbf{y}(0)$ be composed by sources described by $\{\mathbf{z}_n\}_{n\in[N]}$. We can separate the sources by choosing a constrained source (w.l.o.g. the N-th) and sampling, for $n \in [N-1]$, with:

$$S_\theta(\mathbf{x}_n(t), \mathbf{z}_n, \sigma(t)) - S_\theta(\mathbf{y}(0) - \sum_{m=1}^{N-1} \mathbf{x}_m(t), \mathbf{z}_N, \sigma(t))\,. \tag{5.10}$$

We call this method *GMSDI Separator*. We also define a *GMSDI Extractor*, where we extract the n-th source $\mathbf{x}_n$ with:

$$S_\theta(\mathbf{x}_n(t), \mathbf{z}_n, \sigma(t)) - S_\theta(\mathbf{y}(0) - \mathbf{x}_n(t), \bigotimes_{m\neq n} \mathbf{z}_m, \sigma(t))\,, \tag{5.11}$$

constraining the mixture $\sum_{m\neq n} \mathbf{x}_m(t)$, complementary to $\mathbf{x}_n(t)$.

5.3 Experimental Setup

To validate our theoretical claims, we train two time-domain Moûsai-like [26] diffusion models. The first model is trained on Slakh2100 [119]. Slakh2100 is a dataset used in source separation, containing 2100 multi-source waveform music tracks obtained by synthesizing MIDI tracks with high-quality virtual instruments. We train the diffusion model on mixtures containing the stems Bass, Drums, Guitar, and Piano (the most abundant classes). To condition the diffusion model, we use the `t5-small` pre-trained T5 text-only encoder [197], which inputs the concatenation of the stem labels present in the mixture (e.g., "Bass, Drums" if the track contains Bass and Drums). Given that we know the labels describing the sources inside a mixture at training time, such an approach is weakly supervised. The window size is 2^{18} at 22kHz (∼12s).

The second model is trained on a more realistic dataset, namely MTG-Jamendo [203]. MTG-

Table 5.1: Grid search over embedding scale w on 100 chunks (∼12s each) of Slakh2100 test set. Results in SI-SDR$_\text{I}$ (dB – higher is better). The source in parenthesis is the constrained source.

Model	$w = 3.0$	$w = 7.5$	$w = 15.0$	$w = 24.0$
GMSDI Extractor	7.66	**9.61**	6.00	-0.62
GMSDI Separator (Bass)	8.10	6.72	-1.09	-20.60
GMSDI Separator (Drums)	**9.44**	8.69	-1.48	-21.62
GMSDI Separator (Guitar)	5.82	4.37	-2.27	-17.49
GMSDI Separator (Piano)	7.60	6.41	-2.68	-16.90

Jamendo is a music tagging dataset containing over 55000 musical mixtures and 195 tag categories. We train our diffusion model on the `raw_30s/audio-low` version of the dataset, using the first 98 shards for training and the last 2 for validation. The model window is of 2^{19} samples (∼24s) at 22kHz. We condition the model with the pre-trained `music_audioset_epoch_15_esc_90.14.pt`[2] checkpoint of the LAION CLAP contrastive encoder [198]. At training time, we condition the diffusion model with embeddings $E_\phi^{\text{contr}}(\mathbf{y})$ obtained from the training mixtures $\mathbf{y}$ themselves, resulting in an unsupervised model. At inference time, we use ADPM2[3] [204] with $\rho = 1$ for generation and AEuler[2] with $s_{\text{churn}} = 20$ for separation.

5.4 Experimental Results

First, we want to understand whether the model trained on Slakh2100 mixtures can parameterize single sources well. We sample, for each stem, 200 chunks of ∼12s, conditioning with embeddings of single stem labels (e.g., "Bass"). Then, we compute the FAD [190] with VGGish embeddings [191] between such samples and 200 random Slakh2100 test chunks of the same source. In Figure 5.2, we compare our model against the weakly supervised version of MSDM (ISDM), where a model learns the score function for each stem class (a setting requiring access to clean sources). We notice that single-stem prompting is insufficient for obtaining good FAD results, especially for Bass and Drums, causing silence to be generated. We find negative prompts (Section 5.1.2) essential for obtaining non-silent results using "Drums, Guitar, Piano" (Bass), "Bass" (Drums), "Bass, Drums" (Guitar), "Bass, Drums" (Piano). In all settings above, we use 150 sampling steps.

Following, we ask how well the model can perform coherent synthesis with GMSDI. In Figure 5.3, we compute the FAD between 200 random Slakh2100 test mixture chunks (∼ 12s each) and mixture chunks obtained by summing the model's generated stems (unconditional) or the generated stems together with the conditioning tracks (conditional). On total generation (All), we set $\gamma_{\mathbf{y}} = \infty$ and reach ∼ 1 lower FAD point, using 600 sampling steps. On partial generation, we sample using 300 steps, setting $\gamma_{\mathbf{y}} \ll \infty$, to inform the generated mixture about the conditioning sources. In this scenario, MSDM tends to generate silence. To enforce non-silent results with MSDM, we sample 100 examples for each conditioning chunk and select the sample with the highest L^2 norm.

For source separation, we employ the Scale-Invariant SDR Improvement (SI-SDR$_\text{I}$) [205] (as in Chapters 2 and 4) as an evaluation metric and follow the evaluation protocol of the sub-FAD presented in Section 4.5.1. First, we perform a grid search (Table 5.1) to find a good embedding scale

[2] `https://github.com/LAION-AI/CLAP`
[3] `https://github.com/crowsonkb/k-diffusion`

Table 5.2: Quantitative results for source separation on the Slakh2100 test set. Results in SI-SDR$_\text{I}$ (dB – higher is better).

Model	Bass	Drums	Guitar	Piano	All
Demucs + Gibbs (512 steps) [1]	17.16	19.61	17.82	16.32	**17.73**
ISDM	19.36	20.90	14.70	14.13	17.27
MSDM	17.12	18.68	15.38	14.73	16.48
GMSDI Separator	9.76	15.57	9.13	9.57	11.01
GMSDI Extractor	11.00	10.55	9.52	10.13	10.30
Ensamble	11.00	15.57	9.52	10.13	**11.56**

w. For the GMSDI Separator, we do not use negative prompting, while for the GMSDI Extractor, we only use negative prompts for Bass and Drums. We evaluate on the full Slakh2100 test set with $w = 3$ and constrained Drums for GMSDI Separator and $w = 7.5$ for GMSDI Extractor, showcasing results in Table 5.2. Training only with mixtures (plus associated labels), the ensemble of the two separators reaches 11.56 dB, being zero-shot, i.e., we do not target source separation during training [6].

We release qualitative examples for the Slakh2100 and MTG-Jamendo models on our demo page[4].

5.5 Summary and Prospects

We have proposed GMSDI, a compositional music generation method working with any time-domain text-guided diffusion model. Such a method, by exploiting text parameterization, performs source separation while generating, effectively enabling inter-source context processing in a (independent) Bayesian setting.

While the method obtains reasonable generation and separation metrics on Slakh2100, enabling (effective) unsupervised compositional music generation for the first time, it still lags with respect to the supervised MSDM. In future work, we want to extend the technique to latent diffusion models and narrow the gap with supervised methods.

[4] `https://github.com/gladia-research-group/gmsdi`

Chapter 6

Conclusion

In this thesis, we started in the context of universal sound separation (Chapter 2), where dependencies between sources are weak. We showed how to improve metrics over a standard regressor by adopting adversarial losses, typical of GANs. From this point, we started working in the setting of generative source separation, presenting LASS. This method performs (independent) Bayesian inference with autoregressive models in the latent domain of a VQ-VAE (Chapter 3). While performing experiments in the musical domain with this model, the independence assumption made it impossible to model the inter-dependencies between sources, which is typical of the musical domain. At this point (Chapter 4), we began focusing on the musical domain and proposed a Multi-Source Diffusion Model (MSDM) capable of generating the stems in a track, music accompaniments and source separation, all with a single model. Such a model is a source-joint generative model and introduces context into Bayesian inference. Finally, given the high data burden of the previous model, which is supervised, we asked ourselves (Chapter 5) if we could perform the same tasks in the independent (instance-based) Bayesian setting. We saw that such tasks could be solved by modeling both the individual sources and arbitrary combinations of the latter via parameterizations based on text information and performing separation while generating.

At this point, many questions arise about the possible ways to make the proposed models more flexible and high-performing. Although we have provided suggestions for new research in the concluding sections of the different chapters, here we want to look at future perspectives from a higher point of view, and give hints about the possible solutions.

6.1 Improved Guidance Techniques for Diffusion Models

The (Bayesian) inference methods for diffusion models in MSDM and for all tasks in GMSDI are local in time t, because the likelihood is dependent from the values $\mathbf{y}(t)$ and $\mathbf{x}(t)$, as shown in Figure 6.1 on the left. We can call this type of inference *local guidance*. Local guidance requires the likelihood to be compatible with the noise at level t [23]. Both in MSDM and GMSDI, we are compatible to such a requirement, because the likelihood is the probabilistic version of the analytical sum function (see Section B.1).

In many cases, especially when the likelihood term depends on learned auxiliary models, trained only on clean data, we are not compatible with noisy samples $\mathbf{x}(t)$ at time t. This situation also appears when we want to do inference with a latent diffusion model [26, 186, 206]: since we cannot

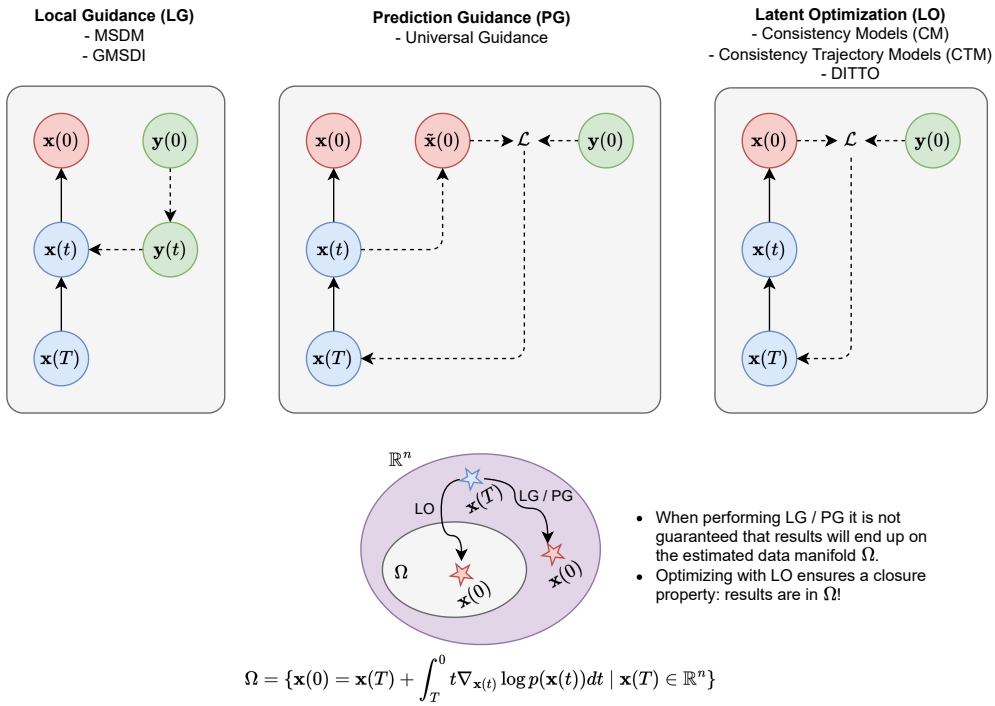

Figure 6.1: An overview of different diffusion-based inference techniques. *Left*: Local Guidance. *Middle*: Prediction Guidance. *Right*: Latent Optimization

sum latent vectors (in an analog manner as we could not sum them in the VQ-VAE domain in Chapter 3), we need to transport them first in the audio domain via the latent decoder, compute the likelihood term as a gradient of a loss $\mathcal{L}$ in audio domain and transport the gradients in latent space via back-propagation. We can perform *prediction guidance* in such a case (Figure 6.1, middle), which is also called *universal gudiance* [135] given its compatibility to any kind of guidance loss $\mathcal{L}$. Prediction guidance, at each step t, predicts the final value $\mathbf{x}(0)$ given $\mathbf{x}(t)$ as $\tilde{\mathbf{x}}(0) = D_\theta(\mathbf{x}(t), \sigma(t))$ (see Eq. 4.4), the loss is computed as $\mathcal{L}(f(\tilde{\mathbf{x}}(t)), \mathbf{y}(0))$, with f typically the sum (in GMSDI we would have also a predicted $\mathbf{y}(0) = \tilde{\mathbf{y}}(0) = D_\theta(\mathbf{y}(t), \sigma(t))$). Then we do guidance at step t with the gradient of such loss.

When performing local or prediction guidance it is not guaranteed that results will converge to the support of the prior density Ω (see Figure 6.1). This is evident when we perform source separation: the sum consistency constraint imposed by a sum likelihood can be so strong that the resulting mix completely resambles the observable mixture, but each stem contains artifacts of the other. Of course we can adjust the weighting of such a term, but finding an ideal value can be a burden (as exemplified by Tables 4.4 and 5.1). We can impose however that our solutions lie in the probabilistic manifold as close as possible by performing *latent optimization* (Figure 6.1, right). The idea is to depart from a fully posterior-based (Bayesian) approach and to optimize directly the latents of the diffusion process (the noise values $\mathbf{x}(T)$ at time T; not to be confused with latents $\boldsymbol{\xi}$ in the sense of autoencoding: the latents of the diffusion model are defined even when we train the generative model directly in the original data space), similar to the optimization techniques in Generative Adversarial Networks (e.g., [88]). Starting from $\mathbf{x}(T)$, we perform a full inference run and obtain $\mathbf{x}(0)$. Then we compute the loss $\mathcal{L}(f(\tilde{\mathbf{x}}(t)), \mathbf{y}(0))$, as with prediction guidance, but this time with the effective final sample. The gradients now backpropagate through the whole diffusion

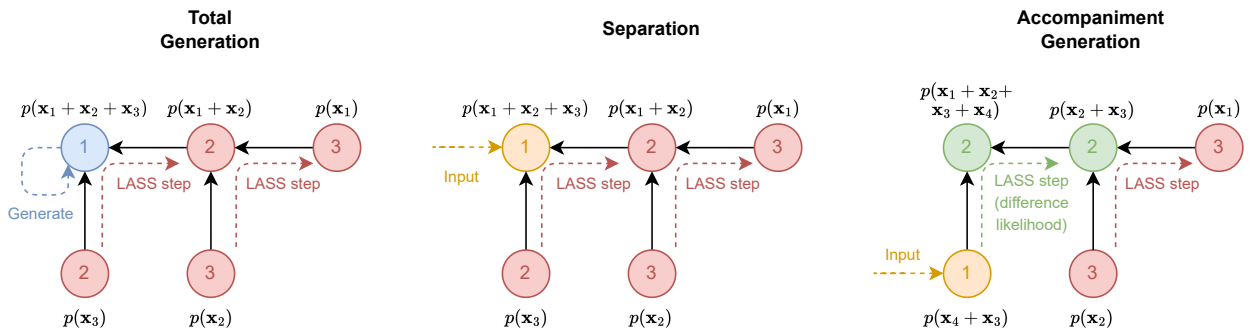

Figure 6.2: Multi-source inverence with autoregressive models. Black arrows define the graphical model while dotted arrows define the inference procedure. The numbers index the steps we perform during inference. *Left:* Total Generation. *Midle:* Source Separation. *Right:* Partial Generation.

sampling process and are applied to $\mathbf{x}(T)$. If our optimization variables $\mathbf{x}(T)$ lie inside the diffusion prior $\mathcal{N}(\mathbf{x}(t); \mathbf{0}, \sigma^2(T)\mathbf{I})$, the solutions should lie inside Ω. At the same time, notice that with the previous approaches, it is more difficult to perform correction during inference, given the greedy nature of local and prediction guidance.

The main difficulty of latent optimization is the high memory complexity induced by transporting gradients via a multi-step inference process (a situation similar to backpropagation in recurrent neural networks [207, 208]). We can solve this in two ways. Firstly, we can distill the diffusion model via modern distillation techniques such as Consistency Models [209, 210], Consistency Trajectory Models [211] or Adversarial Distillation [212] obtaining a version of the diffusion model that can perform sampling in one-step (or a few more for increased quality), similar to GANs. In such a way, inference reduces to a single evaluation of the network and we can back-propagate easily. A second interesting approach called DITTO (Diffusion Inference-Time T-Optimization) [136], by checkpointing the gradients trough sampling, enables latent optimization without modifying the diffusion model. While the authors apply the method for impressive control over the global structure of the generation (intensity, melody and structure controls) it is still to be applied for inter-stem generation [213].

6.2 Multi-Source Latent Autoregressive Inference

As pointed out in the conclusion of Chapter 3, an extension to more than two sources of LASS is to be sought, ideally working with residual quantization and being able to include the tasks of total and partial generation of stems. When we compare the continuous likelihoods of Chapters 4 and 5 (e.g., Gaussian, Dirac) with the discrete likelihood tensor in Eq. (3.6) we see that the main difference is the fact that the first can be easily applied to any number of addends, while the second has a fixed two input structure, and as such it must be applied in a recursive manner.

One way to proceed is to reason over the *graphical model* of the sum relationship at inference step s (after linearizing the residual layers). Such a graphical model is represented in Figure 6.2 for the different inference tasks. Take for example the graph on the left: the leaf nodes represent the single stems, with the related autoregressive densities (here we only distinguish stems by indices but they have to be parameterized via text like in Chapter 5). The sum relationship is represented from left to right were at each inner node we have the conditional probabilities specified by the likelihood

tensor in Eq. 3.6. To perform total generation, we first generate the total sum, then perform LASS steps, moving left-to-right, until we have generated all stems (this is similar to setting $\gamma_{\mathbf{y}} = \infty$ in GMSDI). Source separation is similar except that the mixture token is provided as an input, as in Chapter 3. Finally, to perform partial generation, we give as input the mixture containing the conditioning tracks, use a LASS step with a *difference* likelihood function (we can slice the tensor in Eq. 3.6 across the addend dimensions) and continue with normal LASS steps.

The graphical model employed is a Bayesian network [214], with extra information given by the prior probabilities defined on the inner nodes (in a Bayesian network, marginal probabilities are defined only on the leaf nodes). Considering the graphical model as a Bayesian network (neglecting prior information on inner nodes), we can also perform inference using the belief propagation algorithm [215]. Belief propagation integrates all information over the graph (passing the probabilities both from left-to-right and right-to-left), estimating conditional densities on the leaf nodes, from which one can sample. The downside of such algorithm is the high time-complexity cost required to perform tensor operations during propagation. The algorithms presented in Figure 6.2 are greedy in nature, instead, because we do not propagate probabilities across the whole graph during inference, sampling sequentially from left-to-right. Nonetheless, in this case we have the extra burden of sampling and tracking the intermediate mixtures. Future research will establish which is the best choice between the greedy method and belief propagation.

6.3 Training-Side Methods

In Chapter 5, we returned from the training-side setting of MSDM to the purely inference-side setting of GMSDI, justified by the lack of supervised data. Furthermore, in the two preceding sections, we have provided insights on how to improve inference techniques both in diffusion models and with autoregressive models. As seen from Figure 5.3 and Table 5.2, using GMSDI alone, it is difficult to achieve the results of MSDM, despite being competitive given the weakly-supervised setting. This tells us that techniques like GMSDI, LASS and its multi-source extension should be seen more as regularizers than as baselines, which can enhance the latter. In short, baselines should be obtained on the training side. So the question we ask ourselves is how to best utilize the limited supervised data on the training side? As mentioned in Section 3.1.1, fine-tuning techniques like ControlNet [128] for diffusion models and LoRA [127] for autoregressive models, allow to control a generative model trained on a large pool of data, by only fine-tuning with a relatively small dataset. An example in the musical domain is MusicControlNet [15]. Through this fine-tuning over datasets like MUSDB18-HQ [134] or MoisesDB [216], we can obtain models that generalize MSDM: generative models conditioned by mixtures of stems like StemGen [217], which can be used in a *sequential* manner or unconditional generative models, more similar to MSDM, for which the ControlNet adapter only processes the internal states of the latter, acting like a context processor and making it possible to use the generative model in parallel. Such techniques should be compatible with the presented inference techniques, so as to obtain higher performance.

Bibliography

[1] 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.

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

[3] Chang-Bin Jeon, Hyeongi Moon, Keunwoo Choi, Ben Sangbae Chon, and Kyogu Lee. Medleyvox: An evaluation dataset for multiple singing voices separation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[4] OpenAI. Gpt-4 technical report, 2023.

[5] Tim Brooks, Bill Peebles, Connor Homes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. URL `https://openai.com/research/video-generation-models-as-world-simulators`.

[6] Jordi Pons, Xiaoyu Liu, Santiago Pascual, and Joan Serrà. Gass: Generalizing audio source separation with large-scale data. *arXiv preprint arXiv:2310.00140*, 2023.

[7] Yizhi Li, Ruibin Yuan, Ge Zhang, Yinghao Ma, Xingran Chen, Hanzhi Yin, Chenghua Lin, Anton Ragni, Emmanouil Benetos, Norbert Gyenge, et al. Mert: Acoustic music understanding model with large-scale self-supervised training. *arXiv preprint arXiv:2306.00107*, 2023.

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

[9] Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation. *Advances in Neural Information Processing Systems*, 36, 2024.

[10] Haohe Liu, Qiao Tian, Yiitan Yuan, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Yuping Wang, Wenwu Wang, Yuxuan Wang, and MarkD . Plumbley. Audioldm 2: Learning holistic audio generation with self-supervised pretraining. *ArXiv*, abs/2308.05734, 2023. URL `https://api.semanticscholar.org/CorpusID:260775781`.

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

[12] Ian Clester and Jason Freeman. Composing with generative systems in the digital audio workstation 145-148. In Alison Smith-Renner and Paul Taele, editors, *Joint Proceedings of the IUI 2023 Workshops: HAI-GEN, ITAH, MILC, SHAI, SketchRec, SOCIALIZE co-located with the ACM International Conference on Intelligent User Interfaces (IUI 2023), Sydney, Australia, March 27-31, 2023*, volume 3359 of *CEUR Workshop Proceedings*, pages 145–148. CEUR-WS.org, 2023. URL `https://ceur-ws.org/Vol-3359/paper15.pdf`.

[13] Hounsu Kim, Soonbeom Choi, and Juhan Nam. Expressive acoustic guitar sound synthesis with an instrument-specific input representation and diffusion outpainting. *arXiv preprint arXiv:2401.13498*, 2024.

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

[15] Shih-Lun Wu, Chris Donahue, Shinji Watanabe, and Nicholas J Bryan. Music controlnet: Multiple time-varying controls for music generation. *arXiv preprint arXiv:2311.07069*, 2023.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[17] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 44(11):7327–7347, nov 2022. ISSN 1939-3539. doi: 10.1109/TPAMI.2021.3116668.

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Proc. NIPS*, 27, 2014.

[19] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Proc. NeurIPS*, 30, 2017.

[21] 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.

[22] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pages 11895–11907, 2019.

[23] 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.

[24] 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.

[25] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL `https://openreview.net/forum?id=qw8AKxfYbI`.

[26] Flavio Schneider, Zhijing Jin, and Bernhard Schölkopf. Moûsai: Text-to-music generation with long-context latent diffusion. *arXiv preprint arXiv:2301.11757*, 2023.

[27] 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.

[28] Emilian Postolache, Giorgio Mariani, Michele Mancusi, Andrea Santilli, Luca Cosmo, and Emanuele Rodolà. Latent autoregressive source separation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(8):9444–9452, Jun. 2023. doi: 10.1609/aaai.v37i8.26131. URL `https://ojs.aaai.org/index.php/AAAI/article/view/26131`.

[29] Giorgio Mariani, Irene Tallini, Emilian Postolache, Michele Mancusi, Luca Cosmo, and Emanuele Rodolà. Multi-source diffusion models for simultaneous music generation and separation. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=h922Qhkmx1`.

[30] Chin-Yun Yu, Emilian Postolache, Emanuele Rodolà, and Gyorgy Fazekas. Zero-shot duet singing voices separation with diffusion models. *ArXiv*, abs/2311.07345, 2023. URL `https://api.semanticscholar.org/CorpusID:265150315`.

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

[32] Yu Nakagome, Masahito Togami, Tetsuji Ogawa, and Tetsunori Kobayashi. Efficient and stable adversarial learning using unpaired data for unsupervised multichannel speech separation. In *Interspeech*, pages 3051–3055, 2021.

[33] Kohei Saijo and Tetsuji Ogawa. Remix-cycle-consistent learning on adversarially learned separator for accurate and stable unsupervised speech separation. In *ICASSP*, 2022.

[34] Daniel Arteaga and Jordi Pons. Multichannel-based learning for audio object extraction. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 206–210. IEEE, 2021.

[35] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Adversarial semi-supervised audio source separation applied to singing voice extraction. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2391–2395. IEEE, 2018.

[36] 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.

[37] Jordi Pons, Santiago Pascual, Giulio Cengarle, and Joan Serrà. Upsampling artifacts in neural audio synthesis. In *ICASSP*, 2020.

[38] Berkan Kadıoğlu, Michael Horgan, Xiaoyu Liu, Jordi Pons, Dan Darcy, and Vivek Kumar. An empirical study of Conv-TasNet. In *ICASSP*, 2020.

[39] Xiaoyu Liu and Jordi Pons. On permutation invariant training for speech source separation. In *ICASSP*, 2021.

[40] 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. doi: 10.21437/Interspeech.2020-3115.

[41] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[42] 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.

[43] Efthymios Tzinis, Scott Wisdom, John R Hershey, Aren Jansen, and Daniel PW Ellis. Improving universal sound separation using sound classification. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 96–100. IEEE, 2020.

[44] Scott Wisdom, Aren Jansen, Ron J Weiss, Hakan Erdogan, and John R Hershey. Sparse, efficient, and semantic mixture invariant training: Taming in-the-wild unsupervised sound separation. In *WASPAA*, 2021.

[45] Efthymios Tzinis, Zhepei Wang, Xilin Jiang, and Paris Smaragdis. Compute and memory efficient universal sound source separation. *Journal of Signal Processing Systems*, 94(2):245–259, 2022.

[46] 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.

[47] 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.

[48] Chenxing Li, Lei Zhu, Shuang Xu, Peng Gao, and Bo Xu. CBLDNN-based speaker-independent speech separation via generative adversarial training. In *ICASSP*, 2018.

[49] Lianwu Chen, Meng Yu, Yanmin Qian, Dan Su, and Dong Yu. Permutation invariant training of generative adversarial network for monaural speech separation. In *Interspeech*, 2018.

[50] Ziqiang Shi, Huibin Lin, Liu Liu, Rujie Liu, Shoji Hayakawa, and Jiqing Han. Furcax: End-to-end monaural speech separation based on deep gated (de) convolutional neural networks with adversarial example training. In *ICASSP*, 2019.

[51] Chengyun Deng, Yi Zhang, Shiqian Ma, Yongtao Sha, Hui Song, and Xiangang Li. Conv-TasSAN: Separative adversarial network based on conv-tasnet. In *Interspeech*, pages 2647–2651, 2020.

[52] Daniel PW Ellis. *Prediction-driven computational auditory scene analysis*. PhD thesis, Columbia University, 1996.

[53] Albert S Bregman. *Auditory scene analysis: The perceptual organization of sound*. MIT press, 1994.

[54] Guy J Brown and Martin Cooke. Computational auditory scene analysis. *Computer Speech & Language*, 8(4):297–336, 1994.

[55] Martin Cooke. *Modelling auditory processing and organisation*, volume 7. Cambridge University Press, 2005.

[56] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005.

[57] M.D. Plumbley. Conditions for nonnegative independent component analysis. *IEEE Signal Processing Letters*, 9(6):177–180, 2002. doi: 10.1109/LSP.2002.800502.

[58] M.D. Plumbley. Algorithms for nonnegative independent component analysis. *IEEE Transactions on Neural Networks*, 14(3):534–543, 2003. doi: 10.1109/TNN.2003.810616.

[59] Thomas Blumensath and M Davies. Unsupervised learning of sparse and shift-invariant decompositions of polyphonic music. In *ICASSP*, 2004.

[60] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[61] Michael S. Lewicki and Terrence J. Sejnowski. Learning Overcomplete Representations. *Neural Computation*, 12(2):337–365, 02 2000. ISSN 0899-7667. doi: 10.1162/089976600300015826.

[62] Anastasia Podosinnikova, Amelia Perry, Alexander S Wein, Francis Bach, Alexandre d'Aspremont, and David Sontag. Overcomplete independent component analysis via sdp. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2583–2592. PMLR, 2019.

[63] YW Teh, Max Welling, S Osindero, and GE Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4:1235–1260, 10 2004.

[64] Mikkel N Schmidt and Rasmus Kongsgaard Olsson. Single-channel speech separation using sparse non-negative matrix factorization. In *Interspeech*, 2006.

[65] Paris Smaragdis and Judith C Brown. Non-negative matrix factorization for polyphonic music transcription. In *WASPAA*, 2003.

[66] Cédric Févotte, Nancy Bertin, and Jean-Louis Durrieu. Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis. *Neural computation*, 21(3): 793–830, 2009.

[67] Toni Heittola, Annamaria Mesaros, Tuomas Virtanen, and Antti Eronen. Sound event detection in multisource environments using source separation. In *Machine Listening in Multisource Environments*, 2011.

[68] Jort F. Gemmeke, Lode Vuegen, Peter Karsmakers, Bart Vanrumste, and Hugo Van hamme. An exemplar-based nmf approach to audio event detection. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4, 2013. doi: 10.1109/WASPAA.2013.6701847.

[69] Yuki Mitsufuji, Marco Liuni, Alex Baker, and Axel Roebel. Online non-negative tensor deconvolution for source detection in 3dtv audio. In *ICASSP*, 2014.

[70] Kwang Myung Jeon and Hong Kook Kim. Dual-channel acoustic event detection in multisource environments using nonnegative tensor factorization and hidden markov model. *Journal of the Institute of Electronics and Information Engineers*, 54(1):121–128, 2017.

[71] Sam Roweis. One microphone source separation. In *Advances in Neural Information Processing Systems*, volume 13, 2000.

[72] Po-Sen Huang, Scott Deeann Chen, Paris Smaragdis, and Mark Hasegawa-Johnson. Singing-voice separation from monaural recordings using robust principal component analysis. In *Proc. ICASSP*, pages 57–60. IEEE, 2012.

[73] 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)*, pages 2135–2139, 2015. doi: 10.1109/ICASSP.2015.7178348.

[74] Aditya Arie Nugraha, Antoine Liutkus, and Emmanuel Vincent. Multichannel audio source separation with deep neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(9):1652–1664, 2016. doi: 10.1109/TASLP.2016.2580946.

[75] Jen-Yu Liu and Yi-Hsuan Yang. Denoising auto-encoder with recurrent skip connections and residual regression for music source separation. *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 773–778, 2018.

[76] 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. doi: 10.1109/IWAENC.2018.8521383.

[77] 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.

[78] 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.

[79] Nicolás Schmidt, Jordi Pons, and Marius Miron. Podcastmix: A dataset for separating music and speech in podcasts. *Interspeech*, 2022.

[80] Dong Yu, Morten Kolbæk, Zheng-Hua Tan, and Jesper Jensen. Permutation invariant training of deep models for speaker-independent multi-talker speech separation. In *ICASSP*, 2017.

[81] Scott Wisdom, Hakan Erdogan, Daniel PW Ellis, Romain Serizel, Nicolas Turpault, Eduardo Fonseca, Justin Salamon, Prem Seetharaman, and John R Hershey. What's all the fuss about free universal sound separation data? In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 186–190. IEEE, 2021.

[82] Jürgen Schmidhuber. Generative adversarial networks are special cases of artificial curiosity (1990) and also closely related to predictability minimization (1991). *Neural Networks*, 127: 58–66, 2020.

[83] Santiago Pascual, Antonio Bonafonte, and Joan Serrà. Segan: Speech enhancement generative adversarial network. In *Interspeech*, 2017. URL `https://api.semanticscholar.org/CorpusID:12054873`.

[84] Jae Hyun Lim and Jong Chul Ye. Geometric GAN. *arXiv preprint arXiv:1705.02894*, 2016.

[85] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[86] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[87] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

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

[89] Mikołaj Bińkowski, Jeff Donahue, Sander Dieleman, Aidan Clark, Erich Elsen, Norman Casagrande, Luis C Cobo, and Karen Simonyan. High fidelity speech synthesis with adversarial networks. *arXiv*, 2019.

[90] Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron C Courville. MelGAN: Generative adversarial networks for conditional waveform synthesis. *NeurIPS*, 2019.

[91] Eduardo Fonseca, Jordi Pons Puig, Xavier Favory, Frederic Font Corbera, Dmitry Bogdanov, Andres Ferraro, Sergio Oramas, Alastair Porter, and Xavier Serra. Freesound datasets: a platform for the creation of open audio datasets. In *ISMIR*, 2017.

[92] Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, and Xavier Serra. FSD50k: an open dataset of human-labeled sound events. *IEEE/ACM TASLP*, 30:829–852, 2021.

[93] DCASE2020 challenge: Sound event detection and separation in domestic environments. https://dcase.community/challenge2020/task-sound-event-detection-and-separation-in-domestic-environments, 2020.

[94] Andreas Jansson, Eric Humphrey, Nicola Montecchio, Rachel Bittner, Aparna Kumar, and Tillman Weyde. Singing voice separation with deep U-net convolutional networks. *ISMIR*, 2017.

[95] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[96] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980. URL `https://api.semanticscholar.org/CorpusID:206775608`.

[97] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[98] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[99] Scott Wisdom, John R. Hershey, Kevin Wilson, Jeremy Thorpe, Michael Chinen, Brian Patton, and Rif A. Saurous. Differentiable consistency constraints for improved deep speech enhancement. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 900–904, 2019. doi: 10.1109/ICASSP.2019.8682783.

[100] Kai Li, Runxuan Yang, and Xiaolin Hu. An efficient encoder-decoder architecture with top-down attention for speech separation. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=fzberKYWKsI`.

[101] Yi Luo and Jianwei Yu. Music source separation with band-split rnn. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, PP:1–10, 01 2023. doi: 10.1109/TASLP.2023.3271145.

[102] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proc. ICLR*, 2014. URL `http://arxiv.org/abs/1312.6114`.

[103] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Proc. NeurIPS*, 33:1877–1901, 2020.

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

[105] Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In *Proc. NeurIPS*, 2019.

[106] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[107] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *Proc. ICML*, pages 8821–8831. PMLR, 2021.

[108] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

[109] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

[110] Rodrigo Castellon, Chris Donahue, and Percy Liang. Codified audio language modeling learns useful representations for music information retrieval. *arXiv preprint arXiv:2107.05677*, 2021.

[111] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=gEZrGCozdqR`.

[112] Victor Sanh, Albert Webson, Colin Raffel, et al. Multitask prompted training enables zero-shot task generalization. In *Proc. ICLR*, 2022. URL `https://openreview.net/forum?id=9Vrb9D0WI4`.

[113] Hao Yang, Junyang Lin, An Yang, Peng Wang, Chang Zhou, and Hongxia Yang. Prompt tuning for generative multimodal pretrained models. *arXiv preprint arXiv:2208.02532*, 2022.

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

[115] Efthymios Tzinis, Shrikant Venkataramani, Zhepei Wang, Cem Subakan, and Paris Smaragdis. Two-step sound source separation: Training on learned latent targets. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 31–35. IEEE, 2020.

[116] Dimitrios Bralios, Efthymios Tzinis, Gordon Wichern, Paris Smaragdis, and Jonathan Le Roux. Latent iterative refinement for modular source separation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[117] T. Halperin, A. Ephrat, and Y. Hoshen. Neural separation of observed and unobserved distributions. *36th International Conference on Machine Learning, ICML 2019*, 2019-June: 4548–4557, 2019.

[118] Vivek Jayaram and John Thickstun. Source separation with deep generative priors. In *International Conference on Machine Learning*, pages 4724–4735. PMLR, 2020.

[119] 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)*, pages 45–49. IEEE, 2019.

[120] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pages 4114–4124. PMLR, 2019.

[121] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[122] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[123] 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.

[124] Bo Chen, Chao Wu, and Wenbin Zhao. Sepdiff: Speech separation based on denoising diffusion model. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10095979.

[125] Naoyuki Kamo, Marc Delcroix, and Tomohiro Nakatani. Target Speech Extraction with Conditional Diffusion Model. In *Proc. INTERSPEECH 2023*, pages 176–180, 2023. doi: 10.21437/Interspeech.2023-1234.

[126] Ryan McGrady, Kevin Zheng, Rebecca Curran, Jason Baumgartner, and Ethan Zuckerman. Dialing for videos: A random sample of youtube. *Journal of Quantitative Description: Digital Media*, 3, Dec. 2023. doi: 10.51685/jqd.2023.022. URL `https://journalqd.org/article/view/4066`.

[127] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=nZeVKeeFYf9`.

[128] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3813–3824, 2023. URL https://api.semanticscholar.org/CorpusID:256827727.

[129] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

[130] 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. ISBN 9780999241141.

[131] G. Parisi. Correlation functions and computer simulations. *Nuclear Physics B*, 180(3):378–384, 1981. ISSN 0550-3213. doi: https://doi.org/10.1016/0550-3213(81)90056-0. URL https://www.sciencedirect.com/science/article/pii/0550321381900560.

[132] Ge Zhu, Jordan Darefsky, Fei Jiang, Anton Selitskiy, and Zhiyao Duan. Music source separation with generative flow. *IEEE Signal Processing Letters*, 29:2288–2292, 2022. doi: 10.1109/LSP.2022.3219355.

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

[134] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. Musdb18-hq - an uncompressed version of musdb18, August 2019. URL https://doi.org/10.5281/zenodo.3338373.

[135] Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 843–852, 2023.

[136] Zachary Novack, Julian McAuley, Taylor Berg-Kirkpatrick, and Nicholas J. Bryan. Ditto: Diffusion inference-time t-optimization for music generation. *ArXiv*, abs/2401.12179, 2024. URL https://api.semanticscholar.org/CorpusID:267068663.

[137] Yixiao Zhang, Yukara Ikemiya, Gus G. Xia, Naoki Murata, Marco A. Martínez Ramírez, Wei-Hsiang Liao, Yuki Mitsufuji, and Simon Dixon. Musicmagus: Zero-shot text-to-music editing via diffusion models. *ArXiv*, abs/2402.06178, 2024. URL https://api.semanticscholar.org/CorpusID:267616810.

[138] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=AAWuCvzaVt.

[139] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Proc. NeurIPS*, 29, 2016.

[140] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[141] Wouter Kool, Herke van Hoof, and Max Welling. Ancestral gumbel-top-k sampling for sampling without replacement. *Journal of Machine Learning Research*, 21(47):1–36, 2020. URL `http://jmlr.org/papers/v21/19-985.html`.

[142] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *Proc. ICLR*, 2020. URL `https://openreview.net/forum?id=rygGQyrFvH`.

[143] D Raj Reddy et al. Speech understanding systems: A summary of results of the five-year research effort. *Department of Computer Science. Camegie-Mell University, Pittsburgh, PA*, 17:138, 1977.

[144] 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.

[145] Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Olivier Teboul, David Grangier, Marco Tagliasacchi, and Neil Zeghidour. Audiolm: a language modeling approach to audio generation. *arXiv preprint arXiv:2209.03143*, 2022.

[146] Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. Audiogen: Textually guided audio generation. *arXiv preprint arXiv:2209.15352*, 2022.

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

[148] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL `https://openreview.net/forum?id=ivCd8z8zR2`. Featured Certification, Reproducibility Certification.

[149] Ilaria Manco, Emmanouil Benetos, Elio Quinton, and György Fazekas. Learning music audio representations via weak language supervision. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 456–460. IEEE, 2022.

[150] Qingqing Huang, Aren Jansen, Joonseok Lee, Ravi Ganti, Judith Yue Li, and Daniel P. W. Ellis. Mulan: A joint embedding of music audio and natural language. In *International Society for Music Information Retrieval Conference*, 2022.

[151] Chris Donahue, Antoine Caillon, Adam Roberts, Ethan Manilow, Philippe Esling, Andrea Agostinelli, Mauro Verzetti, Ian Simon, Olivier Pietquin, Neil Zeghidour, et al. Singsong: Generating musical accompaniments from singing. *arXiv preprint arXiv:2301.12662*, 2023.

[152] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[153] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proc. ICCV*, December 2015.

[154] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *Proc. ICLR*, 2018. URL `https://openreview.net/forum?id=Hk99zCeAb`.

[155] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[156] Alain Horé and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *Proc. ICPR*, pages 2366–2369, 2010. doi: 10.1109/ICPR.2010.579.

[157] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6629–6640, 2017.

[158] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Score-based generative modeling with critically-damped langevin diffusion. *ArXiv*, abs/2112.07068, 2021.

[159] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

[160] Zafar Rafii and Bryan Pardo. Repeating pattern extraction technique (repet): A simple method for music/voice separation. *IEEE transactions on audio, speech, and language processing*, 21(1):73–84, 2012.

[161] Prem Seetharaman, Fatemeh Pishdadian, and Bryan Pardo. Music/voice separation using the 2d fourier transform. In *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 36–40. IEEE, 2017.

[162] Fabian-Robert Stöter, Antoine Liutkus, and Nobutaka Ito. The 2018 signal separation evaluation campaign. In *Proc. LVA/ICA*, pages 293–305, 2018.

[163] Naoya Takahashi, Sudarsanam Parthasaarathy, Nabarun Goswami, and Yuki Mitsufuji. Recursive Speech Separation for Unknown Number of Speakers. In *Proc. Interspeech 2019*, pages 1348–1352, 2019. doi: 10.21437/Interspeech.2019-1550.

[164] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved vqgan. *arXiv preprint arXiv:2110.04627*, 2021.

[165] Yilun Xu, Yang Song, Sahaj Garg, Linyuan Gong, Rui Shu, Aditya Grover, and Stefano Ermon. Anytime sampling for autoregressive models via ordered autoencoding. *arXiv preprint arXiv:2102.11495*, 2021.

[166] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17(1): 7184–7220, 2016.

[167] Shahar Lutati, Eliya Nachmani, and Lior Wolf. Separate and diffuse: Using a pretrained diffusion model for improving source separation. *arXiv preprint arXiv:2301.10752*, 2023.

[168] 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.

[169] 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.

[170] Durk P Kingma and Yann LeCun. Regularized estimation of image statistics by score matching. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

[171] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

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

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

[174] Yen-Ju Lu, Yu Tsao, and Shinji Watanabe. A study on speech enhancement based on diffusion probabilistic model. In *2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 659–666. IEEE, 2021.

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

[176] Ryosuke Sawata, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Takashi Shibuya, Shusuke Takahashi, and Yuki Mitsufuji. A versatile diffusion-based generative refiner for speech enhancement. *arXiv preprint arXiv:2210.17287*, 2022.

[177] Koichi Saito, Naoki Murata, Toshimitsu Uesaka, Chieh-Hsin Lai, Yuhta Takida, Takao Fukui, and Yuki Mitsufuji. Unsupervised vocal dereverberation with diffusion-based generative models. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.

[178] Junhyeok Lee and Seungu Han. NU-Wave: A Diffusion Probabilistic Model for Neural Audio Upsampling. In *Proc. Interspeech 2021*, pages 1634–1638, 2021. doi: 10.21437/Interspeech.2021-36.

[179] Chin-Yun Yu, Sung-Lin Yeh, György Fazekas, and Hao Tang. Conditioning and sampling in variational diffusion models for speech super-resolution. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.

[180] Gautam Mittal, Jesse Engel, Curtis Hawthorne, and Ian Simon. Symbolic music generation with diffusion models. In *Proceedings of the 22nd International Society for Music Information Retrieval Conference*, 2021.

[181] Curtis Hawthorne, Ian Simon, Adam Roberts, Neil Zeghidour, Josh Gardner, Ethan Manilow, and Jesse Engel. Multi-instrument music synthesis with spectrogram diffusion. In *International Society for Music Information Retrieval Conference*, 2022.

[182] Kin Wai Cheuk, Ryosuke Sawata, Toshimitsu Uesaka, Naoki Murata, Naoya Takahashi, Shusuke Takahashi, Dorien Herremans, and Yuki Mitsufuji. Diffroll: Diffusion-based generative music transcription with unsupervised pretraining capability. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[183] Simon Rouard and Gaëtan Hadjeres. CRASH: raw audio score-based generative modeling for controllable high-resolution drum sound synthesis. In *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021*, pages 579–585, 2021.

[184] Dongchao Yang, Jianwei Yu, Helin Wang, Wen Wang, Chao Weng, Yuexian Zou, and Dong Yu. Diffsound: Discrete diffusion model for text-to-sound generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[185] Santiago Pascual, Gautam Bhattacharya, Chunghsin Yeh, Jordi Pons, and Joan Serrà. Full-band general audio synthesis with score-based diffusion. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.

[186] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. AudioLDM: Text-to-audio generation with latent diffusion models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 21450–21474. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/liu23f.html`.

[187] Seth* Forsgren and Hayk* Martiros. Riffusion - Stable diffusion for real-time music generation, 2022. URL `https://riffusion.com/about`.

[188] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794. Curran Associates, Inc., 2021.

[189] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[190] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fréchet Audio Distance: A Reference-Free Metric for Evaluating Music Enhancement Algorithms. In *Proc. Interspeech 2019*, pages 2350–2354, 2019. doi: 10.21437/Interspeech.2019-2219.

[191] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron Weiss, and Kevin Wilson. Cnn architectures for large-scale audio classification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[192] Azalea Gui, Hannes Gamper, Sebastian Braun, and Dimitra Emmanouilidou. Adapting frechet audio distance for generative music evaluation. *arXiv preprint arXiv:2311.01616*, 2023.

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

[194] 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.

[195] Lichao Zhang, Ruiqi Li, Shoutong Wang, Liqun Deng, Jinglin Liu, Yi Ren, Jinzheng He, Rongjie Huang, Jieming Zhu, Xiao Chen, and Zhou Zhao. M4singer: A multi-style, multi-singer and musical score provided mandarin singing corpus. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL `https://openreview.net/forum?id=qiDmAaG6mP`.

[196] Yu Wang, Xinsheng Wang, Pengcheng Zhu, Jie Wu, Hanzhao Li, Heyang Xue, Yongmao Zhang, Lei Xie, and Mengxiao Bi. Opencpop: A high-quality open source chinese popular song corpus for singing voice synthesis. In *Interspeech*, 2022. URL `https://api.semanticscholar.org/CorpusID:246035478`.

[197] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

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

[199] Yuancheng Wang, Zeqian Ju, Xu Tan, Lei He, Zhizheng Wu, Jiang Bian, and sheng zhao. AUDIT: Audio editing by following instructions with latent diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=EO1KuHoR0V`.

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

[201] Guillaume Sanchez, Honglu Fan, Alexander Spangher, Elad Levi, Pawan Sasanka Ammanamanchi, and Stella Biderman. Stay on topic with classifier-free guidance. *arXiv preprint arXiv:2306.17806*, 2023.

[202] Ethan Manilow, Patrick O'Reilly, Prem Seetharaman, and Bryan Pardo. Unsupervised source separation by steering pretrained music models. *arXiv preprint arXiv:2110.13071*, 2021.

[203] Dmitry Bogdanov, Minz Won, Philip Tovstogan, Alastair Porter, and Xavier Serra. The mtg-jamendo dataset for automatic music tagging. In *International Conference on Machine Learning*, 2019. URL `https://api.semanticscholar.org/CorpusID:196187495`.

[204] 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. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

[205] 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. IEEE, 2019.

[206] Zach Evans, CJ Carr, Josiah Taylor, Scott H. Hawley, and Jordi Pons. Fast timing-conditioned latent audio diffusion. *ArXiv*, abs/2402.04825, 2024. URL `https://api.semanticscholar.org/CorpusID:267523339`.

[207] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.

[208] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

[209] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[210] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

[211] 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*, 2024. URL `https://openreview.net/forum?id=ymjI8feDTD`.

[212] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*, 2023.

[213] Yixiao Zhang, Yukara Ikemiya, Gus Xia, Naoki Murata, Marco Martínez, Wei-Hsiang Liao, Yuki Mitsufuji, and Simon Dixon. Musicmagus: Zero-shot text-to-music editing via diffusion models. *arXiv preprint arXiv:2402.06178*, 2024.

[214] J. Pearl. *Bayesian Networks: A Model of Self-activated Memory for Evidential Reasoning*. Report (University of California, Los Angeles. Computer Science Dept.). UCLA Computer Science Department, 1985. URL `https://books.google.it/books?id=1sfMOgAACAAJ`.

[215] Judea Pearl. Reverend bayes on inference engines: a distributed hierarchical approach. In *Proceedings of the Second AAAI Conference on Artificial Intelligence*, AAAI'82, page 133–136. AAAI Press, 1982.

[216] Igor G Pereira, Felipe Araujo, Filip Korzeniowski, and Richard Vogl. Moisesdb: A dataset for source separation beyond 4 stems. In *Ismir 2023 Hybrid Conference*, 2023.

[217] Julian D Parker, Janne Spijkervet, Katerina Kosta, Furkan Yesiler, Boris Kuznetsov, Ju-Chiang Wang, Matt Avent, Jitong Chen, and Duc Le. Stemgen: A music generation model that listens. *arXiv preprint arXiv:2312.08723*, 2023.

[218] A.W. Rix, J.G. Beerends, M.P. Hollier, and A.P. Hekstra. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, volume 2, pages 749–752 vol.2, 2001. doi: 10.1109/ICASSP.2001.941023.

[219] Cees H. Taal, Richard C. Hendriks, Richard Heusdens, and Jesper Jensen. An algorithm for intelligibility prediction of time–frequency weighted noisy speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2125–2136, 2011. doi: 10.1109/TASL.2011.2114881.

[220] Yitzhak Katznelson. *An Introduction to Harmonic Analysis*. Cambridge Mathematical Library. Cambridge University Press, 3 edition, 2004. doi: 10.1017/CBO9781139165372.

[221] Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodola. Accelerating transformer inference for translation via parallel decoding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12336–12355, 2023.

[222] Yang Song, Chenlin Meng, Renjie Liao, and Stefano Ermon. Accelerating feedforward computation via parallel nonlinear equation solving. In *International Conference on Machine Learning*, 2020. URL `https://api.semanticscholar.org/CorpusID:235422735`.

Appendix A

Adversarial Permutation Invariant Training for Universal Source Separation

A.1 Adversarial PIT for Speech Source Separation

We give a more detailed description of the Adversarial PIT methods for speech source separation in Table 2.1.

A.1.1 CBLDNN

CBLDNN [48] combines PIT regression losses (at magnitude STFT, filterbank and pitch domains) with adversarial training (a conditioned, context-based D in the magnitude STFT domain). The D input format is (fake / real): $[\mathbf{y}, \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2]$ / $[\mathbf{x}, \mathbf{x}_1, \mathbf{x}_2]$, as in Table 2.1. Hence, D is conditioned on the input mixture ($\mathbf{y}$), and is context-based because D has access to all sources ($\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2$ for fake or $\mathbf{x}_1, \mathbf{x}_2$ for real).

A.1.2 SSGAN-PIT

SSGAN-PIT [49] combines PIT regression in the magnitude STFT domain with three adversarial training variants: *(i)* conditioned, context-based; *(ii)* non-conditioned, context-based; and *(iii)* non-conditioned, instance-based (see Table 2.1). Variants *(i)* and *(ii)* are similar to CBLDNN [48] but *(i)* is conditioned on $\mathbf{y}$ and *(ii)* not. Finally, variant *(iii)* is not conditioned on $\mathbf{y}$, and is instance-based (instead of context-based) because D assesses each source individually (instead of assessing all sources together).

A.1.3 Furcax

Furcax [50] combines PIT regression in the waveform domain with adversarial training. The D is non-conditioned and context-based as SSGAN-PIT [49] variant *(ii)*, see Table 2.1.

A.1.4 Conv-TasSAN

Conv-TasSAN [51] combines PIT regression in the waveform domain with an alternative adversarial training setup called MetricGAN. Magnitude STFTs are fed into the D of MetricGAN to estimate normalized metric scores (from 0 to 1) for every pair of estimates $\hat{\mathbf{x}}_n$ and sources $\mathbf{x}_n$. Hence, the

D outputs 1s for real input pairs $[\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_2]$, and normalized metric scores (from 0 to 1) for the estimated (fake) pairs $[\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \mathbf{x}_1, \mathbf{x}_2]$. Note that if $\hat{\mathbf{x}}_k$ are bad estimates, the D outputs are close to 0s. They use PESQ [218] or STOI [219] as metrics.

Appendix B

Multi-Source Diffusion Models for Simultaneous Music Generation and Separation

B.1 Derivation of MSDM Dirac Posterior Score for Source Separation

We prove the main result of Section 4.3.3. We condition the generative model over the mixture $\mathbf{y}(0) = \mathbf{y}$. As such, we compute the posterior:

$$p(\mathbf{x}(t) \mid \mathbf{y}(0)) = \int_{\mathbf{y}(t)} p(\mathbf{x}(t), \mathbf{y}(t) \mid \mathbf{y}(0)) \mathrm{d}\mathbf{y}(t) = \int_{\mathbf{y}(t)} p(\mathbf{x}(t) \mid \mathbf{y}(t), \mathbf{y}(0)) p(\mathbf{y}(t) \mid \mathbf{y}(0)) \mathrm{d}\mathbf{y}(t)\,.$$

The first equality is given by marginalizing over $\mathbf{y}(t)$ and the second by the chain rule. Following Eq. (50) in [23], we can eliminate the dependency on $\mathbf{y}(0)$ from the first term, obtaining the approximation:

$$p(\mathbf{x}(t) \mid \mathbf{y}(0)) \approx \int_{\mathbf{y}(t)} p(\mathbf{x}(t) \mid \mathbf{y}(t)) p(\mathbf{y}(t) \mid \mathbf{y}(0)) \mathrm{d}\mathbf{y}(t)\,. \tag{B.1}$$

We compute $p(\mathbf{y}(t) \mid \mathbf{y}(0))$, using the chain rule after marginalizing over $\mathbf{x}(0)$ and $\mathbf{x}(t)$:

$$\begin{aligned} p(\mathbf{y}(t) \mid \mathbf{y}(0)) &= \int_{\mathbf{x}(0), \mathbf{x}(t)} p(\mathbf{y}(t), \mathbf{x}(t), \mathbf{x}(0) \mid \mathbf{y}(0)) \mathrm{d}\mathbf{x}(0) \mathrm{d}\mathbf{x}(t) \\ &= \int_{\mathbf{x}(0), \mathbf{x}(t)} p(\mathbf{y}(t) \mid \mathbf{x}(t), \mathbf{x}(0), \mathbf{y}(0)) p(\mathbf{x}(t) \mid \mathbf{x}(0), \mathbf{y}(0)) p(\mathbf{x}(0) \mid \mathbf{y}(0)) \mathrm{d}\mathbf{x}(0) \mathrm{d}\mathbf{x}(t)\,. \end{aligned}$$

By the Markov property of the forward diffusion process, $\mathbf{y}(t)$ is conditionally independent from $\mathbf{x}(0)$ given $\mathbf{x}(t)$ and we drop again the conditioning on $\mathbf{y}(0)$ from the first two terms, following Eq. (50) in [23]. As such, we have:

$$p(\mathbf{y}(t) \mid \mathbf{y}(0)) \approx \int_{\mathbf{x}(0), \mathbf{x}(t)} p(\mathbf{x}(0) \mid \mathbf{y}(0)) p(\mathbf{x}(t) \mid \mathbf{x}(0)) p(\mathbf{y}(t) \mid \mathbf{x}(t)) \mathrm{d}\mathbf{x}(0) \mathrm{d}\mathbf{x}(t)\,. \tag{B.2}$$

We model the likelihood function $p(\mathbf{y}(t) \mid \mathbf{x}(t))$ with the Dirac delta function in Eq. (4.9). The posterior $p(\mathbf{x}(0) \mid \mathbf{y}(0))$ is obtained via Bayes theorem substituting the likelihood:

$$p(\mathbf{x}(0) \mid \mathbf{y}(0)) = \frac{p(\mathbf{x}(0))\mathbb{1}_{\mathbf{y}(0)=\sum_{n=1}^{N}\mathbf{x}_n(0)}}{p(\mathbf{y}(0))} = \begin{cases} \frac{p(\mathbf{x}(0))}{p(\mathbf{y}(0))} & \text{if } \sum_{n=1}^{N}\mathbf{x}_n(0) = \mathbf{y}(0) \\ 0 & \text{otherwise} \end{cases}$$

We substitute it in Eq. (B.2), together with Eq. (4.1) and Eq. (4.9), obtaining:

$$\int_{\mathbf{x}(0):\sum_{n=1}^{N}\mathbf{x}(0)=\mathbf{y}(0)} \frac{p(\mathbf{x}(0))}{p(\mathbf{y}(0))} \int_{\mathbf{x}(t)} \mathcal{N}(\mathbf{x}(t); \mathbf{x}(0), \sigma^2(t)\mathbf{I})\mathbb{1}_{\mathbf{y}(t)=\sum_{n=1}^{N}\mathbf{x}_n(t)}\mathrm{d}\mathbf{x}(t)\mathrm{d}\mathbf{x}(0)\,. \tag{B.3}$$

We sum over the first $N-1$ sources in the inner integral, setting $\mathbf{x}_N(t) = \mathbf{y}(t) - \sum_{n=1}^{N-1}\mathbf{x}_n(t)$:

$$\int_{\mathbf{x}_{1:N-1}(t)} \mathcal{N}(\mathbf{x}_{1:N-1}(t), \mathbf{y}(t) - \sum_{n=1}^{N-1}\mathbf{x}_n(t); \mathbf{x}(0), \sigma^2(t)\mathbf{I})\mathrm{d}\mathbf{x}_{1:N-1}(t) \tag{B.4}$$

$$= \int_{\mathbf{x}_{1:N-1}(t)} \prod_{n=1}^{N-1} \mathcal{N}(\mathbf{x}_n(t); \mathbf{x}_n(0), \sigma^2(t)\mathbf{I})\mathcal{N}(\mathbf{y}(t) - \sum_{n=1}^{N-1}\mathbf{x}_n(t); \mathbf{x}_N(0), \sigma^2(t)\mathbf{I})\mathrm{d}\mathbf{x}_{1:N-1}(t)$$

$$= \mathcal{N}(\mathbf{y}(t); \sum_{n=1}^{N}\mathbf{x}_n(0), N\sigma^2(t)\mathbf{I})\,. \tag{B.5}$$

The second equality is obtained by factorizing the Gaussian, which has diagonal covariance matrix, while the last equality is obtained by iterative application of the convolution theorem [220]. We substitute Eq. (B.5) in Eq. (B.3), obtaining:

$$\begin{aligned} p(\mathbf{y}(t) \mid \mathbf{y}(0)) &\approx \int_{\mathbf{x}(0):\sum_{n=1}^{N}\mathbf{x}_n(0)=\mathbf{y}(0)} \frac{p(\mathbf{x}(0))}{p(\mathbf{y}(0))}\mathcal{N}(\mathbf{y}(t); \sum_{n=1}^{N}\mathbf{x}_n(0), N\sigma^2(t)\mathbf{I})\mathrm{d}\mathbf{x}(0) \\ &= \mathcal{N}(\mathbf{y}(t); \mathbf{y}(0), N\sigma^2(t)\mathbf{I}) \int_{\mathbf{x}(0):\sum_{n=1}^{N}\mathbf{x}_n(0)=\mathbf{y}(0)} \frac{p(\mathbf{x}(0))}{p(\mathbf{y}(0))}\mathrm{d}\mathbf{x}(0) \\ &= \mathcal{N}(\mathbf{y}(t); \mathbf{y}(0), N\sigma^2(t)\mathbf{I})\,. \end{aligned} \tag{B.6}$$

At this point, we apply Bayes theorem in Eq. (B.1), substituting the Dirac likelihood:

$$p(\mathbf{x}(t) \mid \mathbf{y}(0)) \approx \int_{\mathbf{y}(t)} \frac{p(\mathbf{x}(t))p(\mathbf{y}(t) \mid \mathbf{x}(t))}{p(\mathbf{y}(t))} p(\mathbf{y}(t) \mid \mathbf{y}(0))\mathrm{d}\mathbf{y}(t) \tag{B.7}$$

$$= \int_{\mathbf{y}(t)} \frac{p(\mathbf{x}(t))\mathbb{1}_{\mathbf{y}(t)=\sum_{n=1}^{N}\mathbf{x}_n(t)}}{p(\mathbf{y}(t))} p(\mathbf{y}(t) \mid \mathbf{y}(0))\mathrm{d}\mathbf{y}(t) \tag{B.8}$$

$$= \frac{p(\mathbf{x}(t))}{p(\sum_{n=1}^{N}\mathbf{x}_n(t))} p(\sum_{n=1}^{N}\mathbf{x}_n(t) \mid \mathbf{y}(0))\,. \tag{B.9}$$

Estimating Eq. (B.9), however, requires knowledge of the mixture density $p(\sum_{n=1}^{N}\mathbf{x}_n(t))$, which we do not acknowledge. As such, we approximate Eq. (B.8) with Monte Carlo, using the mean of

$p(\mathbf{y}(t) \mid \mathbf{y}(0))$, namely $\mathbf{y}(0)$ (see Eq. (B.6)), obtaining:

$$p(\mathbf{x}(t) \mid \mathbf{y}(0)) \approx \frac{p(\mathbf{x}(t))\mathbb{1}_{\mathbf{y}(0)=\sum_{n=1}^{N}\mathbf{x}_n(t)}}{p(\mathbf{y}(0))} = \begin{cases} \frac{p(\mathbf{x}(t))}{p(\mathbf{y}(0))} & \text{if } \sum_{n=1}^{N}\mathbf{x}_n(t) = \mathbf{y}(0) \\ 0 & \text{otherwise} \end{cases} \tag{B.10}$$

Similar to how we constrained the integral in Eq. (B.4), we parameterize the posterior, without loss of generality, using the first $N-1$ sources $\tilde{\mathbf{x}}(t) = (\mathbf{x}_1(t), \dots, \mathbf{x}_{N-1}(t))$. The last source is constrained setting $\mathbf{x}_N(t) = \mathbf{y}(0) - \sum_{n=1}^{N-1}\mathbf{x}_n(t)$ and the parameterization is defined as:

$$F(\tilde{\mathbf{x}}(t)) = F(\mathbf{x}_1(t), \dots, \mathbf{x}_{N-1}(t)) = (\mathbf{x}_1(t), \dots, \mathbf{x}_{N-1}(t), \mathbf{y}(0) - \sum_{n=1}^{N-1}\mathbf{x}_n(t)) \,. \tag{B.11}$$

Plugging Eq. (B.11) in Eq. (B.10) we obtain the parameterized posterior:

$$p(F(\tilde{\mathbf{x}}(t)) \mid \mathbf{y}(0)) \approx \frac{p(F(\tilde{\mathbf{x}}(t)))}{p(\mathbf{y}(0))} \tag{B.12}$$

At this point, we compute the gradient of the logarithm of Eq. (B.12) with respect to $\tilde{\mathbf{x}}(t)$:

$$\begin{aligned} \nabla_{\tilde{\mathbf{x}}(t)} \log p(F(\tilde{\mathbf{x}}(t)) \mid \mathbf{y}(0)) &\approx \nabla_{\tilde{\mathbf{x}}(t)} \log \frac{p(F(\tilde{\mathbf{x}}(t)))}{p(\mathbf{y}(0))} \\ &= \nabla_{\tilde{\mathbf{x}}(t)} \log p(F(\tilde{\mathbf{x}}(t))) - \nabla_{\tilde{\mathbf{x}}(t)} \log p(\mathbf{y}(0)) \\ &= \nabla_{\tilde{\mathbf{x}}(t)} \log p(F(\tilde{\mathbf{x}}(t))) \,. \end{aligned} \tag{B.13}$$

Using the chain-rule for differentiation on Eq. (B.13) we have:

$$\nabla_{\tilde{\mathbf{x}}(t)} \log p(F(\tilde{\mathbf{x}}(t)) \mid \mathbf{y}(0)) \approx \nabla_{F(\tilde{\mathbf{x}}(t))} \log p(F(\tilde{\mathbf{x}}(t))) J_F(\tilde{\mathbf{x}}(t)), \tag{B.14}$$

where $J_F(\tilde{\mathbf{x}}(t)) \in \mathbb{R}^{(N\times D)\times((N-1)\times D)}$ is the Jacobian of F computed in $\tilde{\mathbf{x}}(t)$, equal to:

$$J_F(\tilde{\mathbf{x}}(t)) = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} \\ -\mathbf{I} & -\mathbf{I} & \dots & -\mathbf{I} \end{bmatrix}$$

The gradient with respect to a source $\mathbf{x}_m(t)$ with $1 \le m \le N-1$ in Eq. (B.14) is thus equal to:

$$\begin{aligned} \nabla_{\mathbf{x}_m(t)} \log p(F(\tilde{\mathbf{x}}(t)) \mid \mathbf{y}(0)) \approx\ & [\nabla_{F(\tilde{\mathbf{x}}(t))} \log p(F(\tilde{\mathbf{x}}(t))]_m \\ & - [\nabla_{F(\tilde{\mathbf{x}}(t))} \log p(F(\tilde{\mathbf{x}}(t))]_N \,, \end{aligned}$$

where we index the components of the m-th and N-th sources in $\nabla_{F(\tilde{\mathbf{x}}(t))} \log p(F(\tilde{\mathbf{x}}(t))$. Finally, we replace the gradients with the score networks:

$$\nabla_{\mathbf{x}_m(t)} \log p(F(\tilde{\mathbf{x}}(t))|\mathbf{y}(0)) \approx S_{\theta,m}((\mathbf{x}_1(t), \ldots, \mathbf{x}_{N-1}(t), \mathbf{y}(0) - \sum_{n=1}^{N-1} \mathbf{x}_n(t)), \sigma(t)) \\ - S_{\theta,N}((\mathbf{x}_1(t), \ldots, \mathbf{x}_{N-1}(t), \mathbf{y}(0) - \sum_{n=1}^{N-1} \mathbf{x}_n(t)), \sigma(t)) \,, \tag{B.15}$$

where S_m^θ and S_N^θ are the entries of the score network corresponding to the m-th and N-th sources.

B.2 Derivation of Gaussian and Weakly-Supervised Posterior Scores for Source Separation

In this Section we derive the formulas for 'MSDM Gaussian', 'ISDM Dirac' and 'ISDM Gaussian'. We first adapt the Gaussian posterior introduced in [118] to continuous-time score-based diffusion models [169]. We plug the Gaussian likelihood function (Eq. (4.8)) into Eq. (B.7), obtaining:

$$p(\mathbf{x}(t) \mid \mathbf{y}(0)) \approx \int_{\mathbf{y}(t)} \frac{p(\mathbf{x}(t))\mathcal{N}(\mathbf{y}(t); \sum_{n=1}^{N} \mathbf{x}_n(t), \gamma^2(t)\mathbf{I})}{p(\mathbf{y}(t))} p(\mathbf{y}(t) \mid \mathbf{y}(0)) \mathrm{d}\mathbf{y}(t) \tag{B.16}$$

Following [118], $\mathbf{y}(t)$ is not re-sampled during inference and is always set to $\mathbf{y}(0)$. As such, we perform Monte Carlo in Eq. (B.16) with $\mathbf{y}(0)$, the mean of $p(\mathbf{y}(t) \mid \mathbf{y}(0)$ (see Eq. (B.6)), obtaining:

$$p(\mathbf{x}(t) \mid \mathbf{y}(0)) \approx \frac{p(\mathbf{x}(t))\mathcal{N}(\mathbf{y}(0); \sum_{n=1}^{N} \mathbf{x}_n(t), \gamma^2(t)\mathbf{I})}{p(\mathbf{y}(0))} \,. \tag{B.17}$$

At this point, we compute the gradient of the logarithm of Eq. (B.17) with respect to $\mathbf{x}_m(t)$:

$$\begin{aligned} \nabla_{\mathbf{x}_m(t)} \log p(\mathbf{x}(t) \mid \mathbf{y}(0)) &\approx \nabla_{\mathbf{x}_m(t)} \log \frac{p(\mathbf{x}(t))\mathcal{N}(\mathbf{y}(0); \sum_{n=1}^{N} \mathbf{x}_n(t), \gamma^2(t)\mathbf{I})}{p(\mathbf{y}(0))} \\ &= \nabla_{\mathbf{x}_m(t)} \log p(\mathbf{x}(t)) + \nabla_{\mathbf{x}_m(t)} \log \mathcal{N}(\mathbf{y}(0); \sum_{n=1}^{N} \mathbf{x}_n(t), \gamma^2(t)\mathbf{I}) \\ &= \nabla_{\mathbf{x}_m(t)} \log p(\mathbf{x}(t)) \; - \frac{1}{2\gamma^2(t)} \nabla_{\mathbf{x}_m(t)} \|\mathbf{y}(0) - \sum_{n=1}^{N} \mathbf{x}_n(t)\|_2^2 \\ &= \nabla_{\mathbf{x}_m(t)} \log p(\mathbf{x}(t)) \; - \frac{1}{\gamma^2(t)} (\mathbf{y}(0) - \sum_{n=1}^{N} \mathbf{x}_n(t)) \,. \end{aligned} \tag{B.18}$$

We obtain the 'MSDM Gaussian' posterior score by replacing the contextual prior with the score network:

$$\nabla_{\mathbf{x}_m(t)} \log p(\mathbf{x}(t) \mid \mathbf{y}(0)) \approx S_{\theta,m}((\mathbf{x}_1(t), \ldots, \mathbf{x}_N(t)), \sigma(t)) - \frac{1}{\gamma^2(t)} (\mathbf{y}(0) - \sum_{n=1}^{N} \mathbf{x}_n(t)) \,. \tag{B.19}$$

The weakly-supervised posterior scores are obtained by approximating:

$$p(\mathbf{x}_1(t), \ldots, \mathbf{x}_N(t)) \approx \prod_{n=1}^{N} p_n(\mathbf{x}_n(t)) \,,$$

where p_n are estimated with independent score functions $S_{\theta,n}$. In the contextual samplers in Eq. (B.15) ('MSDM Dirac') and Eq. (B.19) ('MSDM Gaussian'), $S_{\theta,n}((\mathbf{x}_1(t), \ldots, \mathbf{x}_N(t)), \sigma(t))$ refers to a slice of the full score network on the components of the $n-$th source. In the weakly-supervised cases, $S_{\theta,n}$ is an individual function. To obtain the 'ISDM Dirac' posterior score, we factorize the prior in Eq. (B.13), then use the chain rule of differentiation, as in Appendix B.1, to obtain:

$$\begin{aligned}\nabla_{\mathbf{x}_m(t)} \log p(F(\tilde{\mathbf{x}}(t)) \mid \mathbf{y}(0)) &\approx \nabla_{\mathbf{x}_m(t)} \log p_m(\mathbf{x}_m(t)) + \nabla_{\mathbf{x}_m(t)} \log p_N(\mathbf{y}(0) - \sum_{n=1}^{N-1} \mathbf{x}_n(t)) \\ &\approx S_{\theta,m}(\mathbf{x}_m(t), \sigma(t)) - S_{\theta,N}(\mathbf{y}(0) - \sum_{n=1}^{N-1} \mathbf{x}_n(t), \sigma(t)) \,.\end{aligned}$$

We obtain the 'ISDM Gaussian' posterior score by factorizing the joint prior in Eq. (B.18):

$$\nabla_{\mathbf{x}_m(t)} \log p(\mathbf{x}(t) \mid \mathbf{y}(0)) \approx S_{\theta,m}(\mathbf{x}_m(t), \sigma(t)) - \frac{1}{\gamma^2(t)}(\mathbf{y}(0) - \sum_{n=1}^{N} \mathbf{x}_n(t)) \,.$$

Acknowledgements

I first want to thank my loving wife Johana, who is always by my side and supports me every day; my wonderful mother Iulia, who worked hard to allow me to obtain an excellent education, and to which I am enormously grateful, and to my brother Dario, who has always lifted me in the difficult moments. As mentioned in the opening pages of the thesis, I dedicate this thesis to my father, Ioan, an excellent sculptor who, unfortunately, is no longer with us. He always worked diligently for the family's good and encouraged me always to give my best.

I want to thank Prof. E. Rodolà, head of the GLADIA[1] research group in Sapienza University of Rome, who, after the oral exam of the legendary course 'Fundamentals of Computer Graphics' in 2019, decided to allow me to collaborate with him, first as a research fellow and later as a Ph.D. student. He has always inspired me to pursue intriguing scientific topics that can have a real-world impact. Additionally, I thank him for the freedom he allowed us in choosing our study paths, which I greatly appreciated given my way of doing things: in fact, I transitioned from research in geometry processing, where I always appreciated the mathematical aspect, to research in audio, which I valued for its enormous potential and practical applicability.

This transition would not have been possible without the fortunate encounter with Dr. M. Mancusi, my colleague in the GLADIA group, with whom I shared scientific interests and a great friendship for many years. It was he who sparked my interest in audio topics, which I found to be a fantastic test-bed for my general interest in generative models. I also want to thank my colleague and great friend, G. Mariani, for our ongoing collaborations over the past years, from which I have learned a great deal, especially regarding the experimental and methodological aspects of research. Another person I want to thank is my colleague and great friend Irene Tallini, a researcher with a sharp mathematical intuition, who contributed significantly to the work presented in Chapter 4 by devising the Dirac separator (which I only helped formalizing). I also want to thank my GLADIA colleague and buddy A. Santilli, with whom we won the Imminent grant offered by the company Translated in 2022, having defined together parallel decoding in the field of automatic translation [221], following the preliminary work in [222].

Furthermore, I want to thank Prof. L. Cosmo, who has guided my scientific journey and growth from my first paper, effectively acting as my second advisor alongside Prof. E. Rodolà. I want to thank him for supporting us during difficult times, especially when we thought we would not achieve the desired results: he encouraged us to improve and not give up.

I collaborated for a considerable time with Profs. S. Melzi and S. Scardapane. Although my interests have shifted from those we initially shared, their valuable advice and the activities we carried out together have been beneficial to me (e.g., applying for the Galileo grant, which we subsequently won, and participating in the organization of the STAG 2021 conference, where I served as Web Chair).

Next, I want to thank Dr. J. Pons, who allowed me to pursue an internship at Dolby Laboratories in Barcelona. From him, I learned new technical and practical knowledge and, more importantly, how to better organize my work and become more proficient as a researcher.

I also want to thank Dr. E. Benetos, who gave me the opportunity to visit the Center for Digital Music (C4DM) at Queen Mary University of London. During this time, I experienced one of the

[1] `https://gladia.di.uniroma1.it/`

most beautiful periods of my life, engaging with the vibrant community of researchers at the center, meeting my wife Johana, and collaborating with the brilliant researchers M. Comunità, C.Y. Yu, and S. Saurjya, among the many others. I thank the Professor for his stimulating discussions and excellent suggestions during my time at the center.

Following, I thank Dr. N. Polouliakh and T. Akama from Sony CSL Tokyo, who allowed me to internship at the renowned Japanese laboratory in the last months of my Ph.D., where I worked on reconstructing music from EEG brainwaves.

The other co-authors that I have not already acknowledged are Dr. R. Marin, S. Severino, V. Maiorca, M. Fumero, Dr. S. Pascual, Dr. J. Serrà, Prof. J. D. Reiss, Prof. D. Comminiello, R. F. Gramaccioni, R. Ciranni, Prof. Hiroaki Kitano, and A. Connelly. I thank all of them for the hard work they put into our papers and all the support I received from them. I also thank the reviewers of my thesis, Dr. E. Tzinis and Prof. M. D. Plumbley, for their valuable suggestions for improving the manuscript.

Finally, I acknowledge the grants that financially supported me during my Ph.D. studies: ERC grant no. 802554 (SPECGEO), PRIN 2020 project no. 2020TA3K9N (LEGO.AI), PRIN 2022 project no. 2022AL45R2 (EYE-FI.AI, CUP H53D2300350-0001), PNRR MUR project no. PE0000013-FAIR, ICASSP-2024 Travel Grant (IEEE Signal Processing Society), AAAI-2023 Student Scolarship (Association for the Advancement of Artificial Intelligence), and Imminent Grant 2022 (Translated).

Emilian Postolache, Rome, May 2024