

SAPIENZA - UNIVERSITÀ DI ROMA

DOCTORAL THESIS

Harnessing the capabilities of Generative Models

Author:

Giorgio MARIANI

Supervisor:

Prof. Emanuele RODOLÀ



SAPIENZA
UNIVERSITÀ DI ROMA

“It’s easy to play any musical instrument: all you have to do is touch the right key at the right time and the instrument will play itself.”

Johann Sebastian Bach

SAPIENZA - UNIVERSITÀ DI ROMA

Abstract

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Computer Science Department

Computer Science

Harnessing the capabilities of Generative Models

by Giorgio MARIANI

Generative models have experienced significant advancements in recent years, driven by the introduction of architectures such as Stable Diffusion, GPT-3, ChatGPT, and many others. These models are designed to learn probability distributions and efficiently sample from them during inference, typically conditioned on inputs like text. Trained on large volumes of unlabeled data, these models possess extensive knowledge that can be transferred to address specific tasks. In this thesis, we show how they can be harnessed to address a variety of tasks across different domains, including reasoning, image processing, and music generation. In particular, we will explore diverse methodologies to guide the generation process of a learned model to better suit the task at hand.

Acknowledgements

I want to give my deepest gratitude to my supervisor, Prof. Emanuele Rodolà. His invaluable guidance has been crucial to my growth as a researcher and as a person. I am truly grateful for his mentorship and for all the opportunities he gifted me before and during this PhD.

A big thanks also goes to Prof. Luca Cosmo and Prof. Simone Melzi for their insightful advice throughout my PhD journey, and for having the skill and patience to help me and my colleagues in our journey. I'm also very grateful to Prof. Xavier Serra and all his collaborators at the Music Technology Group for welcoming me to their amazing research group during my visit to Pompeu Fabra University.

I owe a heartfelt thank you to my family for their unconditional love and support. They are the reason behind the person that I am today. I want to especially praise my father for his incredible dedication to providing for me and my siblings during our education and academic journeys.

Finally, I want to thank my collaborators, colleagues, and friends. Without their precious assistance and friendship, I would not have been able to reach my academic goals.

Contents

Abstract	iii
Acknowledgements	v
1 Introduction	1
1.1 Motivation	1
1.2 Thesis Outline	2
2 Background	5
2.1 Auto-Encoders	5
2.1.1 Variational Auto-Encoders	6
2.1.2 Vector Quantized VAEs	7
2.2 Generative Models	7
2.2.1 Autoregressive Models	7
2.2.2 Diffusion Models	8
2.3 Datasets	10
I Generative modeling in Reasoning and Language	13
3 Explanatory Learning	15
3.1 Introduction	16
3.2 Explanatory Learning	17
3.3 The Odeen Environment	19
3.3.1 Dataset	21
3.4 Critical Rationalist Networks	22
3.5 Experimental Results	24
3.6 Conclusions	27
4 Beyond the Imitation Game: BIG-bench	29
4.1 Symbol Interpretation Task	29
4.2 Experimental Result	34
4.3 Limitations and future work	35
II Generative modeling in the Signal domain	37
5 Latent Autoregressive Source Separation	39
5.1 Introduction	39
5.2 Related Work	41

5.3	Method	43
5.3.1	Latent Autoregressive Source Separation	43
5.3.2	Discrete Likelihoods for Source Separation	44
5.3.3	Inference Procedure	45
5.4	Experimental Results	46
5.4.1	Image Source Separation	46
5.4.2	Music Source Separation	49
5.5	Limitations	50
5.6	Conclusion	51
6	Diffusion Models for Multi-Source Music Generation	53
6.1	Introduction	53
6.2	Related Work	55
6.2.1	Generative Models for Audio	55
6.2.2	Compositional Waveform Music Generation	56
6.2.3	Audio Source Separation	57
6.3	Multi-Source Diffusion Models	58
6.3.1	Compositional Tasks	58
6.3.2	Experimental Results	62
	Music Generation	63
	Source Separation	64
6.3.3	Limitations	66
6.4	Generalized Multi-source Diffusion Inference	67
6.4.1	Total generation	67
6.4.2	Partial generation	69
6.4.3	Source separation	69
6.4.4	Experimental Setup	70
6.4.5	Experimental Results	70
6.4.6	Limitations	72
6.5	CompoNet	73
6.5.1	Experimental Setup	74
6.5.2	Experimental Results	75
6.6	Conclusions	76
7	Coherence-Oriented Contrastive Learning for Audio	77
7.1	Introduction	77
7.2	Related Work	78
7.2.1	Contrastive Methods for Audio	78
7.3	Method	80
7.3.1	Stem-Level Contrastive Learning	80
7.3.2	The COCOLA Score	81
7.4	Experimental Setup	81
7.5	Experimental Results	82
7.5.1	Coherent Sub-Mix Classification	82
7.6	Conclusion	83
8	Conclusions	85

Dedicated to Silvia, for always being there for me.

Chapter 1

Introduction

In this thesis, I will guide the reader through my academic journey in the field of generative modeling, and explore some of the various ways these models can be utilized.

1.1 Motivation

The AI spring. In the last decade, Artificial Intelligence has experienced remarkable growth in scale, research, and public interest. This surge has led to increased utilization across various fields and the emergence of high-impact companies such as *OpenAI*, *Hugging Face*, *Stability AI*, and many others. AI is now recognized for its potential to generate substantial business outcomes and reshape industries [1], sparking an *arms race* [2] in the development and improvement of AI systems and algorithms.

Several factors contribute to this success. However, the ease of accessing and processing large quantities of data, combined with the increased power of general-purpose Graphical Processing Units (GPUs), have been crucial. Particularly notable is the rise of powerful, large-scale generative models like GPT-2 through GPT-4, LLaMA 1 through 3, and Stable Diffusion versions 1.0, XL, and 3.0. These advancements have significantly increased business interest in AI applications. This phenomenon has been described as a “*pervasive economic and organizational phenomenon*” [3], and it is speculated that AI could boost the global economy by over \$15 trillion by 2030.

What do we mean by *harnessing* generative models? Generative models are typically trained using vast amounts of unlabeled (or poorly labeled) data, usually scraped from the web. A reduced and high-quality number of annotations are then used to guide this strong *self-supervised* representation to solve some different and usually more specific *downstream-tasks*. This is a common pipeline in the current AI setting and is sometimes referred to as *semi-supervised* learning [4, 5].

This thesis will explore and delve into several methods to effectively harness the generative power of the latest AI architectures and algorithms. Specifically, we will cover:

- (i) a combination of multiple models, cooperating with each other in order to solve a difficult reasoning task. We will see an example of this type of

approach in section 3.4, where a generative autoregressive model cooperates with a discriminator to improve their output. Similar approaches are used in image generation, where a contrastive model is used to select the best result with respect to a written prompt.

- (ii) the design and practical usage of algorithms that constrain the generation process to find a plausible solution for a specific task. An example that will appear multiple times in this thesis is the source separation problem¹,
- (iii) fine-tuning of a pre-trained generative model to solve a specific *downstream-task*. This is the typical *semi-supervised* pipeline [4, 5]. We adopted this approach for the model proposed in section 6.5.

1.2 Thesis Outline

The chapters of this thesis reflect our academic publications, each introducing and describing in detail the contributions, methodology, and results of our work.

During my first year of PhD, we started to explore the generative modeling world. This includes (i) a first approach to program synthesis/reasoning through the usage of autoregressive models, and (ii) participation in a collaborative benchmark for natural language LLMs. This line of research has led to the following works:

- *Explanatory Learning: Beyond Empiricism in Neural Networks [6]*. This paper introduces a reasoning approach that—using a combination of a generative model and a binary classifier—allows the production of improved *explanation* given an observed phenomenon. An in-depth explanation of this approach can be found in chapter 3, while the full article can be found at URL <https://arxiv.org/abs/2201.10222>.
- *Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models [7]*. This work is a collaborative benchmark designed to assess the performance of Large Language Models (LLMs) over a variety of difficult tasks. Our contribution was the proposal of the *Symbol Interpretation Task*. In this task, we test the understanding and ability to associate symbols with concepts in LLMs. This work is further presented in chapter 4. The whole benchmark paper can be found at <https://arxiv.org/abs/2206.04615>.

Throughout my second and third years, I became more interested in generative modeling in the music domain, where we experimented with various architectures and signal-processing tasks. Hereafter—in chronological order—are the research works resulting from this period:

¹See https://source-separation.github.io/tutorial/intro/src_sep_101.html for an explanation of the source separation problem.

- *Latent Autoregressive Source Separation [8]*. In this work, we explore the utilization of generative models in order to perform source separation on both the music and image domains. In particular, since the models work in the latent domain, a procedure to estimate sums in such domains is utilized. See chapter 5 for more information about this approach. The resulting paper can be found at <https://arxiv.org/abs/2301.08562>.
- *Multi-Source Diffusion Models for Simultaneous Music Generation and Separation [9]*. Most generative models on music and audio tend to work directly on mixture. In this work, we explore the utilization of a source-aware model that can be adapted to solve different music generation tasks. This work is further presented in section 6.3, while the pdf is available at <https://arxiv.org/abs/2302.02257>.
- *Generalized Multi-Source Inference for Text Conditioned Music Diffusion Models [10]* In this work, we explore an inference time procedure that allows the generation of coherent multiple-source audio. The full article can be found at <https://arxiv.org/abs/2403.11706>.
- *COCOLA: Coherence-Oriented Contrastive Learning of Musical Audio Representations [11]*. In this paper, we proposed a novel contrastive model useful for the evaluation of coherence between musical sources. While this is not directly related to generative models, it might be useful as a specific evaluation metric and possibly as a conditioning classifier throughout the generation procedure (similarly to DiffusionCLIP [12]). More about this work in chapter 7. The pdf is available at <https://arxiv.org/abs/2404.16969>.

Chapter 2

Background

This chapter provides a simple background on some of the key architectures and models that form the foundation of this thesis, specifically focusing on auto-encoders and generative models.

Auto-encoders. Auto-encoders are a class of neural network architectures designed for unsupervised learning tasks. They consist of (i) an *encoder* that maps input data to a latent representation, and (ii) a *decoder* that reconstructs the input from this latent space. Auto-encoders are widely used for dimensionality reduction, feature learning, and data denoising. Section 2.1 will delve into some auto-encoders variants, highlighting their unique characteristics and applications.

Generative models. On the other hand, generative models are designed to generate new data samples according to a given training data distribution. In recent years, these models have gained significant attention due to their ability to learn complex data distributions. In this chapter, we will explore two main types of generative models: *autoregressive models* and *diffusion-based models*. Autoregressive models generate data sequentially, one element at a time, by modeling the conditional distribution of each element given the previous ones. In contrast, diffusion-based models generate data by gradually transforming a simple initial distribution into the target distribution through a series of learned steps.

By providing a thorough understanding of these models, section 2.2 sets the stage for the subsequent discussion of their applications and implications in the context of this research. The insights gained here will be instrumental in appreciating the novel contributions presented in the later chapters of this thesis.

2.1 Auto-Encoders

Autoencoders are a type of unsupervised machine learning architecture commonly used for dimensionality reduction and feature extraction. They compress the input data into a lower dimensional latent vector, which is then decoded into the original data by a decoder network. In practice, we can think of the compressed latent vector as a more meaningful and semantic representation of the original data. Figure 2.1 shows a typical auto-encoder architecture.

Properties. Similarly to other dimensionality reduction approaches, auto-encoders are data-specific and lossy, thus they might not preserve the original data quality, nor generalize well on out-of-distribution data. On the other hand, they are able to extract informative—often semantic—and non-linear representations of the original data.

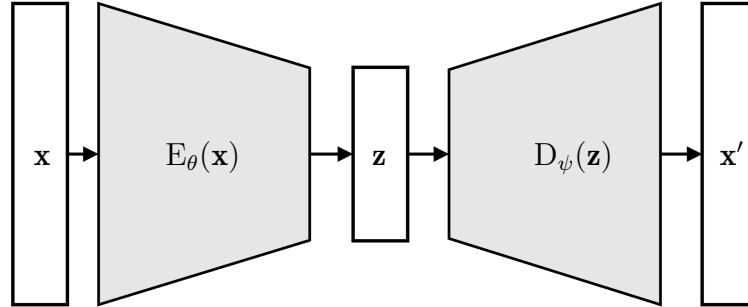


FIGURE 2.1: Illustration of a generic auto-encoder architecture.

Formally, a data point $\mathbf{x} \in \mathbb{R}^N$ (with N the total length of the data point¹) can be mapped to a smaller latent vector $\mathbf{z} \in \mathbb{R}^C$ via an encoder network $E_{\theta}(\mathbf{x}) = \mathbf{z}$. The vector \mathbf{z} can then be brought back into the initial domain through the decoder network $D_{\psi}(\mathbf{z}) = \mathbf{x}'$. If \mathbf{x} is in the data distribution and both E_{θ} and D_{ψ} have reached an adequate enough solution, then we should have that

$$\mathbf{x} \approx D_{\psi}(E_{\theta}(\mathbf{x})).$$

2.1.1 Variational Auto-Encoders

While useful, simple auto-encoder architectures suffer from an important drawback: they can easily overfit the training data, thus making the latent representation less strong and possibly highly irregular. A possible way to alleviate this issue is the use of *Variational Auto-Encoders* (VAEs) [13]. Simply put, VAE can be seen as auto-encoders whose latent distribution is regularized by an improved loss function, which utilizes variational inference theory.

The training loss for VAEs is composed of two terms which—intuitively—denote the reconstruction loss of the autoencoder and the regularization loss for the latent space.

$$\mathcal{L}_{\text{VAE}}(\mathbf{x}) = \alpha \|\mathbf{x} - D_{\theta}(\mathbf{z})\|^2 + \text{Div}_{\mathcal{KL}}(\mathcal{N}(\mu_{\mathbf{x}}, \sigma_{\mathbf{x}}) \parallel \mathcal{N}(0, 1)) \quad (2.1)$$

with $(\mu_{\mathbf{x}}, \sigma_{\mathbf{x}}) = E_{\theta}(\mathbf{x})$ and \mathbf{z} sampled from $\mathcal{N}(\mu_{\mathbf{x}}, \sigma_{\mathbf{x}})$ using the *reparametrization trick* [13]. The hyper-parameter $\alpha \in \mathbb{R}$ in eq. (2.1) balances the two loss terms.

¹For example, the length of the audio sequence or the number of pixel channels in an image

2.1.2 Vector Quantized VAEs

Vector Quantized VAEs (VQ-VAE) [14] are an auto-encoder architecture that allows the mapping between continuous-feature vectors into vectors of discrete features. This type of translation can be useful if a discrete representation of the data is necessary².

Formally, a data point $\mathbf{x} \in \mathbb{R}^N$ can be mapped to a discrete latent domain via a VQ-VAE. First an encoder $E_\theta : \mathbb{R}^N \rightarrow \mathbb{R}^{S \times C}$ 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 $B : \mathbb{R}^{S \times C} \rightarrow [K]^S$ casts the encoding into a discrete sequence $\mathbf{z} = (z_1, \dots, z_S)$ by mapping each \mathbf{h}_s into the index³ $z_s = B(\mathbf{h}_s)$ of the nearest neighbor \mathbf{e}_{z_s} contained in an ordered set $\mathcal{C} = \{\mathbf{e}_k\}_{k=1}^K$ of learned vectors—called codes—in \mathbb{R}^C . A decoder $D_\psi : [K]^S \rightarrow \mathbb{R}^N$ maps the latent sequence back into the data domain, obtaining a reconstruction $\hat{\mathbf{x}} = D_\psi(\mathbf{z})$.

Discriminator augmented training. VQ-GANs [15] are an enhanced version of the VQ-VAE architecture, where the training loss is augmented with a discriminator and a perceptual loss. These additions improve reconstruction quality while increasing the compression rate of the autoencoder. We refer the reader to [14] and [15] for more details on VQ-VAE and VQ-GAN. In the remainder of the thesis, we will refer to both models as VQ-VAE and make distinctions only if necessary.

2.2 Generative Models

Generative models are a class of Machine Learning algorithms that aim to reproduce—as best as possible—a given training data distribution. A variety of generative models have been developed, each with its advantages and disadvantages. However, for our most of our purposes, we are interested in two kinds: autoregressive and diffusion-based. At the moment, these are the state-of-the-art in text, image, and audio generation.

2.2.1 Autoregressive Models

An *autoregressive model* [16–18] defines a probability distribution over a discrete domain⁴ $[K]^S$. In particular, they are often used to model the probabilities of sequences $\mathbf{z} = (z_1, \dots, z_S)$ of (usually) variable length. Typically, autoregressive models are used for Natural Language Processing tasks, however, they have seen extensive usage in several other domains [17–20].

²For example, this is usually the case for autoregressive models.

³Sometime also referred to as “token”.

⁴e.g. the words of the English language, or the latent domain of a VQ-VAE.

Sampling. In autoregressive architectures, the joint probability over the sequence $\mathbf{z} = (z_1, \dots, z_S)$ is decomposed via the chain rule:

$$P_\phi(\mathbf{z}) = \prod_{s=1}^S P_\phi(z_s | \mathbf{z}_{<s}), \quad (2.2)$$

where $p_\phi(\cdot)$ is a learned parametric model⁵. At inference time, samples can be drawn from eq. (2.2) using several possible sampling procedures. Ancestral sampling is often used, where at each step, the token z_s is sampled stochastically from the conditional $P_\phi(z_s | \mathbf{z}_{<s})$, possibly employing top- k [23] filtering to improve the diversity of the generated data [24]. When the goal is instead to maximize the probability of the whole sequence (w.r.t. all the sequences), heuristics like beam search can also be used [25]. Beam search maintains B possible hypotheses (beams) $\mathbf{z}^1, \dots, \mathbf{z}^B$ in parallel during inference. At each step s , it computes the conditional distributions $P_\phi(z_s^b | \mathbf{z}_{<s}^b)$ for each beam and selects the B new hypotheses that maximize the joint distributions $P_\phi(\mathbf{z}_{<s}^b) \cdot P_\phi(z_s^b | \mathbf{z}_{<s}^b)$.

2.2.2 Diffusion Models

Diffusion Models [26–29] are a class of generative models that are able to sample by iteratively denoising random noise. Intuitively, they learn how data can be progressively perturbed until it is no longer recognizable. Then, they use this knowledge to approximate the inverse process: starting from some random noise, they progressively denoise it until a valid sample is reached.

Formal definition. Like many other generative approaches, Diffusion models do not try to model the probability density of the training data. Instead, they model the gradient of the log-probability density of the *perturbed* training data, usually through the use of a neural network. Thus, a diffusion model $S_\theta(\cdot)$ parameterizes:

$$S_\theta(\mathbf{x}(t), \sigma(t)) \approx \nabla_{\mathbf{x}(t)} \log p(\mathbf{x}(t)). \quad (2.3)$$

More specifically, diffusion models borrow a lot from the score-matching theory: The central idea of score-matching [30–32] is to approximate the *score function* of the target density $p(\mathbf{x})$, namely $\nabla_{\mathbf{x}} \log p(\mathbf{x})$, rather than the density 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:

$$p(\mathbf{x}(t) | \mathbf{x}(0)) = \mathcal{N}(\mathbf{x}(t); \mathbf{x}(0), \sigma^2(t)\mathbf{I}), \quad (2.4)$$

where the parameter $\sigma(t)$ regulates the degree of noise added to the data. Following the authors in [29], 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 [27]:

$$d\mathbf{x}(t) = -\sigma(t)\nabla_{\mathbf{x}(t)} \log p(\mathbf{x}(t)) dt. \quad (2.5)$$

⁵Generally neural networks such as CNNs [21, 22] or Transformers [16].

For $t = T \gg 0$, 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. Equation (2.5) can be inverted in time, resulting in the following backward ODE that describes the denoising process:

$$d\mathbf{x}(t) = \sigma(t)\nabla_{\mathbf{x}(t)} \log p(\mathbf{x}(t)) dt. \quad (2.6)$$

NOTE:

Sampling can be performed from the data distribution integrating eq. (2.6) with a standard ODE solver, starting from an initial (noisy) sample drawn from $\mathcal{N}(\mathbf{x}(t); \mathbf{0}, \sigma^2(T)\mathbf{I})$.

Training procedure. The score function $S_\theta(\mathbf{x}(t), \sigma(t))$ is approximated by minimizing the following denoising score matching loss:

$$\mathbb{E}_{t, \mathbf{x}(0), \mathbf{x}(t)} [\|S^\theta(\mathbf{x}(t), \sigma(t)) - \nabla_{\mathbf{x}(t)} \log p(\mathbf{x}(t) | \mathbf{x}(0))\|_2^2] \quad (2.7)$$

with the random variable t , $\mathbf{x}(0)$, $\mathbf{x}(t)$ following the densities

$$\begin{aligned} t &\sim \mathcal{U}([0, T]), \\ \mathbf{x}(0) &\sim p(\mathbf{x}(0)), \\ \mathbf{x}(t) &\sim p(\mathbf{x}(t) | \mathbf{x}(0)). \end{aligned}$$

Even if feasible, the score function is -most of the time- not directly approximated by a neural network. Instead, some transformations are usually applied to the raw output of the network—denoted as $F_\theta(\mathbf{x})$. In our case, following the theory of [29], we define the score function as

$$S_\theta(\mathbf{x}, \sigma) = \frac{D_\theta(\mathbf{x}; \sigma) - \mathbf{x}}{\sigma^2} \quad (2.8)$$

with the *denoiser function* $D_\theta(\mathbf{x}; \sigma)$ equal to

$$D_\theta(\mathbf{x}; \sigma) = c_{\text{skip}}(\sigma)\mathbf{x} + c_{\text{out}}(\sigma)F_\theta(c_{\text{in}}(\sigma)\mathbf{x}; c_{\text{noise}}(\sigma)).$$

Finally, the score-matching loss in eq. (2.7) can be simplified by expanding the term $p(\mathbf{x}(t) | \mathbf{x}(0))$ with eq. (2.4) and substituting S_θ with eq. (2.8). The resulting loss function is then defined as

$$\mathbb{E}_{t, \mathbf{x}(0), \boldsymbol{\epsilon}} [\|D_\theta(\mathbf{x}(0) + \boldsymbol{\epsilon}; \sigma(t)) - \mathbf{x}(0)\|_2^2],$$

with the Gaussian noise variable $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \sigma^2(t)\mathbf{I})$.

Text-conditioned diffusion models. It is possible to add a variable \mathbf{z} representing an (often textual) conditioning input. In other words, eq. (2.3)

slightly changes to the following equation:

$$\nabla_{\mathbf{x}(t)} \log p(\mathbf{x}(t) \mid \mathbf{z}) \approx S_{\theta}(\mathbf{x}(t), \mathbf{z}, \sigma(t)), \quad (2.9)$$

The practical way in which conditioning is performed may depend on the architecture itself and the nature of the conditioning embedding \mathbf{z} .

Classifier-free guidance. Using an implicit classifier defined from the model itself, *classifier-free guidance* [33] enables the sampling of values that better suit the conditioning prompt \mathbf{z} . This can be done by defining a new score function $S_{\theta}^*(\mathbf{x}(t), \mathbf{z}, \sigma(t))$ defined as

$$S_{\theta}^*(\mathbf{x}(t), \mathbf{z}, \sigma(t)) = S_{\theta}(\mathbf{x}(t), \mathbf{z}, \sigma(t)) + w (S_{\theta}(\mathbf{x}(t), \mathbf{z}, \sigma(t)) - S_{\theta}(\mathbf{x}(t), \emptyset, \sigma(t))) ,$$

where \emptyset is a fixed embedding modeling the unconditional log probability density $\nabla_{\mathbf{x}(t)} \log p(\mathbf{y}(t))$, and $w \in \mathbb{R}$ is the embedding scale hyper-parameter. We can use a *negative embedding* [34] instead of \emptyset to better guide inference. With an abuse of notation, we will refer to S_{θ}^* as S_{θ} , and make a distinction only if necessary.

2.3 Datasets

For our research, we utilized a variety of datasets. Here, we highlight some of the most important ones, providing brief descriptions of their respective sizes and contents.

MNIST [35] is a well-established Machine Learning dataset, being used in several academic works. It includes 70,000 images of hand-written digits, 60,000 of which compose the training set, while the remaining 10,000 are used as the test set. Each image is the size of 28×28 pixels and in grayscale color.

CelebA [36] is a large-scale image dataset, composed of more than 200k images of celebrity faces. For each image, there are 40 attribute labels available, ranging from hairstyles, glasses, face shape, and many more. All images in CelebA have a resolution of 32×32 pixels. CelebA-HQ [37] is a higher resolution variant of CelebA, composed of 30,000 images with a resolution of 1024×1024 pixels each.

ImageNet [38] is an important dataset in the Computer Vision community. It has been used in several works, ranging from object recognition to generative modeling tasks. It contains more than 14 million images, each annotated to a WordNet [39] synset by human annotators.

MUSDB18-HQ [40] is the uncompressed version (in WAV format) of the MUSDB18 dataset, initially introduced in [41]. This dataset is a standard for evaluating music source separation systems. It comprises 150 tracks—100 for training and 50 for testing—totaling approximately 10 hours of

professional-quality audio. Each track is divided into four stems: *Bass*, *Drums*, *Vocals*, and *Other*, with *Other* covering any components not classified under the first three categories.

MoisesDB [42] features 240 music tracks across diverse genres and artists, accumulating more than 14 hours of music. Unlike MUSDB18-HQ, MoisesDB is a genuine multi-track dataset, offering a two-tier taxonomy of 11 distinct stems. Each stem in this dataset includes more detailed type annotations (e.g., a guitar might be labeled as *Acoustic Guitar* or *Electric Guitar*).

MTG-Jamendo [43] is an open a dataset for the music auto-tagging task. It contains music accessible at the Jamendo⁶ website. This dataset contains over 55,000 audio mixtures, annotated with a total of 195 tags. The information in these annotation is about the musical genre, which instruments are in the mixtures, and the mood/theme of the song.

Slakh2100 [44] is synthesized from the Lakh MIDI Dataset v0.1 [45] employing high-quality sample-based virtual instruments. It features 2100 tracks organized into 1500 tracks for training, 375 for validation, and 225 for testing, together amounting to 145h of audio. The tracks are annotated into 34 stem categories. While such a dataset contains an order of magnitude more data than MUSDB18-HQ and MoisesDB, it does not share the same level of realism as the latter, being the tracks synthesized from MIDI.

CocoChorales [46] is a chorale audio music dataset created through a synthesis process like Slakh2100. However, it comprises a substantially vaster collection of 240000 tracks, extending over 1411 hours of audio data. It is produced by generating symbolic notes via a Coconet model, performing their synthesis with MIDI-DDSP [47]. This dataset is richly annotated, featuring details on performance attributes and synthesis parameters. CocoChorales includes a diverse range of 13 instruments spanning *Strings*, *Brass*, *Woodwind*, and *Random ensembles*.

⁶<https://www.jamendo.com>

Part I

Generative modeling in Reasoning and Language

Chapter 3

Explanatory Learning

In this chapter, we introduce Explanatory Learning (EL), a framework aiming to improve machines' capability to interpret and utilize existing knowledge embedded in symbolic sequences.

This interpreter is developed using a limited collection of symbolic sequences paired with observations of various phenomena. It can then be used to make predictions about new phenomena based on their explanations and even discover these explanations with minimal observations, akin to the methods employed by human scientists. We conceptualize the EL problem as a straightforward binary classification task, allowing simple end-to-end machine learning approaches, to potentially solve it. However, we propose an alternative with Critical Rationalist Networks (CRNs). These networks adopt a rationalist perspective on knowledge acquisition. CRNs are inherently designed to exhibit several desirable properties: they are genuinely explainable, can adjust their processing at test time for more challenging inferences, and provide strong confidence guarantees in their predictions.

As a part of our contributions, we introduced Odeen, an elementary EL environment that simulates a simple flatland-style universe populated with phenomena to explain. Using Odeen as a testing ground, we show how our CRNs architecture outperforms end-to-end approaches of comparable size and architecture in discovering explanations for new phenomena. This chapter delves into the principles of Explanatory Learning, the construction and advantages of Critical Rationalist Networks, and the experimental validation using the Odeen environment.

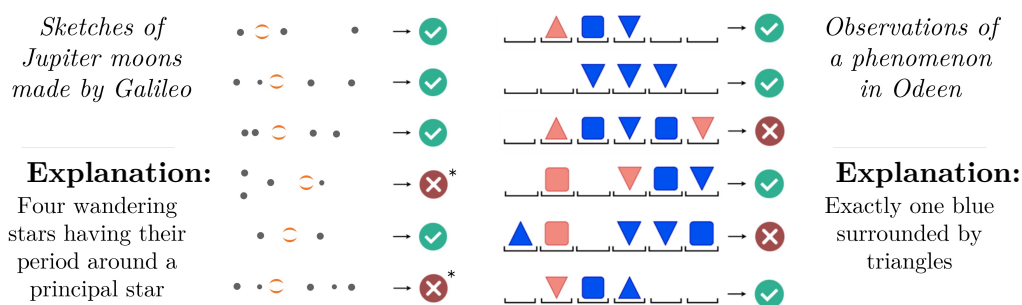


FIGURE 3.1: **The Odeen universe.** The intention behind this universe is to create a convenient environment to study and test the process of knowledge discovery in machines.

3.1 Introduction

Predicting the future with accuracy is arguably a key ability for survival and prosperity in any habitat. Living beings appear to have evolved various systems to achieve this, such as memory [48], and many seem capable of predicting the course of complex phenomena [49]. However, no animal seems to match the predictive capabilities of humans, which might stem from a unique system in nature.

At the core of this potential system is something we refer to as an *explanation*, formed through language. This potentially allows explanations to be transferred to another human who speaks the same language, enabling them to predict new phenomena without prior experience. When this transfer is successful, we say that the human has *understood* the explanation. This process is a fundamental aspect of human success. An individual might make accurate predictions about numerous phenomena without undergoing a painful discovery process for each one. All that seems necessary is an operating system—mastery of a language—and someone to communicate the relevant explanations. This would allow the individual to focus on unexplained phenomena. When an explanation is found, it is added to the existing shared collection, known as *knowledge*.

How might we integrate machines into this system? In this work, we aim to explore this intriguing problem. Specifically, we propose a learning procedure that could enable machines (i) to *understand* existing explanations, as described above, and (ii) to create new explanations for unexplained phenomena, much like humans seem to do.

Contributions. Our contribution in this sense is threefold:

- (i) We formulate the challenge of creating a machine that masters a language as the problem of learning an interpreter from a collection of examples in the form (explanation, observations). The only assumption we make is this dual structure of data; explanations are free strings and are not required to fit any formal grammar. This results in the *Explanatory Learning* (EL) framework described in section 3.2.
- (ii) We present Odeen, a basic environment to test EL approaches, which draws inspiration from the board game Zendo [50]. Odeen simulates the work of a scientist in a small universe of simple geometric figures, see fig. 3.1. We present it in section 3.3, and it is openly available for download¹.
- (iii) We argue that the *empiricist* Machine Learning approaches might not be best suited for EL problems. Instead, we propose the *Critical Rationalist Networks* (CRNs), a family of models designed following the epistemological philosophy pushed forward by [51]. Although a CRN is implemented

¹The Odeen dataset and all the code useful to reproduce the results discussed in this chapter can be found at <https://github.com/gladia-research-group/explanatory-learning>

using two neural networks, the working hypothesis of such a model does not coincide with the adjustable network parameters, but rather with a language proposition that can only be accepted or refused *in toto*. We will present CRNs in section 3.4, and test their performance on Odeen in section 3.5.

3.2 Explanatory Learning

Humans do not master a language from birth. For instance, a toddler cannot understand the message “this soap stings” and predict the burning sensation from contact with the substance. Instead, the child gradually *learns* to interpret such messages and make predictions about a wide range of phenomena [52]. We call this process *mastering a language* and aim to replicate it in machines through a similar learning process.

By using a set of explanations alongside observations of various phenomena, we aim to develop an interpreter capable of determining whether a given phenomenon matches a provided explanation. Furthermore, we want to uncover these explanations using only a limited number of observations of new phenomena. We first describe the problem setup in the following paragraph, comparing it to existing ML problems; then we detail our approach in section 3.4.

Problem setup. Formally, let phenomena P_1, P_2, P_3, \dots be subsets of a universe U , which is a large set with no special structure (i.e., all the possible observations $U = \{x_1, \dots, x_z\}$). Over a universe U , one can define a language L as a pair $(\Sigma_L, \mathcal{I}_L)$, where Σ_L is a finite collection of short strings over some alphabet A , with $|\Sigma_L| \gg |A|$, and \mathcal{I}_L is a binary function $\mathcal{I}_L : U \times \Sigma_L \rightarrow \{0, 1\}$, which we call *interpreter*.

Definition. We say that a phenomenon P_i is explainable in a language L if there exists a string $e \in \Sigma_L$ such that, for any $x \in U$, it occurs $\mathcal{I}_L(x, e) = \mathbf{1}_{P_i}(x)$, where $\mathbf{1}_{P_i}(x)$ is the indicator function of P_i . We call the string e an *explanation*, in the language L , for the phenomenon P_i .

Consider the general problem of making a new prediction for a phenomenon $P_0 \subset U$. In our setting, this is phrased as a binary classification task: given a sample $x' \in U$, establish whether $x' \in P_0$ or not. We are interested in two instances of this problem, with different underlying assumptions:

- **The communication problem: we have an explanation.** We are given an explanation e_0 for P_0 , in an unknown language L . This means that we do not have access to an interpreter \mathcal{I}_L ; e_0 looks like Japanese to a non-Japanese speaker. Instead, we are also given other explanations $\{e_1, \dots, e_n\}$, in the same language, for other phenomena P_1, \dots, P_n , as well as observations of them, i.e., datasets $\{D_1, \dots, D_n\}$ in the form $D_i = \{(x_1, \mathbf{1}_{P_i}(x_1)), \dots, (x_m, \mathbf{1}_{P_i}(x_m))\}$, with $m \ll |U|$. Intuitively, here we expect the learner to use the explanations paired with the observations

to build an approximated interpreter $\hat{\mathcal{I}}_L$, and then use it to make the proper prediction for x' by evaluating $\hat{\mathcal{I}}_L(x', e_0)$.

- **The scientist problem: we do not have an explanation.** We are given explanations $\{e_1, \dots, e_n\}$ in an unknown language L for other phenomena P_1, \dots, P_n and observations of them $\{D_1, \dots, D_n\}$. However, we do not have an explanation for P_0 ; instead, we are given just a small set of observations $D_0 = \{(x_1, \mathbf{1}_{P_0}(x_1)), \dots, (x_k, \mathbf{1}_{P_0}(x_k))\}$ and two guarantees, namely that P_0 is explainable in L , and that D_0 is *representative* for P_0 in L . That is, for every phenomenon $P \neq P_0$ explainable in L there should exist at least a $x_i \in D_0$ such that $\mathbf{1}_{P_0}(x_i) \neq \mathbf{1}_P(x_i)$. Again, we expect the learner to build the interpreter $\hat{\mathcal{I}}_L$, which should first guide the search for the missing explanation e_0 based on the clues D_0 , and then provide the final prediction through $\hat{\mathcal{I}}_L(x', e_0)$.

Several existing works fall within the formalization above. The seminal work of [53] on learning regular sets is an instance of the scientist problem, where finite automata take the role of explanations, while regular sets are the phenomena. More recently, CLEVR [54] posed a communication problem in a universe of images of simple solids, where explanations are textual and read like “*There is a sphere with the same size as the metal cube*”. Another example is CLIP [55], where 400,000,000 captioned internet images are arranged in a communication problem to train an interpreter, thereby elevating captions to the status of explanations rather than treating them as simple labels. With EL, we aim to offer a unified perspective on these works, making explicit the core problem of learning an interpreter purely from observations.

Relationship with other ML problems. EL can be framed in the general meta-learning framework. The learner gains experience over multiple tasks to improve its general learning algorithm, thus requiring less data and computation on new tasks. However, differently from current meta-learning approaches [56], we are not optimizing for any meta-objective. Instead, we expect the sought generality to be a consequence of implicitly defining an interpreter through a limited set of examples rather than an explicit goal to optimize for.

To many, the concept of explanation may sound close to the concept of program; similarly, the scientist problem may seem a rephrasing of the fundamental problem of Inductive Logic Programming (ILP) [57] or Program Synthesis (PS) [58]. While similar in nature, this is not the case. ILP has the analogous goal of producing a hypothesis from positive/negative examples accompanied by background knowledge. Yet, ILP requires observations to be expressed as logic formulas, a task requiring a human; only then the ILP solver outputs an explanation in the form of a logic proposition, which in turn is interpreted by a human expert. With EL, data can be fed as-is without being translated into logic propositions, and a learned interpreter plays the expert’s role. PS also admits raw data as input, it yields a program as output, and replaces the expert with a handcrafted interpreter; still, the sequence of symbols produced by a PS system only makes sense to a human (who designed the interpreter), not to the system itself. Instead, in EL, the interpreter is learned from data rather than

hardcoded. An empirical comparison demonstrating the benefits of EL over PS is given in section 3.5.

3.3 The Odeen Environment

In this section, we introduce Odeen, an environment and benchmark to experiment with the EL paradigm. We can think of Odeen as an environment composed of *rules*, *labels*, and *structures*. In this universe, it is possible to play many games, each using:

- (i) A single “hidden” rule, unknown to the player. Some examples of rules are: “*At least one red square*”, “*Exactly one circle*”, or “*At least one square at the right of a blue circle*”.
- (ii) A group of structures and boolean labels pairs, with each pair indicating whether or not a structure is consistent with the hidden rule. A structure itself can be seen as a sequence of simple geometric shapes.

The player then looks at the set of structures, labeled according to the secret rule, and their goal is to guess it. To win the game, a player must prove to know the rule by correctly tagging a large set of new structures² with the appropriate labels. Figure 3.2 shows a typical situation in a game of Odeen; for example—in this particular game—the rule cannot possibly be “*A structure must contain at least one red square*” since the fifth structure on the left does not contain a red square, but respects the rule (as marked by the green label).

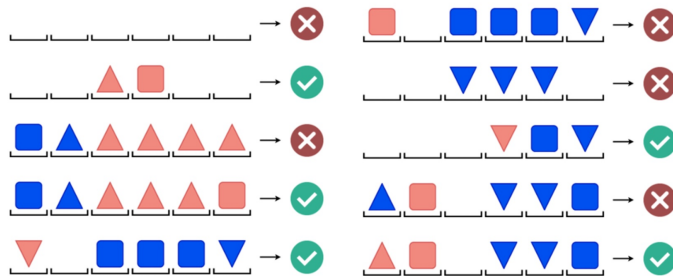


FIGURE 3.2: **Example of an Odeen game.** The hidden rule to this particular instance is “*At least one square at the right of a red pyramid*”. Indeed, the attentive reader will notice that it is consistent with all the provided observations.

Connection with Explanatory Learning. We can view each game of Odeen as a distinct phenomenon within a universe, where each element is a sequence of geometric figures. In this universe, players act as scientists—akin to Galileo observing Jupiter’s moons—attempting to explain new phenomena (refer to fig. 3.1 for a more illustrative example). The challenge for an Odeen scientist

²Odeen is inspired by the board game Zendo, in which players must explicitly guess a hidden rule, known only to a master. In Zendo, players can also experiment by submitting new structures to the master.

can then be framed as follows: make accurate predictions for a new phenomenon given a few observations of it, alongside explanations and observations of other phenomena. This encapsulates the essence of the Odeen Explanatory Learning problem, as illustrated in fig. 3.3 (A and B).

Each game of Odeen is a different phenomenon P_i of a universe U whose elements x are sequences of geometric figures. The specific task is to make correct predictions for a new phenomenon P_0 (a new game) given: (i) a few observations D_0 of P_0 (labeled structures), in conjunction with (ii) explanations $\{e_1, \dots, e_n\}$ and observations $\{D_1, \dots, D_n\}$ of other phenomena (other games and their secret rules). More formally:

Definition. *Let us be given s unexplained phenomena with k observations each, and n explained phenomena with m observations each; let the n phenomena be explained in an unknown language, i.e., e_1, \dots, e_n are plain strings without any interpreter. The task is to make ℓ correct predictions for each of the s unexplained phenomena.*

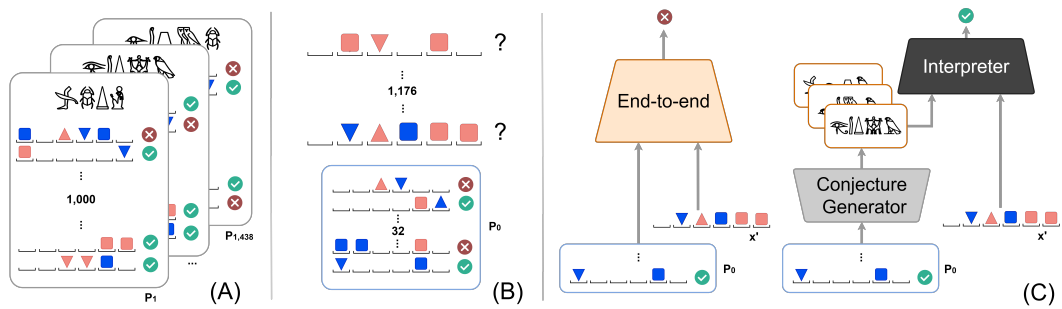


FIGURE 3.3: **The Odeen Explanatory Learning problem.**

Given observations and explanations in an unknown language for some phenomena (A), plus a few observations of a new phenomenon, explain the latter and prove this knowledge by correctly tagging a large set of new samples (B). An empiricist approach attempts to extract this knowledge from data (C, left); a rationalist one conceives data as theory-laden observations, used to find the true explanation among a set of conjectures (C, right).

Instead of requiring the player to reveal the secret explanation explicitly, we follow the principle of zero-knowledge proofs [59]. In our setting, this is done by asking the player to correctly tag many unseen structures according to the discovered rule. This makes it possible for any binary classification method to fit our EL environment without generating text.

Metrics. As described above, the task is to label ℓ new structures for each of s unexplained games. An EL algorithm addressing this task encodes the

predicted rule as an ℓ -dimensional binary vector \mathbf{v} per game (predicted vector), where $v_i = 1$ means that the i -th structure satisfies the predicted rule, and $v_i = 0$ otherwise (see fig. 3.4). Let \mathbf{w}^* be the ground-truth vector, obtained by tagging the ℓ structures according to the correct secret rule. Then, the Hamming distance $d_H(\mathbf{v}, \mathbf{w}^*)$ measures the number of wrong tags assigned by the EL algorithm; if $d_H(\mathbf{v}, \mathbf{w}^*) < d_H(\mathbf{v}, \mathbf{w}_i)$, where $\mathbf{w}_i \neq \mathbf{w}^*$ ranges over all the possible rules, then we deem the solution \mathbf{v} predicted by the algorithm to be correct. Thus, we define the *Nearest Rule Score* (NRS) as the number of

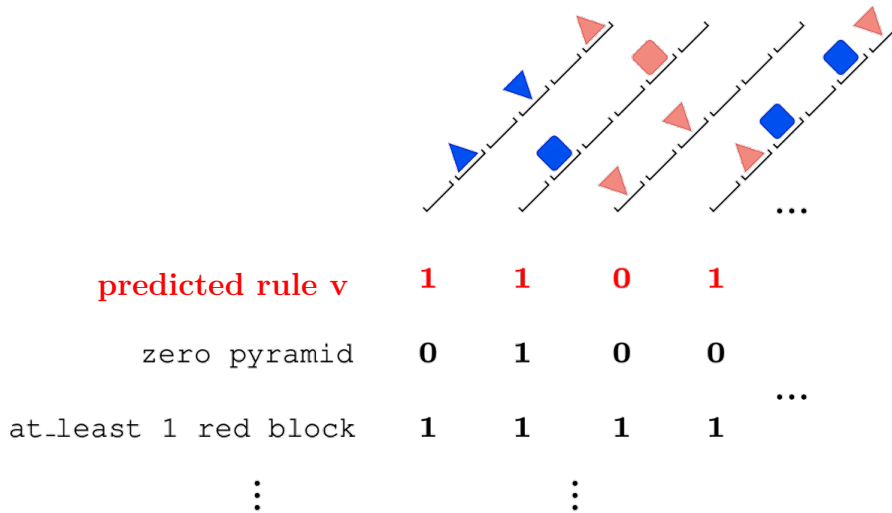


FIGURE 3.4: **Illustration of a prediction vector for Odeen.**
 For each structure s_i in the game, we have that \mathbf{v}_i indicates the label predicted by the EL algorithm being tested.

correctly predicted rules over a total of s games. A second score, the *Tagging Accuracy* (T-Acc), directly counts the number of correct labels averaged over s games.

3.3.1 Dataset

Having described the components of the Odeen universe, we propose a dataset that can be used to train and test possible EL algorithms. This dataset is composed of several training sets—each including different number of training rules and structures—and a single test set.

Odeen structures are sequences of six elements including spaces, blue or red circles, blue or red squares, blue or red pyramids—the latter either pointing up or down. The size of the universe is thus $|U| = 7^6 = 117,649$ possible structures. We further created a small language with objects, attributes, quantifiers, logical conjunctions, and interactions (e.g., “touching”). The grammar generates $\approx 25\text{k}$ valid rules in total. Each of the $|U|$ structures is tagged according to all the rules.

Training sets. The total number of rules produced by the Odeen grammar is 24,794. We consider training sets varying from 500 to 1438 rules. We choose these rules such that each token and each syntactic construct appears at least once; then, we uniformly select the others from the distribution. Each rule is associated with a set of 100, 1000, or 10000 labeled structures that unambiguously identify it. Thus, we have six possible training sets: $500r \times 100s$, $500r \times 1000s$, $500r \times 10000s$, $1438r \times 100s$, $1438r \times 1000s$, $1438r \times 10000s$.

NOTE:

We removed from the training set any rule containing the bigram `exactly 2`, as well as any rule of the form `at_least 2 X` and `at_most 2 X`, equivalent to `exactly 2 X`. This was purposely done to better test our models' generalization capabilities.

Test set. We generate the 1,132 games that compose the test set the same way. In the test set 72 rules contain the bigram `exactly 2`. Rules in the test set are associated with just 32 labeled structures. The first 10 structures are chosen by searching pairs of similar structures with different labels, following a common human strategy in Zendo. The remaining 22 structures are selected to ensure the lack of ambiguity on the board.

3.4 Critical Rationalist Networks

In principle, an EL problem like Odeen can be approached by training an end-to-end neural network to predict $\hat{y} = \mathbf{1}_{P_i}(x')$, given as input a set of observations D_i and a single sample x' (see fig. 3.3 C, left). Such a model would assume that all the information needed to solve the task is embedded in the data, ignoring the explanations; we refer to this as the “radical empiricist” approach [60]. A variant that includes the explanations in the pipeline can be done by adding a textual head to the network. This way, we expect performance to improve because predicting the explanation string can aid the classification task. As we show in the experiments, the latter approach (called “conscious empiricist”) indeed improves upon the former; yet, it is still a far cry from providing acceptable results.

We introduce a “rationalist” approach to solving EL problems in the following. This approach recognizes the given explanations as existing knowledge and focuses on interpreting them. Our *Critical Rationalist Networks* (CRNs) tackle the EL scientist problem introduced in section 3.2: to find $l = \mathbf{1}_{P_0}(s')$ given a structure s' , a D_0 , $\{D_1, \dots, D_n\}$, $\{e_1, \dots, e_n\}$. The way CRNs approach this task is by using two independently trained models:

Conjecture Generator (CG): This is a language model that can be used to sample a rule $r \in \Sigma$ given a batch of labeled structures $D = \{(s_j, l_j)\}_j$. Formally, we can say that

$$CG(D) \sim P_\theta(r|D) \sim P_\theta(r|\{(s_j, l_j)\}_j),$$

with θ the parameters of the language model. The idea behind this model is to synthesize plausible explanations (i.e. Odeen rules) for the given observations D .

Interpreter (\mathcal{I}) This model estimates whether a structure $s \in U$ is consistent with respect to a rule $r \in \Sigma$. Thus, we have that $\mathcal{I}(s, r) = \hat{l}$, with $\hat{l} \in \{0, 1\}$. As the name implies, the goal of this is *interpret* the semantics of a provided rule.

Algorithm 1 CRN inference procedure

Input: dataset D_0 , sample s'

Output: explanation \hat{e}_0 , prediction \hat{l}'

```

1: for  $t = 1 \dots T$  do
2:    $r_t \sim \mathcal{CG}(D_0)$  {Sample from language model}
3: end for
4:  $H(\cdot) \leftarrow \emptyset$  {Empty dictionary with 0 as default value for keys}
5: for  $t = 1 \dots T$  do
6:   for  $(s, l) \in D_0$  do
7:     if  $\mathcal{I}(r_t, s) = l$  then
8:        $H(r_t) \leftarrow H(r_t) + 1$  {Update hamming distance for rule  $r_t$ }
9:     end if
10:  end for
11: end for
12:  $\hat{e}_0 \leftarrow \arg \max_t H(r_t)$ 
13:  $\hat{l}' \leftarrow \mathcal{I}(\hat{e}_0, s')$ 
14: return  $\hat{e}_0, \hat{l}'$ 

```

At test time, we are given a trained \mathcal{CG} and a trained \mathcal{I} , and we must predict whether some $\hat{s} \notin D_0$ belongs to P_0 or not. Our approach is to generate t probable conjectures by sampling from $\mathcal{CG}(D_0)$ t times; then, each conjecture is verified by counting how many times the interpreter \mathcal{I} outputs a consistent prediction over D_0 . The conjecture with the highest hit rate is our candidate explanation \hat{r}_0 for P_0 . Finally, we obtain the prediction \hat{l}' as $\mathcal{I}(\hat{r}_0, s')$. See algorithm 1 for the step-by-step pseudo code.

The interpreter \mathcal{I} is a crucial component of our approach. A poor \mathcal{I} may fail to identify e_0 among the generated conjectures, or yield a wrong prediction l' when given the correct e_0 . On the other hand, the role of \mathcal{CG} is to trade off performance for computational cost; This is controlled by the parameter t , that is, the number of conjectures that we would like to sample from our learned distribution. Larger values for t imply more generated conjectures, corresponding to exhaustive search if taken to the limit. This potential asymmetry in quality between \mathcal{CG} and \mathcal{I} is intuitive, since the learning problem solved by \mathcal{CG} is generally harder.

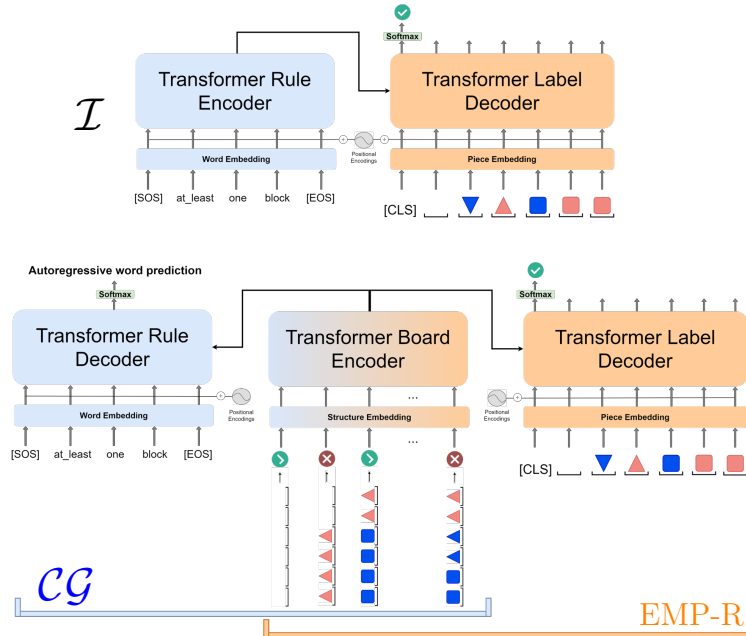


FIGURE 3.5: CRNs are implemented using encoder-decoder transformers blocks. **Top:** \mathcal{I} denotes the interpreter model (rule encoder and label decoder). **Bottom:** The conjecture generator \mathcal{CG} is composed by blue blocks. The “radical empiricist” (EMP-R) is composed of orange blocks. The “conscious empiricist” (EMP-C) baseline model consists of all the transformer blocks in the right-bottom figure, board encoder with rule and label decoders (all the blue and orange blocks).

Implementation. Figure 3.5 illustrates the architecture of CRNs, which we implement using the encoder-decoder transformer architecture [61]. The figure also shows the architecture of the baseline methods EMP-R and EMP-C, corresponding to the end-to-end NN model and its variant with a textual head, respectively.

3.5 Experimental Results

In this section, we compare our CRN approach to the radical (EMP-R) and conscious (EMP-C) empiricist models over the Odeen challenge, and analyze several fundamental aspects.

Generalization capabilities. The Odeen challenge addresses the generalization capability by asking for explanations for unexplained phenomena. This is evaluated over $s = 1132$ new games, where each game is given with $k = 32$ tagged structures (guaranteed to satisfy a unique, yet unknown rule) and requires to correctly tag $\ell = 1176$ unseen structures according to the unknown rule. The training set are $n = 1438$ games with ground-truth explanations and $m = 1000$ tagged structures per game. The test set does not include *any* rule equivalent to the training rules. One important example is the bigram “*exactly two*”, which appears in the test set, but was deliberately excluded from training; the training

rules only contain “*at least/most two*” and “*exactly one*”. The CRN guessed 40% of the 72 test rules with “*exactly two*”, while the empiricist models (EMP-C, EMP-R) scored 4% and 0% respectively.

Model	NRS	T-Accuracy	R-Accuracy
CRN	77.7%	98.0%	73.7%
Emp-C	22.5%	90.5%	3.5%
Emp-R	15.6%	89.8%	-

TABLE 3.1: **Evaluation results.** This table contains the results for the *CRN*, *radical empiricist*, and *conscious empiricist* models. As can be seen, our CRN approach yields much better results for the Odeen challenge.

The various models’ evaluation results can be seen in table 3.1. The NRS of 77.7% denotes that the CRN discovered the correct explanation for 880 out of 1132 new phenomena. Using the same data and a similar number of learnable parameters, the empiricist models score 22.5% at most. The R-Accuracy measures how frequently an output explanation is equivalent to the correct one; two rules A and B are equivalent if the tags assigned by the hard-coded interpreter to all the $\sim 117k$ structures in U are the same for A and B . As expected, the explanation predicted by the conscious empiricist model is rarely correct (R-Acc 3.5%), even when it tags some structures properly (NRS 22.5%); indeed, EMP-C gives no guarantee for the predicted explanation to be consistent with the tags prediction. Conversely, the CRN consistently provides the correct explanation when it is able to properly tag the new structures (NRS 77.7%, R-Acc 73.7%). The 4% gap between the two scores is clarified in the next paragraph.

Handling ambiguity and contradiction. One may reasonably expect that a CRN equipped with the ground-truth interpreter used to generate the dataset, would perform better than a CRN with a learned interpreter. Remarkably, this is not always the case, as reported in Table 3.2.

The better performance of the fully learned interpreter over the ground-truth one is due to its ability to process ill-formed conjectures generated by the \mathcal{CG} . The conjecture “at least one pointing up” makes the hard-coded interpreter fail, since “pointing up” must always follow the word “pyramid” by the grammar. Yet, in Odeen, pyramids are the only objects that point, and the learned \mathcal{I} interprets the conjecture correctly. Other examples include: “exactly one red block touching pyramid blue” (“pyramid” and “blue” are swapped), or the contradictory “at least one two pyramid pointing up and exactly one red pyramid”, which was interpreted correctly by ignoring the first “one”. When the learned interpreter is not very accurate, the negative effect of errors in tagging prevails.

Making sense out of ambiguous or contradictory messages³ is a crucial difference between a learned interpreter vs a hardcoded one. As [63] reminds us, a concept does not need to be precisely defined in order to be meaningful. Our

³This is one of seven essential abilities for intelligence as found in *GEB* [62, Introduction].

Train Data		NRS		T-Acc.
		Fully-learned CRN	Hardcoded \mathcal{I} CRN	Learned \mathcal{I}
10K struct.	1438 rules	0.813	0.801	0.997
1K struct.	1438 rules	0.777	0.754	1.000
100 struct.	1438 rules	0.402	0.406	0.987
10K struct.	500 rules	0.354	0.377	0.923
1K struct.	500 rules	0.319	0.336	0.924
100 struct.	500 rules	0.109	0.101	0.920

TABLE 3.2: **Explanatory Learning vs Program Synthesis paradigm.** Performance comparison of a data-driven vs ground-truth interpreter in a CRN. The last column shows the tag prediction accuracy of the learned \mathcal{I} , when provided with the correct rule.

everyday reasoning is not precise, yet it is effective. “After the small tower, turn right”; we will probably reach our destination, even when our best attempts at defining “tower”, as found, e.g., in the Cambridge dictionary, begin with “a *tall*, narrow structure...”.

Explainability. The predictions of a CRN are *directly caused* by a human understandable explanation that is available in the output; this makes CRNs explainable by construction. Further, CRNs allow counterfactuals; one may deliberately change the output explanation with a new one to obtain a new prediction. The bank ML algorithm spoke: “Loan denied”; explanation: “Two not paid loan in the past and resident in a district with a high rate of insolvents”. With a CRN, we can easily discard this explanation and compute a new prediction for just “Two not paid loan in the past”.

Importantly, by choosing a training set, we control the language used for explanations; i.e., we explicit the biases that will steer the learning of generalizations [64]. This allows a CRN to ignore undesirable patterns in the data (e.g., skin color) if these can not be expressed in the chosen language. If the Odeen training set had no rule with “pointing up/down”, the learned interpreter would see all equal pyramids, even with unbalanced training data where 90% of pyramids point up.

On the contrary, current explainability approaches for NNs (end-to-end empiricist models) either require some form of reverse engineering, e.g., by making sense out of neuron activations [65], or introduce an ad-hoc block to generate an explanation *given* the prediction, without establishing a cause-effect link between the two [66, 67]. This practice produces explanations that are not reliable and can be misleading [68], on the contrary CRNs’ explanations are faithful to what the model actually computes.

Prediction confidence. As explained in section 3.4, at test time the CRN selects the conjecture with the highest hit rate among the ones generated by the \mathcal{CG} . Alternatively, one may keep only the conjectures coherent with *all* the

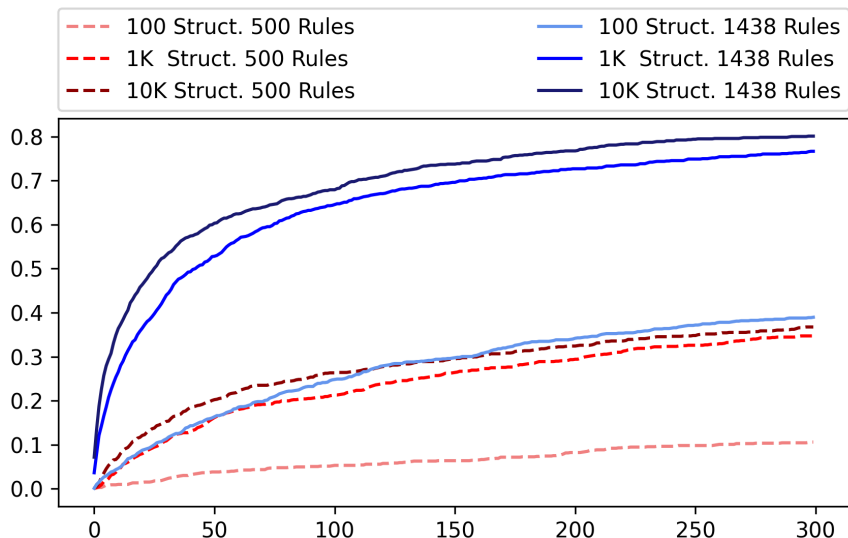


FIGURE 3.6: **Adjustable thinking time.** CRNs have a test-time parameter t , corresponding to the number of generated conjectures, which trades off computational cost for performance. In the inset, we plot the cumulative R-Acc score (y axis) against the number t of generated conjectures (x axis). The curves show that $> 60\%$ of correct explanations are found within the first 50 candidates, and $> 80\%$ are within the first 300. As a reference, a brute force exhaustive search would reach 100% over a search space of 24,794 possible explanations.

structures in the table, returning an “unknown explanation” signal if no such conjectures are found. If the interpreter is sufficiently accurate, this stricter condition barely deteriorates the CRN performance, and it will never return a prediction based on a possibly wrong explanation. For example, tested in a setting with $n = 1438$, $m = 1000$ (same as the *Generalization power* paragraph), this stricter CRN discovers the correct explanation for 861 out of 1132 new phenomena (76%), and admits its ignorance on the other 271. Conversely, evaluating the confidence of an end-to-end neural network remains an open problem [69].

3.6 Conclusions

Recently, the attention on the epistemological foundations of deep learning has been growing. The century-old debate between empiricists and rationalists about the source of knowledge persists, with two Turing prizes on opposite sides; [70] argues that empiricism still offers a fruitful research agenda for deep learning, while [60] supports a rationalist steering to embrace model-based science principles. This new debate is relevant, since as Pearl notes, today we can submit the balance between empiricism and innateness to experimental evaluation on digital machines.

Limitations and future directions. EL models the essential part of the knowledge acquisition process, namely the interval that turns a mute sequence of symbols into an explanation with reach. However, our modeling assumes a representative set of observations D_0 to be given (the $k = 32$ structures of the new phenomenon). A more comprehensive explanatory model would allow the player to do without these observations, and instead include an interaction phase with the environment where the D_0 itself is actively discovered. We see this as an exciting direction for follow-ups.

Finally, we expect CRNs to be more resilient than end-to-end models to adversarial attacks. For a given data point $x' \in P_0$ classified correctly by an empiricist model, a small adversarial change on D_0 can flip the prediction for x' while remaining unnoticed. Conversely, suppose that a CRN made the prediction for x' , and assume that the correct explanation was ranked as the 5th most likely by the \mathcal{CG} . The same attack on D_0 will have the effect of moving the correct explanation lower in the ranking; however, as long as it stays within the first t conjectures (300 in this paper), it will always be found by the interpreter as the correct solution.

Chapter 4

Beyond the Imitation Game: BIG-bench

As they scale, language models have shown significant quantitative improvements and new qualitative capabilities. Despite their potential to transform various fields, arguably these new capabilities remain poorly understood. To guide future research, prepare for disruptive advancements, and mitigate socially harmful effects, it is crucial to understand the current and near-future capabilities and limitations of language models. To tackle this challenge, the *Beyond the Imitation Game benchmark* [7]—also known as *BIG-bench*—was introduced. This benchmark consists of 204 tasks contributed by 450 authors from 132 institutions, covering a wide range of topics. These topics span linguistics, childhood development, mathematics, common-sense reasoning, biology, physics, social bias, software development, and more. BIG-bench focuses on tasks that are believed to be beyond the capabilities of current language models.

In this chapter, we will describe one of such tasks; the *Symbol Interpretation Task* (SIT). This task in particular was developed by us and included in the collaborative effort of BIG-bench. More information regarding this task, alongside the actual benchmark files, can be found at [this page](#).

NOTE:

BIG-bench was designed to encompass a large and diverse set of tasks, optionally supporting arbitrary programmatic tasks. This wide scope is a significant strength of BIG-bench. However, it also means that full evaluation can be computationally expensive. To mitigate this problem, a subset of 24 tasks (see table 4.1) for a lightweight evaluation set, known as *BIG-bench Lite* (BBL) has been selected by the BIG-bench core contributors. Amongst these, our Symbol Interpretation Task was also chosen, showing further interest from the research community.

4.1 Symbol Interpretation Task

The *Symbol Interpretation Task* (SIT) was partially inspired by our Odeen dataset, previously described in section 3.3. This task asks language models to choose the sentence—amongst a given set—consistent with two observed

Big-bench Lite Tasks

auto_debugging	logical_deduction
bbq_lite_json	misconceptions_russian
code_line_description	novel_concepts
conceptual_combinations	operators
conlang_translation	parsinlu_reading_comprehension
emoji_movie	play_dialog_same_or_different
formal_fallacies_...	repeat_copy_logic
hindu_knowledge	strange_stories
known_unknowns	strategyqa
language_identification	symbol_interpretation_task
linguistics_puzzles	vitaminc_fact_verification
logic_grid_puzzle	winowhy


TABLE 4.1: **BIG-bench Lite tasks in alphabetical order.**

This is a diverse subset of JSON tasks that can be cheaply evaluated by most language models. This subset includes our *Symbol Interpretation Task*.

structures, where a structure is a sequence of six pieces represented by emojis. This task is composed of five similar smaller tasks that require interpreting statements referring to structures of an input simple world. This world is built using emojis; a structure is represented as a sequence of six emojis. Crucially, in every variation, we make explicit the semantic link between the emojis and their respective names. Some examples of this mapping can be seen in table 4.2.

As previously mentioned, SIT is composed of five different sub-tasks that try to measure slightly different things. These are named *Plain*, *Adversarial*, *Tricky*, *Agnostic name-side*, and *Agnostic emoji-side*.

Plain: This can be considered as our baseline task. It is the simpler among all other sub-tasks, thus it is expected for language models to perform better at it. Each example in this sub-task is constituted by three sections; (i) A description of the SIT environment, describing how structures of pieces are made. (ii) Two structures, composed of a sequence of six symbols (emojis) each. (iii) A collection of five sentences expressed in natural language, such that only one of them is both consistent with the first structure and not consistent with the other. See fig. 4.1 for an example.

Agnostic name-side: In this sub-tasks, instead of using natural language names to refer to symbols, we use simple sequences of arbitrary letters. An example would be “ is a X Y”. The motivation behind these changes is to test the ability to use arbitrary placeholders rather than meaningful words.

Agnostic emoji-side: This sub-task is symmetric to the *Agnostic name-side* one. Indeed, instead of mapping shape emojis to an arbitrary name, we map arbitrary emojis to their descriptive name. An example of this

mapping would be “🟡 is a red square”. The motivation behind these changes is to test the ability to ground meaningful words to unrelated world objects.

Adversarial: This is a similar sub-task to the previous one, with the difference being that instead of mapping arbitrary emojis to a “common-sense” name, we purposely select shape emojis that might be confusing for the model. For example, “🟡 is a red square”. The motivation behind these changes is to test the ability of language models to remap the “common-sense” word-meaning association presumably seen at training.

Tricky: This sub-task is similar to the *Plain* sub-task. However, instead of using the “common-sense” name, we use its reversed string. An example of this emoji-to-name mapping is “🟡 is a der reauqs”. The motivation behind these changes is to test the ability to compose capabilities (i.e., work with words in reverse order).

Sub-Task	Mapping Example
Plain	“🟡 is a red square”
Adversarial	“🟡 is a red square”
Tricky	“🟡 is a der reauqs”
Agnostic name-side	“🟡 is a X Y”
Agnostic emoji-side	“🟡 is a red square”

TABLE 4.2: Examples of the semantic mapping between symbols—emojis in our case—and their names. Each sub-task tries to measure different capabilities from the language models.

What is SIT trying to measure? The task is trying to measure the ability of the language model to reason and interpret a simple scene described solely in natural language. The model has to ground the observations in natural language and reason about the relationships between the objects in the scene. In particular, we speculate that for language models to successfully solve these tasks, they require the combination of several key abilities:

Domain separation of text tokens: The ability to treat text tokens in two fundamentally different ways, as language tokens or as references to abstract objects. Thus, the language model should be able to use a certain level of abstraction.

Language grounding: Ground language tokens to objects of a hypothetical world. The ability to parse and then immediately use descriptions of new objects, i.e., assimilate semantic maps at inference time. Invariance to wrong object symbols and/or nonsensical object names, which again indicates a sort of abstract understanding of the meaning convention underlying language.

SIT-plain example

In the SIT-plain world a structure is a sequence of six emojis. Hereafter are reported the emojis used along with their descriptions.

-  is a red circle;
-  is a blue circle;
-  is a yellow circle;
-  is a red triangle pointing up;
-  is a red triangle pointing down;
-  is a red square;
-  is a blue square;
-  is a yellow square;
- `_` is an empty space.

Choose the sentence consistent with the structure  and not consistent with the structure .

- There are zero yellow pieces.
- There is exactly one blue piece.
- There is at most one yellow piece.
- There is exactly one red square.
- There are at most two yellow squares.

FIGURE 4.1: **Instance of a Plain sub-task example.** For illustration purposes, the correct answer has been crossed in this case.

Perception: Perceive objects in the scene, alongside their qualities.

- (i) *Object identification:* Identify the types of pieces, i.e. whether it is a square, a circle, or a triangle.
- (ii) *Attribute identification:* Identify the various attributes of pieces. This means understanding which color each piece has and -optionally for triangles- their orientation.

Reasoning: To adequately solve these tasks the evaluated language models should be able to perform some basic reasoning operation about the objects in the scene. In particular, we identified that LLMs should be able to;

- (i) *Counting:* quantify various types of pieces in the given structures. For example, understand whether there are one, two, or more blue circles.
- (ii) *Relational reasoning:* perform reasoning about positional relationships between pieces (at the right of, at the left of, touching, surrounded by) and use their common-sense meaning in the SIT context.
- (iii) *Logical reasoning:* evaluate simple logic operations like “and/or” in the properties that are being assessed.

We speculate that all the aforementioned abilities are required to give an acceptable solution to the SIT benchmark.

Sentence groups. There are three sentence groups of increasing difficulty for each subtask:

- The first group comprises 66 examples with simple *quantification sentences* (e.g., “*There is exactly one blue circle*”). We expect these examples to be the easiest to solve among the ones in the task since they require only the capability to count the number of elements in the observed structures.
- The second group of 66 examples contains simple sentences connected with *logical operators*. An instance of these types of sentences would be “*There are at least two yellow squares and exactly one blue circle*”. These pose a slightly more difficult challenge since they require the language model to apply boolean reasoning on top of counting and grounding.
- Finally, the last 66 examples use sentences expressing *positional relationships* between pieces in the structure (e.g., “*There is exactly one triangle at the right of a yellow circle*”).

This was done to possibly have a more fine-grained understanding of the results from the benchmark. Indeed, by only testing some groups, it would be possible to extrapolate more information about the tested LLM (e.g. whether it is able to count, apply logical operators, or have an understanding of spatiality in the structures).

Tricky vs Agnostic name-side. The subtask SIT-tricky may seem not so different from SIT agnostic name-side at first sight. However, the main difference is that the tricky subtask contains more information that an agent could exploit to solve it. In particular, in the tricky subtask, the text is the same as SIT-plain but in reverse order (e.g., “red square” -> “der erauqs”). Once an agent understands how to reverse words (GPT-3, for example, seems able to generate anagrams and reverse words), it could use the reverse function and the same knowledge learned in solving SIT-plain to solve SIT-tricky. In the name-agnostic subtask, words are entirely unrelated. We think that the SIT-tricky subtask introduces interesting variations like the ability to compose capabilities; it could be interesting to see the comparison between SIT-tricky and SIT agnostic name-side.

To solve the SIT sub-tasks, humans tend to implement the following strategy, which starts from the answers rather than the question: for each possible choice, check if it is consistent with the first structure and **not** consistent with the second structure. If this is the case, the choice is the correct one. Implementing this strategy could be challenging for a language model since it has to pay attention to each choice separately and, for each choice, test the consistency and **not** consistency of the structures in the question.

4.2 Experimental Result

For each sub-task, we computed the accuracy on several models. In particular, we tested on GPT-2, RoBERTa, and BART (the last two fine-tuned on MultiNLI). Given a SIT example, we compute the accuracy using the following procedure: the model’s probability is predicted for each of the possible five sentences. The score is then 1 if the predicted sentence is consistent with the input structures otherwise the score is equal to 0. The final sub-task score is then the average across all examples in the sub-task. The results are presented in table 4.3.

Model	Plain	Adversarial	Tricky	Agnostic-name	Agnostic-emoji
GPT-2	15.1%	17.6%	20.2%	16.6%	17.6%
RoBERTa	14.1%	15.6%	19.1%	18.1%	14.6%
BART	20.2%	23.2%	21.2%	21.2%	23.2%
Random	20.0%	20.0%	20.0%	20.0%	20.0%

TABLE 4.3: **Multiple-choice accuracy for different LLMs.**

This table reports the multiple-choice accuracy evaluated on different LLM architectures. The results for all the models we tested are similar to the random guess accuracy.

NOTE:

We investigated the byte-level Byte-Pair Encoding (BPE) tokenization [71] capabilities of GPT-2, RoBERTa, and BART. We assessed that emojis are tokenized correctly in all the sub-tasks.

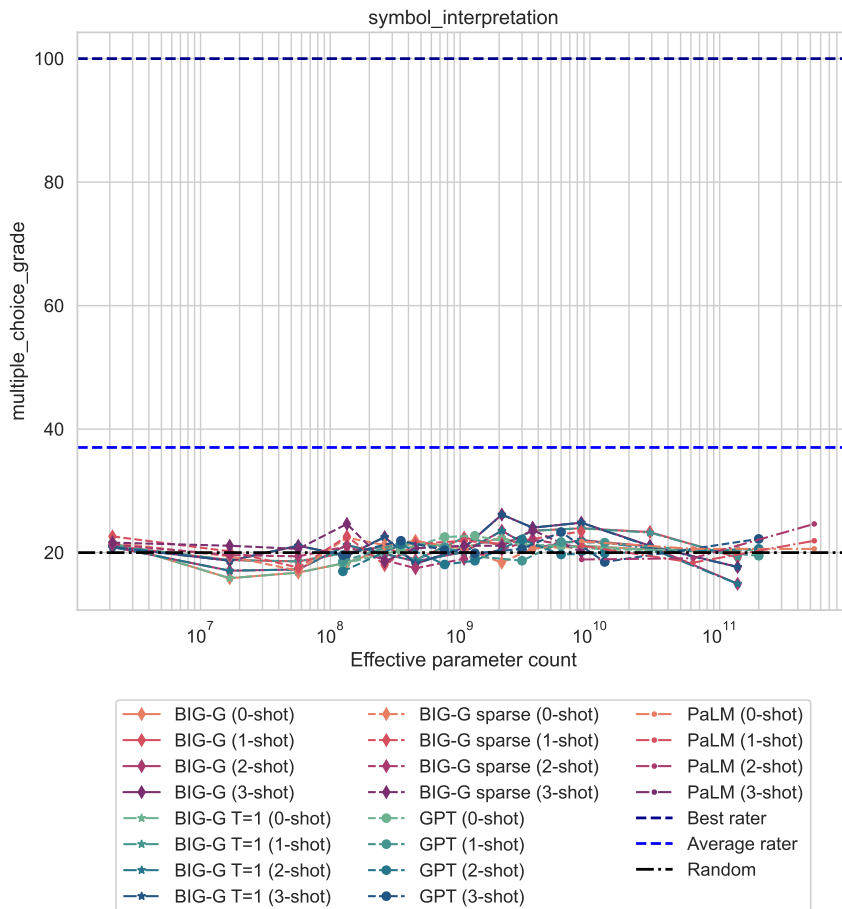


FIGURE 4.2: **Overall multiple-choice accuracy as the number of parameters grows.** This plot shows the performance of various LLMs on the SIT task. Performance is computed as the average of the multiple-choice accuracy over all sub-tasks. Note that the human baseline significantly surpasses the models’ performance. This is especially true if we consider the best human rater. All tested models don’t diverge much from the random guess baseline, thus suggesting that this task is good for assessing the capabilities of future LLMs.

4.3 Limitations and future work

A perfect score on this task does not imply a general ability to reason about real-world problems, objects, and relationships between objects. However, it might help in designing models in this direction. Solving this task is a necessary but not sufficient condition for tackling general reasoning since the model could have applied a heuristic unknown to us to solve it.

Rephrasing the task with a free-text answer instead of multi-choice could be a promising research area for future work. In this case, the task is much harder for the model (conjecture the hidden sentence), but also more difficult to evaluate quantitatively.

Part II

Generative modeling in the Signal domain

Chapter 5

Latent Autoregressive Source Separation

Autoregressive models have made remarkable strides across various domains, demonstrating exceptional generation quality and performance in downstream tasks. In continuous domains, an important element of this success is the implementation of quantized latent spaces¹. These quantized spaces allow for dimensionality reduction and faster inference times, thus improving the efficiency and effectiveness of these models. However, leveraging existing pre-trained models for novel tasks often presents challenges, as it might necessitate fine-tuning or similar approaches.

This chapter introduces LASS, an approach to vector-quantized Latent Autoregressive Source Separation. LASS aims to de-mix an input signal into its constituent sources without the need for further gradient-based optimization or modifications to existing models. Our method relies on a Bayesian framework with autoregressive models serving as priors. For the likelihood function, we construct a discrete (and sparse) mapping by performing frequency counts over latent sums of addend tokens. We evaluate our approach using both images and audio, exploring various sampling strategies. Our results demonstrate that LASS not only competes effectively with existing separation methods in terms of quality but also offers substantial improvements in inference time and scalability to higher-dimensional data.

5.1 Introduction

Autoregressive models have achieved impressive results in a plethora of domains ranging from natural language [72] to densely-valued domains such as audio [73] and vision [15, 74], including multimodal joint spaces [75, 76]. 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 [14]. This way, generating higher resolution samples while simultaneously reducing inference time is possible. Additionally, the learned latent representations are useful for downstream tasks [77]. However, in order to perform new non-trivial tasks, the standard practice is to fine-tune the model or, in alternative, elicit prompting by scaling training [78, 79]. The former is usually

¹such as those obtained via VQ-VAE autoencoders

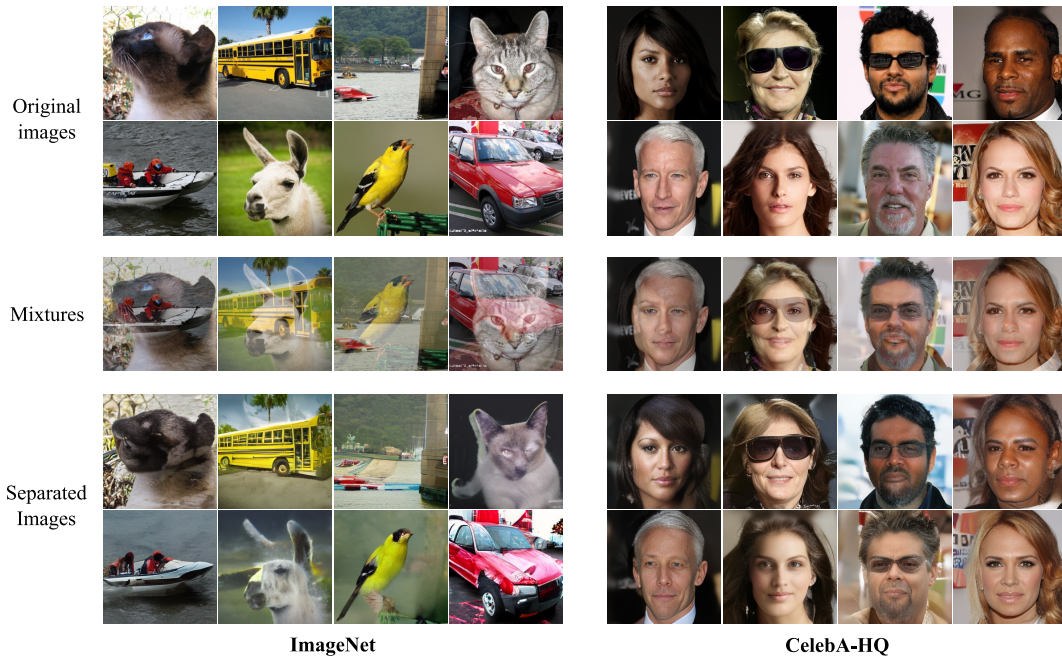


FIGURE 5.1: **256x256 separations obtained with LASS.** To perform these image separations, we used pre-trained autoregressive models. Left: class-conditional ImageNet. Right: unconditional CelebA-HQ.

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 [80, 81].

Our work aims at tackling one of such tasks, namely *source separation*, leveraging existing vector-quantized autoregressive models without requiring any gradient-based optimization or architectural modifications. The task of separating two or more sources from a mixture signal has recently received much attention following the success of deep learning, especially in the audio domain, ranging from speech [82], music [83], and universal source separation [84, 85]. Although not as prominent as its audio counterpart, image source separation has been addressed in literature [86]. Most successful approaches use explicit supervision to achieve notable results [87, 88], or leverage large-scale unsupervised regression [89].

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). 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 LASS (*Latent Autoregressive Source Separation*).

We can summarize our contributions as follows:

- (i) We present LASS as a Bayesian inference technique for source separation, capable of utilizing existing pre-trained autoregressive models within quantized latent spaces.
- (ii) We experiment with LASS in the image domain and demonstrate competitive results at a significantly lower inference time cost compared to competitors on MNIST and CelebA (32×32). We also present qualitative results on ImageNet (256×256) and CelebA-HQ (256×256), highlighting LASS’s scalability with pre-trained models. To our knowledge, this is the first method to extend generative source separation to higher resolution images.
- (iii) We experiment with LASS in the music source separation task on the Slakh2100 dataset. 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.

5.2 Related Work

With the advent of deep learning, most prominent methods for source separation can be classified as regression-based or generative-based methods. The problem of source separation has traditionally been addressed in an unsupervised manner, often referred to as *blind source separation* [90–93]. In this context, no information is available about the sources that need to be separated from a mixture signal. Consequently, these methods rely on broad mathematical priors, such as source independence [91] or repetition [94], to achieve separation. With the advent of deep learning, the most prominent methods for source separation can now be classified as either regression-based or generative-based approaches.

Regression-based source separation. In this setting, a mixture is fed to a parametric model (i.e., 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., \mathcal{L}_1 or \mathcal{L}_2) [95]. Supervised regression has been applied to image source separation [86], but it has been mainly investigated in the audio domain, where 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 STFT domain [96–101]. 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 [87, 88, 102].

Generative source separation. Following the success of deep generative models [13, 27, 28, 103], a new class of generative source separation methods is gaining prominence. This setting emphasizes the exploitation of broad generative models (especially pre-trained ones) to solve the separation task without needing a specialized architecture (as with regression-based models).

Following early work on deep generative separation based on GANs [104–106], [107] propose the generative separation method BASIS in the image setting using score-based models [26] (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 [108], obtaining good quantitative and qualitative results. The authors extend the Langevin dynamics inference procedure to autoregressive models by re-training them with a noise schedule, introducing the Parallel and Flexible (PnF) method [109]. 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 paper, 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.

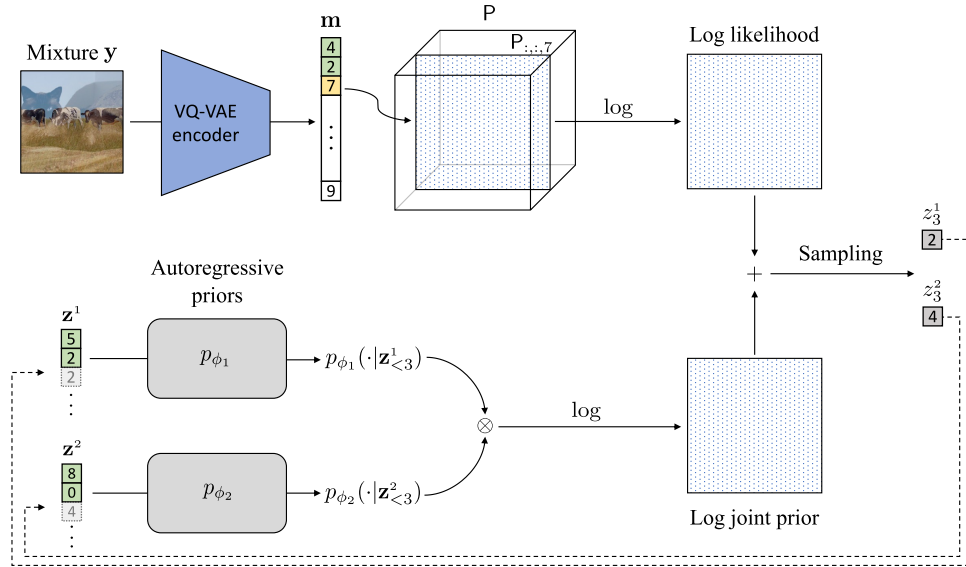


FIGURE 5.2: **Schematic of the LASS separation procedure.**

Illustration of 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 \mathbf{z}^1 and \mathbf{z}^2 via the VQ-VAE decoder (not depicted in the picture).

We refer the reader to algorithm 2 for more details.

5.3 Method

Let $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2) \in \mathbb{R}^{2 \times N}$ denote two sources distributed according to $p_{\text{data}} = (p_{\text{data}}^1, p_{\text{data}}^2)$ and $\mathbf{y} = (\mathbf{x}^1 + \mathbf{x}^2)/2$ an observable mixture. The goal of generative source separation is to estimate the sources \mathbf{x} given the mixture \mathbf{y} , using the Bayesian posterior (assuming independent sources):

$$p(\mathbf{x}^1, \mathbf{x}^2 | \mathbf{y}) \propto p_{\text{data}}^1(\mathbf{x}^1) p_{\text{data}}^2(\mathbf{x}^2) p(\mathbf{y} | \mathbf{x}^1, \mathbf{x}^2). \quad (5.1)$$

Working directly with Eq. (5.1) 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 subsequently 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.

5.3.1 Latent Autoregressive Source Separation

This paper explores the case in which p_{data} is estimated by a unique autoregressive model p_ϕ for all the sources (unsupervised²) 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 $\mathbf{z} = (\mathbf{z}^1, \mathbf{z}^2) = B(E_\theta(\mathbf{x}))$ and $\mathbf{m} = B(E_\theta(\mathbf{y}))$. The posterior distribution in Eq. (5.1) can be locally expressed in the latent domain as:

$$p(\mathbf{z}_s | \mathbf{z}_{<s}, \mathbf{m}_{\leq s}) \propto p_\phi(\mathbf{z}_s | \mathbf{z}_{<s}) p(\mathbf{m}_{\leq s} | \mathbf{z}_{\leq s}), \quad (5.2)$$

for all $s = 1, \dots, S$. The first factor is the (joint) Bayesian prior, modeled with autoregressive distributions. The second factor is the likelihood function, which quantifies the likelihood of the sequences $\mathbf{z}_{<s}^1, \mathbf{z}_{<s}^2$ to combine into $\mathbf{m}_{\leq s}$.

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. (5.2), approximating it as:

$$p(\mathbf{m}_{\leq s} | \mathbf{z}_{\leq s}) \approx p(m_s | \mathbf{z}_s). \quad (5.3)$$

²Not to be confused with the unsupervised blind setting, i.e., in our unsupervised setting we have access to sources but we do not have class labels.

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

$$p(m_s | \mathbf{z}_s) = p(m_s | z_s^1, z_s^2) = p(m | z^1, z^2). \quad (5.4)$$

The following subsection describes how LASS models the likelihood function.

5.3.2 Discrete Likelihoods for Source Separation

Previous works in generative source separation [107, 109] 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 z_s^1 and z_s^2 (or the associate dense codes $\mathbf{e}_{z_s^1}$ and $\mathbf{e}_{z_s^2}$) 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. (5.4) using discrete conditionals, represented with rank-3 tensors³ $\mathbf{L} \in \mathbb{R}^{K \times K \times K}$:

$$p(\cdot | z^1, z^2) = \mathbf{L}_{z^1, z^2, \cdot}.$$

In order to learn \mathbf{L} , 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 $\mathbf{F}^0 \in \mathbb{N}^{K \times K \times K}$. 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 $\mathbf{z}^1 = B(E_\theta(\mathbf{x}^1))$, $\mathbf{z}^2 = B(E_\theta(\mathbf{x}^2))$ and $\mathbf{m} = B(E_\theta(\mathbf{y}))$. For each entry $(z_s^1, z_s^2, m_s) \in (\mathbf{z}^1, \mathbf{z}^2, \mathbf{m})$, at step t , we simply increment the previous count by one:

$$\begin{aligned} \mathbf{F}_{z_s^1, z_s^2, m_s}^t &= \mathbf{F}_{z_s^1, z_s^2, m_s}^{t-1} + 1, \\ \mathbf{F}_{z_s^2, z_s^1, m_s}^t &= \mathbf{F}_{z_s^2, z_s^1, m_s}^{t-1} + 1. \end{aligned}$$

We permute the order of the addends in order to enforce the commutative property of the sum. After performing the statistics, we can define \mathbf{L} as:

$$\begin{aligned} \mathbf{L}_{z^1, z^2, \cdot} &= \frac{1}{\sum_{k=1}^K \mathbf{F}_{z^1, z^2, k}} \mathbf{F}_{z^1, z^2, \cdot} \\ Z_{\text{mix}} &= Z_1 + Z_2 \end{aligned}$$

At inference time, the likelihood function (parametric in z^1 and z^2 , with m fixed) can be obtained by slicing the tensor along m , namely:

$$p(m | \cdot, \cdot) = \mathbf{L}_{\cdot, \cdot, m}.$$

At first glance, modeling the conditional distributions without parameters could seem memory inefficient, with a complexity of $O(K^3)$. In practice, the

³We follow the notation for tensors as in [110].

Algorithm 2 LASS inference**Input:** \mathbf{y} **Output:** $\mathbf{x}^1, \mathbf{x}^2$

```

1:  $\mathbf{m} \leftarrow B(E_\theta(\mathbf{y}))$ 
2:  $\mathbf{z}^1 \leftarrow []$ 
3:  $\mathbf{z}^2 \leftarrow []$ 
4: for  $s = 1$  to  $S$  do
5:   prior  $\leftarrow \log(p_{\phi_1}(\cdot | \mathbf{z}^1) \otimes p_{\phi_2}(\cdot | \mathbf{z}^2))$ 
6:   likelihood  $\leftarrow \log(\mathbf{L}_{[:, :, m_s]})$ 
7:   posterior  $\leftarrow$  prior  $+ \lambda$  likelihood
8:    $(z_s^1, z_s^2) \leftarrow \text{Sampler}(\text{posterior})$ 
9:    $\mathbf{z}^1 \leftarrow \text{concat}(\mathbf{z}^1, z_s^1)$ 
10:   $\mathbf{z}^2 \leftarrow \text{concat}(\mathbf{z}^2, z_s^2)$ 
11: end for
12:  $\mathbf{x}^1 \leftarrow D_\psi(\mathbf{z}^1)$ 
13:  $\mathbf{x}^2 \leftarrow D_\psi(\mathbf{z}^2)$ 
14: return  $\mathbf{x}^1, \mathbf{x}^2$ 

```

tensor \mathbf{L} is *highly sparse*. We showcase this in table 5.1 for all our experiments, where the density of \mathbf{L} is defined as the percentage of nonzero elements in \mathbf{L} .

NOTE:

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.

5.3.3 Inference Procedure

Given an observable mixture \mathbf{y} , the autoregressive priors p_{ϕ_1}, p_{ϕ_2} and the estimated likelihood tensor \mathbf{L} , it is possible to perform inference and estimate $\mathbf{x}^1, \mathbf{x}^2$, as described in Algorithm 2 and depicted in Figure 5.2.

We start by mapping \mathbf{y} to the latent domain obtaining $\mathbf{m} = B(E_\theta(\mathbf{y}))$ and initializing the estimates $\mathbf{z}^1, \mathbf{z}^2$ with the empty sequences. The algorithm iterates over $s = 1, \dots, S$. 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 $\mathbf{L}_{[:, :, m_s]}$. 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 (z_s^1, z_s^2) from the posterior. In our experiments, we used ancestral sampling (with and without top- k filtering) and beam search. After

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

TABLE 5.1: **Statistics on likelihood functions over different datasets.** K is the number of VQ-VAE latent codes. Likelihood density is the percentage of nonzero elements in the likelihood tensor L .

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 \mathbf{x}^1 and \mathbf{x}^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:

$$\mathbf{e}_{t+1}^1 = \mathbf{e}_t^1 + \alpha \nabla_{\mathbf{e}_t^1} \|D_\psi(\mathbf{e}_t^1) + D_\psi(\mathbf{e}_t^2) - 2\mathbf{y}\|_2 \quad (5.5)$$

$$\mathbf{e}_{t+1}^2 = \mathbf{e}_t^2 + \alpha \nabla_{\mathbf{e}_t^2} \|D_\psi(\mathbf{e}_t^1) + D_\psi(\mathbf{e}_t^2) - 2\mathbf{y}\|_2 \quad (5.6)$$

for $t = 1, \dots, T - 1$ and $\mathbf{e}_1^1 = E_\theta(\mathbf{x}^1)$, $\mathbf{e}_1^2 = E_\theta(\mathbf{x}^2)$. 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 2. We found this strategy particularly helpful on the MNIST dataset, where we assess the quality of the separation through a pixel-wise metric (PSNR) and the VQ-VAE tends to produce smooth images.

5.4 Experimental Results

We performed quantitative and qualitative experiments on various datasets to demonstrate the efficacy and scalability of *LASS*. In the image domain, we evaluate on MNIST [35] and CelebA (32×32) [36] and present qualitative results on the higher resolution datasets CelebA-HQ (256×256) [37] and ImageNet (256×256) [38]. In the audio domain, we test on Slakh2100 [111], 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 listed on the companion website⁴.

5.4.1 Image Source Separation

We choose the Transformer architecture [16] as the autoregressive backbone for all image source separation experiments. With MNIST and CelebA, we first

⁴github.com/gladia-research-group/latent-autoregressive-source-separation

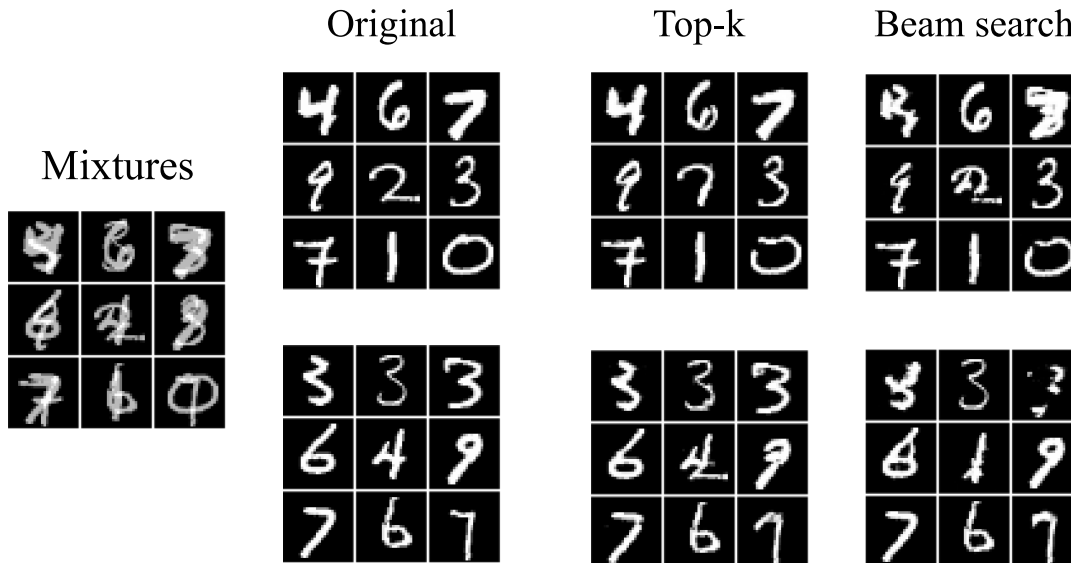


FIGURE 5.3: **Results on MNIST with top- k sampling ($k = 32$) over a random batch of examples.** Top- k sampling produces more defined digits, in agreement with the results in table 5.3.

Separation Method	MNIST (PSNR) \uparrow	CelebA (FID) \downarrow
Average	14.9	15.19
NMF	9.4	-
S-D	18.5	-
BASIS Glow	22.7	-
BASIS NCSN	29.3	7.55
LASS (Ours)	24.2	8.96

TABLE 5.2: **Comparison with other methods.** We evaluate LASS on both the MNIST and CelebA test sets. Results are reported in PSNR (higher is better) and FID (lower is better).

train a VQ-VAE, then train the autoregressive 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 [15] alongside the pre-trained Transformers published by the authors⁵ (`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 L tensor using the procedure presented in the section “Method”. As shown in table 5.1, 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.

⁵github.com/CompVis/taming-transformers

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 [107], the GAN-based “S-D” method [105], 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 PSNR (Peak Signal to Noise Ratio) [112]. We follow the experimental procedure of [107] 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 5.3 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 5.3). 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 \mathcal{L}_2 distance), and finally perform the refinement procedure in Eqs. (5.5), (5.6) with $T = 500$ and $\alpha = 0.1$. Evaluation metrics on this experiment are shown in table 5.2, while inference time is reported in table 5.4. 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 [26] on CelebA. In this case, we evaluate against the FID metric [113] instead of PSNR, given that for datasets that feature more variability than MNIST, source separation can be an underdetermined task [107]: 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 VQ-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 a known fact in the literature that score-based models outperform autoregressive models on FID metrics [114] on different datasets, yet our method paired with an autoregressive model shows competitive results with respect to the score-based “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 \mathbf{L} is learned. We showcase a curated results list in Figure 5.1 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

Sampling Method	MNIST (PSNR)	Slakh (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 5.3: **LASS performance using different sampling strategies.** On MNIST, the reported score is PSNR (dB) (higher is better), while on Slakh 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 \mathcal{L}_2 distance to the input mixture.

	Method	Time
MNIST	LASS (Ours)	$4.49 \text{ s} \pm 0.27 \text{ s}$
	BASIS NCSN	$53.34 \text{ s} \pm 0.51 \text{ s}$
Slakh	LASS (Ours)	$1.33 \text{ min} \pm 0.87 \text{ s}$
	PnF	$42.29 \text{ min} \pm 1.08 \text{ s}$

TABLE 5.4: **Inference speed comparisons for performing one source separation.** To estimate variance, we repeat inference 10 times on MNIST and 3 times on Slakh. We consider 3-second-long mixtures on Slakh.

autoregressive models without re-training (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.

5.4.2 Music Source Separation

We perform experiments on the Slakh2100 dataset [111] 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 [73] 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 5.1). We compare LASS to a set of unsupervised blind source separation methods -“rPCA” [92], “ICA” [91], “HPSS” [94], “FT2D” [115] - and to two supervised baselines Demucs [88] and Conv-Tasnet [87] using the SDR (dB) evaluation metric computed with the `museval` library [116]. To evaluate the methods,

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

TABLE 5.5: **Comparison with other source separation methods on Slakh (“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.

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 5.3), using $B = 100$ beams. Evaluation results are reported in table 5.5: 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” [109] by evaluating the time required to separate a mixture of 3 seconds sampled at 22 kHz (piano vs. voice on “PnF”). Results in table 5.4 show that LASS is significantly faster, and as such, it can be adopted in more realistic inference scenarios.

5.5 Limitations

In this work, we limit our analysis to the separation of two sources. Even if this is a common setup especially in image separation [86, 109], 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 [117].

Another limitation of the proposed method is the locality assumption taken in eq. (5.3). 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) [118]. Adopting a VQ-VAE quantized with respect to the latent channels [119] 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 paper.

5.6 Conclusion

In this chapter, we introduced LASS as a source separation method for latent autoregressive models. Thanks to its unintrusive approach it does not modify the structure of the priors, allowing the utilization of pretrained latent space autoregressive models. 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 our competitors. We believe our method will benefit from the improved quality of newer autoregressive models, improving both the quantitative metrics and the perceptive results.

Chapter 6

Diffusion Models for Multi-Source Music Generation

In this chapter, we introduce a series of diffusion-based generative architectures intended to allow a more instrument-aware and compositional control over music generation. This is important if we are interested in utilizing generative models as useful tools in an audio-processing and/or music-producing pipeline. In particular, our research effort resulted in three main works. These—in chronological order of development—are:

Multi-Source Diffusion Models (MSDM): This is the first audio diffusion model we designed: it is able to take both music synthesis and source separation by learning the score of the joint probability density of sources sharing a context. In addition to classic total inference tasks, such as generating a mixture and separating sources, we also explore the partial generation (or *accompaniment generation*) task, where a subset of the sources is generated given the others. An example would be the generation of a piano track that complements an existing drum track.

Generalized Multi-Source Inference (GMSI): This work is a generalization of our previous MSDM approach to arbitrary time-domain and text-conditioned diffusion models. These models do not require separated data as they are trained on mixtures, and can parameterize an arbitrary number of sources, thus giving a hypothetical user extensive control. We propose an inference procedure enabling the coherent generation of sources and accompaniments.

CompoNet: This is a diffusion architecture based on ControlNet [120], which allows the performing of several types of generative tasks through a new fine-training procedure, unifying several compositional models (MSDM, GMSDI, StemGen [121], and InstructME [122]). Beyond the flexibility offered by the ability to perform numerous compositional tasks, our model is the first to manage stems of the same type in the same track and introduces semantic control at the stem level.

6.1 Introduction

Generative models have recently gained a lot of attention thanks to their successful application in many fields, such as NLP [123, 124], image synthesis [125, 126]

or protein design [127]. The audio domain is no exception to this trend [18, 128]. Indeed, the task of automatic music production has seen significant advancements thanks to recent developments in generative AI. In particular, the families of generative models showcasing state-of-the-art music synthesis are latent language models [129] and diffusion models [26, 28, 130]. Latent language models map a continuous-domain (time or spectral) signal to a sequence of discrete tokens and estimate a density over such sequences autoregressively [18, 131] or via mask-modeling [132]. Diffusion models [128, 133], on the other hand, operate on continuous¹ representations directly, capturing the gradient of the log-density perturbed by a Gaussian process. Despite differences between these generative models, they typically share some mechanisms for conditioning on rich textual embeddings, obtained either using text-only encoders [134] or audio-text contrastive encoders [135–137]. Such a mechanism allows generating a musical track following a natural language prompt.

Multi-source coherency. A peculiarity of the audio domain is that an audio sample \mathbf{y} can be seen as the sum of multiple—usually coherent—individual sources $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, resulting in a mixture $\mathbf{y} = \sum_{n=1}^N \mathbf{x}_n$. Indeed, unlike in other sub-fields of the audio domain (like speech), musical sources² present in musical mixtures share a *context* given their strong interdependence. For example, the bass line of a song follows the drum’s rhythm and harmonizes with the melody of the guitar. Mathematically, this fact can be expressed by saying that the joint distribution of the sources $p(\mathbf{x}_1, \dots, \mathbf{x}_N)$ does *not* factorize into the product of individual source distributions $\{p_n(\mathbf{x}_n)\}_{n=1, \dots, N}$. Knowing the joint $p(\mathbf{x}_1, \dots, \mathbf{x}_N)$ implies knowing the distribution over the mixtures $p(\mathbf{y})$ since the latter can be obtained through the sum. The converse is more difficult mathematically, being an inverse problem.

Nevertheless, humans have developed the ability to process multiple sound sources simultaneously in terms of synthesis (i.e., musical composition or generation) and analysis (i.e., source separation). More specifically, composers can invent multiple sources $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ that sum to a consistent mixture \mathbf{y} and, extract information about the individual sources $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ from a mixture \mathbf{y} . This ability to compose and decompose sound is crucial for a generative music model. A model designed to assist in music composition should be capable of isolating individual sources within a mixture and allow for independent operation on each source. Without this feature, it might be hard for musical generative models to be effectively employed in music production tasks: the subsequent manipulation of sub-tracks, creation of accompaniments, and source separation are often required. Therefore, we argue that the task of compositional music generation is highly connected to the task of music source separation.

To the best of our knowledge, no model in deep learning literature was able to perform both tasks simultaneously before our model. Models designed for the generation task directly learn the distribution $p(\mathbf{y})$ over mixtures, collapsing the information needed for the separation task. In this case, we have accurate

¹(e.g. time, spectral, or latent domains)

²Often referred to as *stems*.

mixture modeling but no information about the individual sources. It is worth noting that approaches that model the distribution of mixtures conditioning on textual data [18, 138] face the same limitations. Conversely, models for source separation [139] either target $p(\mathbf{x}_1, \dots, \mathbf{x}_N | \mathbf{y})$, conditioning on the mixture, or learn a single model $p_n(\mathbf{x}_n)$ for each source distribution (in a weakly-supervised manner) and condition on the mixture during inference [8, 107]. In both cases, generating mixtures is impossible. In the first case, the model inputs a mixture, which hinders the possibility of unconditional modeling, not having direct access to $p(\mathbf{x}_1, \dots, \mathbf{x}_N)$ (or equivalently to $p(\mathbf{y})$). In the second case, while we can accurately model each source independently, all essential information about their interdependence is lost, preventing the possibility of generating coherent mixtures.

Contributions. For this line of research on multi-source generation on music, our contributions can be summarize as follow:

- (i) We bridged the gap between source separation and music generation by learning $p(\mathbf{x}_1, \dots, \mathbf{x}_N)$, the joint (prior) distribution of contextual sources (i.e., those belonging to the same song). For this purpose, we use the denoising score-matching framework to train a *Multi-Source Diffusion Model (MSDM)*. We can perform both source separation and music generation during inference by training this single model.
- (ii) With MSDM, we obtained competitive results on source separation against state-of-the-art discriminative models [140] on the Slakh2100 [44] dataset. This is partially due to a novel procedure for computing the posterior score based on *Dirac delta functions*, exploiting the functional relationship between the sources and the mixture.
- (iii) Using our inference time procedure GMSDI, we can tackle all of MSDM capabilities requiring only mixture data for training. The result is thus an unsupervised algorithm when paired with a contrastive encoder.
- (iv) Thanks to its flexible approach, GMSDI can parameterize an arbitrary number and type of sources, allowing for rich semantic control. This might be especially useful for instruments that are less common in stem-separated datasets.
- (v) We proposed CompoNet, a powerful fine-tuned variant of AudioLDM2, able to tackle a wide variety of compositional music generation tasks.

6.2 Related Work

6.2.1 Generative Models for Audio

Deep generative models for audio, learn—either directly or implicitly—the probability density of audio mixtures, represented in our notation by $p(\mathbf{y})$, possibly conditioning on additional data such as text. Various general-purpose

generative models, such as autoregressive models, GANs [141], and diffusion models, have been adapted for use in the audio field.

Autoregressive models have a well-established presence in audio modeling [142]. Jukebox [20] proposed to model musical tracks with Scalable Transformers [16] on hierarchical discrete representations obtained through VQ-VAEs [129]. Furthermore, using a lyrics conditioner, this method generated tracks with vocals following the text. However, while Jukebox could model longer sequences in latent space, the audio output suffered from quantization artifacts. By incorporating residual quantization [143], newer latent autoregressive models [144, 145] can handle extended contexts and output more coherent and naturally sounding generations. State-of-the-art latent autoregressive models for music, such as MusicLM [18], can guide generation by conditioning on textual embeddings obtained via large-scale contrastive pre-training [135, 146]. MusicLM can also input a melody and condition on text for style transfer. A concurrent work, SingSong [147], introduces vocal-to-mixture accompaniment generation. Our accompaniment generation procedures differ from the latter since we aim to perform generation at the stem level in a composable way, while the former outputs a single accompaniment mixture.

DiffWave [148] and WaveGrad [149] 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 [150–153], audio upsampling [154], MIDI-to-waveform [155, 156], or spectrogram-to-MIDI generation [157]. The first work in source-specific generation with diffusion models is CRASH [158]. [128, 159, 160] 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 [161] and Moûsai [138]. Riffusion fine-tunes Stable Diffusion [126], 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.

6.2.2 Compositional Waveform Music Generation

In the music domain, we usually have N distinct source waveforms $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ with $\mathbf{x}_n \in \mathbb{R}^D$ for each n . The sources coherently sum to a mixture $\mathbf{y} = \sum_{n=1}^N \mathbf{x}_n$.

In this setting, multiple tasks can be performed: one may generate a coherent set of waveforms $\{\mathbf{x}_i\}_i$ —such that their sum is a valid mixture \mathbf{y} —or separate the individual sources \mathbf{x} from a given mixture \mathbf{y} . We refer to the first task as *total 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 coherently. We call this task *partial generation* or *accompaniment generation*. In general, we say that compositional music generation (in the waveform domain³) consists in

³as opposed to symbolic domains such as MIDI or sheet music [162]

the generation of source-aware music tracks. That is, we say that a model can perform *compositional* music generation if it allows the manipulation and/or extraction of specific stems during the generation process.

An example of a (non-musical) compositional model is AUDIT [163]. It proposes a diffusion model conditioned by a T5 Encoder [134], trained with instructions that allow the addition, removal (drop), and replacement of sources in an input audio mixture. This model operates on general audio signals with weak dependencies between the sources (e.g., environmental sounds). While MSDM is an unconditional generative model that processes single sources in parallel, AUDIT is a conditional generative model that processes mixtures sequentially. InstructME [122] introduces AUDIT in the musical setting of MSDM, where sources are highly interdependent.

Another model is StemGen [121]: given an instrument tag and an input audio mixture, it generates a single accompaniment source, via a masked music language model [132].

6.2.3 Audio Source Separation

Existing audio source separation models can be broadly classified into discriminative and generative. Discriminative source separators are deterministic parametric models that input the mixtures and systematically extract one or all sources, maximizing the likelihood of some underlying conditional distribution $p(\mathbf{x}_1, \dots, \mathbf{x}_N | \mathbf{y})$. These models are typically trained with a regression loss [164] on the estimated signal represented as waveform [87, 139, 165], STFT [166, 167], or both [83]. On the other hand, generative source separation models learn a prior model for each source, thus targeting the distributions $\{p_n(\mathbf{x}_n)\}_{n=1, \dots, N}$. The mixture is observed only during inference, where a likelihood function connects it to its constituent sources. The literature has explored different priors, such as GANs [104, 106, 168], normalizing flows [107, 169], and autoregressive models⁴ [109].

The separation method closer to ours is the NCSN-BASIS algorithm [107]. This method was proposed for source separation in the image domain, performing Langevin Dynamics for separating the mixtures with an NCSN score-based model. It employs a Gaussian likelihood function during inference, which, as we demonstrate experimentally, is sub-optimal compared to our novel Dirac-based likelihood function.

NOTE:

The main difference between our methods with respect to other generative source separation methods (including NCSN-BASIS) is the modeling of the sources' joint distribution. As such, we can perform source separation and generate mixtures or subsets of stems with a single model.

⁴Such as our LASSsource separation procedure, discussed in section 5.3.

Contextual information between sources is explicitly modeled in [140] and [170]. The first work models the relationship between sources by training an orderless NADE estimator, which predicts a subset of the sources while conditioning on the input mixture and the remaining sources. The subsequent study achieves universal source separation [171, 172] through adversarial training, utilizing a context-based discriminator to model the relationship between sources. Both methods are discriminative, as they are conditioned on the mixtures architecturally. The same architectural limitation is present in discriminative approaches for source separation that use diffusion-based [173, 174] or diffusion-inspired [175] methods. Our method sets itself apart as it proposes a model not constrained architecturally by a mixture conditioner, so we can also perform unconditional generation.

6.3 Multi-Source Diffusion Models

As briefly mentioned in section 6.1, the key contribution of *Multi-Source Diffusion Models (MSDM)* is the ability to perform all compositional tasks⁵ of *total generation*, *accompaniment generation*, and *source separation* simultaneously. This can be done by training a single diffusion model that captures the prior $p(\mathbf{x}_1, \dots, \mathbf{x}_N)$. Indeed, the model—illustrated in fig. 6.1—approximates the noisy score function:

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

with a neural network:

$$S_\theta(\mathbf{x}(t), \sigma(t)) : \mathbb{R}^{N \times D} \times \mathbb{R} \rightarrow \mathbb{R}^{N \times D}, \quad (6.1)$$

where $\mathbf{x}(t) = (\mathbf{x}_1(t), \dots, \mathbf{x}_N(t))$ denotes the sources perturbed with the Gaussian kernel in eq. (2.4).

6.3.1 Compositional Tasks

The three tasks of our method are solved during inference by discretizing the backward eq. (2.6). Although different tasks require distinct score functions, they all originate directly from the prior score function in eq. (6.1). We analyze each of these score functions in detail.

Total Generation The total generation task is performed by sampling from eq. (2.6) using the score function in $S_t \theta(\mathbf{x}, \sigma)$. The mixture is then obtained by summing over all the generated sources.

Partial Generation In the partial generation task, we fix a subset of source indices $\mathcal{I} \subset \{1, \dots, N\}$ and the relative sources $\mathbf{x}_{\mathcal{I}} := \{\mathbf{x}_n\}_{n \in \mathcal{I}}$. The goal is to generate the remaining sources $\mathbf{x}_{\bar{\mathcal{I}}} := \{\mathbf{x}_n\}_{n \in \bar{\mathcal{I}}}$ consistently, where $\bar{\mathcal{I}} =$

⁵Described in section 6.2.2.

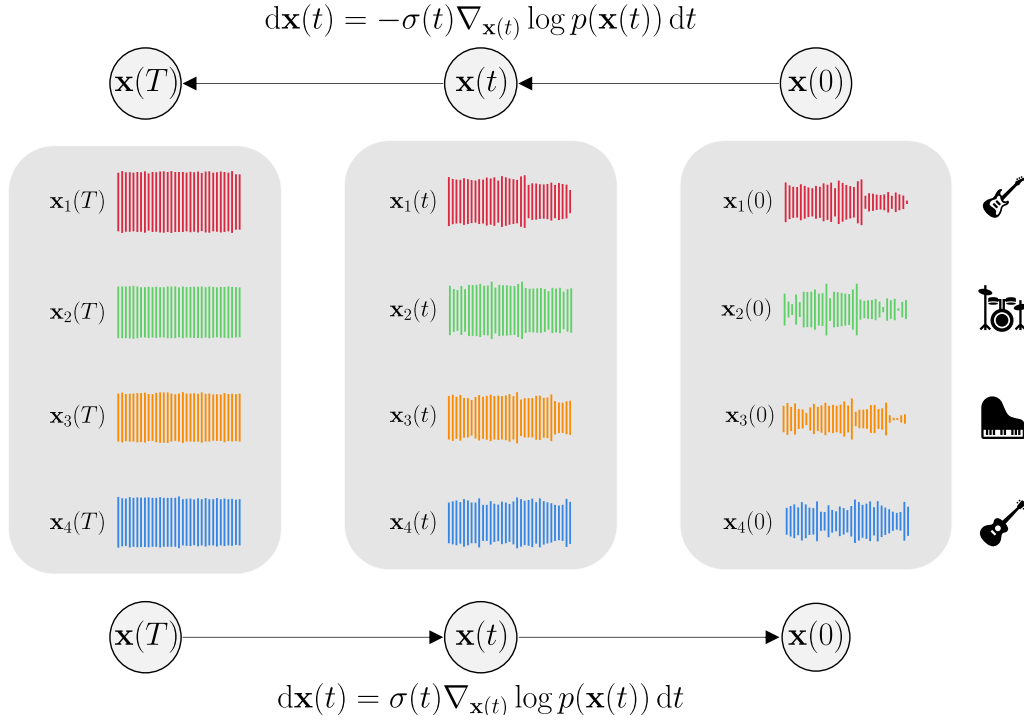


FIGURE 6.1: **Diagram illustrating the MSDM method.** MSDM leverages a forward Gaussian process (right-to-left) to learn the score over contextual sets (indicated by large rectangles) of instrumental sources (represented by waveforms) across different time steps t . During inference, the process is reversed (left-to-right), enabling us to perform tasks such as total generation, partial generation, or source separation (detailed in fig. 6.2).

$\{1, \dots, N\} - \mathcal{I}$. To do so, we estimate the gradient of the conditional distribution:

$$\nabla_{\mathbf{x}_{\bar{\mathcal{I}}}(t)} \log p(\mathbf{x}_{\bar{\mathcal{I}}}(t) \mid \mathbf{x}_{\mathcal{I}}(t)). \quad (6.2)$$

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 [27]. The gradient in eq. (6.2) is approximated as follows:

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

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:

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

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

Source Separation We view source separation as a specific instance of conditional generation, where we condition the generation process on the given

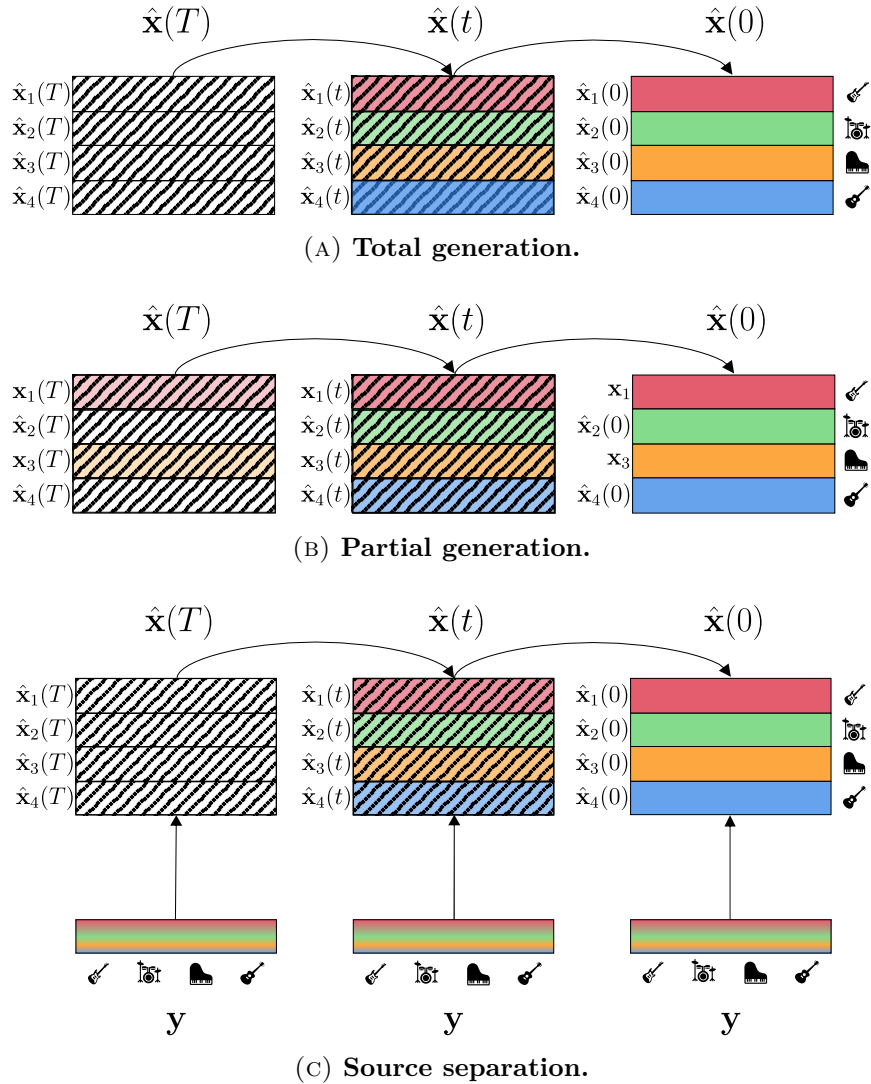


FIGURE 6.2: **Inference tasks with MSDM.** Oblique lines represent the presence of noise in the signal, decreasing from left to right, with the highest noise level at time T when we start the sampling procedure. **A:** We generate all stems in a mixture, obtaining a total generation. **B:** We perform partial generation (accompaniment generation) by fixing the sources \mathbf{x}_1 (Bass) and \mathbf{x}_3 (Piano) and generating the other two sources $\hat{\mathbf{x}}_2(0)$ (Drums) and $\hat{\mathbf{x}}_4(0)$ (Guitar). We denote with $\mathbf{x}_1(t)$ and $\mathbf{x}_3(t)$, the noisy stems obtained from \mathbf{x}_1 and \mathbf{x}_3 via the perturbation kernel in eq. (2.4). **C:** We perform source separation by conditioning the prior with a mixture \mathbf{y} , following algorithm 3.

Algorithm 3 ‘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}

- 1: Initialize $\hat{\mathbf{x}} \sim \mathcal{N}(0, \sigma_I^2 \mathbf{I})$
- 2: $\alpha \leftarrow \min(S_{\text{churn}}/I, \sqrt{2} - 1)$
- 3: **for** $i \leftarrow I$ **to** 1 **do**
- 4: **for** $r \leftarrow R$ **to** 0 **do**
- 5: $\hat{\sigma} \leftarrow \sigma_i \cdot (\alpha + 1)$
- 6: $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
- 7: $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} + \sqrt{\hat{\sigma}^2 - \sigma_i^2} \epsilon$
- 8: $\mathbf{z} \leftarrow [\hat{\mathbf{x}}_{1:N-1}, \mathbf{y} - \sum_{n=1}^{N-1} \hat{\mathbf{x}}_n]$
- 9: **for** $n \leftarrow 1$ **to** $N - 1$ **do**
- 10: $\mathbf{g}_n \leftarrow S_n^\theta(\mathbf{z}, \hat{\sigma}) - S_N^\theta(\mathbf{z}, \hat{\sigma})$
- 11: **end for**
- 12: $\mathbf{g} \leftarrow [\mathbf{g}_1, \dots, \mathbf{g}_{N-1}]$
- 13: $\hat{\mathbf{x}}_{1:N-1} \leftarrow \hat{\mathbf{x}}_{1:N-1} + (\sigma_{i-1} - \hat{\sigma}) \mathbf{g}$
- 14: $\hat{\mathbf{x}} \leftarrow [\hat{\mathbf{x}}_{1:N-1}, \mathbf{y} - \sum_{n=1}^{N-1} \hat{\mathbf{x}}_n]$
- 15: **if** $r > 0$ **then**
- 16: $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
- 17: $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2} \epsilon$
- 18: **end if**
- 19: **end for**
- 20: **end for**
- 21: **return** $\hat{\mathbf{x}}$

mixture $\mathbf{y} = \mathbf{y}(0)$. This requires computing the score function of the posterior distribution:

$$\nabla_{\mathbf{x}(t)} \log p(\mathbf{x}(t) \mid \mathbf{y}(0)). \quad (6.3)$$

Standard methods for implementing conditional generation for diffusion models involve directly estimating the posterior score in eq. (6.3) at training time (i.e., Classifier Free Guidance, as described in [33]) 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 [176].

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 [107], where they use a Gaussian likelihood function:

$$p(\mathbf{y}(t) \mid \mathbf{x}(t)) = \mathcal{N}(\mathbf{y}(t) \mid \sum_{n=1}^N \mathbf{x}_n(t), \gamma^2(t) \mathbf{I}), \quad (6.4)$$

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. (6.3) by modeling $p(\mathbf{y}(t) | \mathbf{x}(t))$ as a Dirac delta function centered in $\sum_{n=1}^N \mathbf{x}_n(t)$:

$$p(\mathbf{y}(t) | \mathbf{x}(t)) = \mathbb{1}_{\mathbf{y}(t)=\sum_{n=1}^N \mathbf{x}_n(t)}. \quad (6.5)$$

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:

$$\begin{aligned} \nabla_{\mathbf{x}_m(t)} \log p(\mathbf{x}(t) | \mathbf{y}(0)) &\approx S_m^\theta((\mathbf{x}_1(t), \dots, \mathbf{x}_{N-1}(t), \mathbf{y}(0) - \sum_{n=1}^{N-1} \mathbf{x}_n(t)), \sigma(t)) \quad (6.6) \\ &- S_N^\theta((\mathbf{x}_1(t), \dots, \mathbf{x}_{N-1}(t), \mathbf{y}(0) - \sum_{n=1}^{N-1} \mathbf{x}_n(t)), \sigma(t)), \end{aligned} \quad (6.7)$$

where $1 \leq m \leq N - 1$ and S_m^θ, S_N^θ denote the entries of the score network corresponding to the m -th and N -th sources. Our approach models the limiting case wherein $\gamma(t) \rightarrow 0$ 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 generative source separation [8, 107, 169]. 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 $p(\mathbf{x}_1, \dots, \mathbf{x}_N) = \prod_{n=1}^N p_n(\mathbf{x}_n)$ and train a separate model for each source class. We call the resulting model ‘Independent Source Diffusion Model with Dirac Likelihood’ or ‘ISDM Dirac’. While the ISDM method lacks generative capabilities, it enables us to demonstrate the effectiveness of generative source separation when combined with Dirac likelihood.

The sampler. Our approach utilizes a first-order ODE integrator, specifically the Euler method, and incorporates stochasticity via the S_{churn} mechanism as discussed in [29]. Additionally, we apply a correction step as described in [27, 107]. 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 3.

6.3.2 Experimental Results

We perform experiments on Slakh2100 [44], a common dataset for music source separation. We chose Slakh2100 because it has a significantly larger quantity of data (145h) than other multi-source waveform datasets like MUSDB18 [177]

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 6.1: **Comparison between total generation capabilities of MSDM and an equivalent architecture trained on mixtures.** Both subjective (quality and coherence) and objective (FAD) evaluations are shown. Subjective evaluation is performed through listening tests, where subjects are asked to evaluate songs from 1 to 10 with respect to the overall quality of the chunk and to coherence (i.e., how the instruments sound plausible together). Results show a very small 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.*

(10h). The amount of data plays a decisive role in determining the quality of a generative model, making Slakh2100 a preferable choice.

Music Generation

The performance of MSDM on the generative tasks is tested through subjective and objective evaluation. Subjective evaluation is carried out through listening tests. Concisely, we produced an online form used for the results shown in table 6.1. In this form, subjects were asked to rate—from 1 to 10—the quality and instrument coherence of 30 generated chunks, of which 15 are generated from the mixture model and 15 from MSDM.

As for the objective evaluation of the generative tasks, we generalize the FAD protocol in [147] to our total generation and partial generation tasks. 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}) and 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 and discussed in table 6.1 and table 6.3 respectively. Concisely, table 6.1 shows that the generative power of MSDM is the same of a model with the same architecture and trained on mixtures of the same dataset. Table 6.3 shows that the task of partial generation can be performed with non-trivial quality and can be used as a baseline for future works on general accompaniment generation.

Model	Bass	Drums	Guitar	Piano	All
Demucs [139, 140]	15.77	19.44	15.30	13.92	16.11
Demucs + Gibbs (512 steps) [140]	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 [107]					
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

TABLE 6.2: **Quantitative results for source separation on the Slakh2100 test set.** We use the SI-SDR_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 [140] overall.* We stress again that while the baselines are trained on the separation task alone, MSDM is able to perform also generative tasks.

Source Separation

We compare our supervised MSDM and weakly-supervised MSDM with the ‘Demucs’ [139] and ‘Demucs + Gibbs (512 steps)’ regressor baselines from [140], the state-of-the-art for supervised music source separation on Slakh2100, aligning with the evaluation procedure of [140].

NOTE:

To evaluate source separation, we use the scale-invariant SDR improvement (SI-SDR_i) metric [178]. 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 $\alpha = \frac{\mathbf{x}_n^\top \hat{\mathbf{x}}_n + \epsilon}{\|\mathbf{x}_n\|^2 + \epsilon}$ and $\epsilon = 10^{-8}$. The improvement with respect to the mixture baseline is defined as $\text{SI-SDR}_i = \text{SI-SDR}(\mathbf{x}_n, \hat{\mathbf{x}}_n) - \text{SI-SDR}(\mathbf{x}_n, \mathbf{y})$.

Slakh2100		
Generated Sources	Input Sources	FAD
Single source imputation		
<i>Bass</i>	<i>Drums, Guitar, Piano</i>	0.45
<i>Drums</i>	<i>Bass, Guitar, Piano</i>	1.09
<i>Guitar</i>	<i>Bass, Drums, Piano</i>	0.11
<i>Piano</i>	<i>Bass, Drums, Guitar</i>	0.76
Two sources imputation		
<i>Bass, Drums</i>	<i>Guitar, Piano</i>	2.09
<i>Bass, Guitar</i>	<i>Drums, Piano</i>	1.00
<i>Bass, Piano</i>	<i>Drums, Guitar</i>	2.32
<i>Drums, Guitar</i>	<i>Bass, Piano</i>	1.45
<i>Drums, Piano</i>	<i>Bass, Guitar</i>	1.82
<i>Guitar, Piano</i>	<i>Bass, Drums</i>	1.65
Three sources imputation		
<i>Bass, Drums, Guitar</i>	<i>Piano</i>	2.93
<i>Bass, Drums, Piano</i>	<i>Guitar</i>	3.30
<i>Bass, Guitar, Piano</i>	<i>Drums</i>	4.90
<i>Drums, Guitar, Piano</i>	<i>Bass</i>	3.10

TABLE 6.3: **Quantitative results for the partial generation task on Slakh2100.** We use the FAD as our objective evaluation metric (lower is better). No baseline is reported since our work is the first able to generate any combination of accompaniments; the results thus pose a baseline for future works on general accompaniment generation.

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_i on such segments. We report results comparing our Dirac score posterior with the Gaussian score posterior of [107], using the best parameters of the ablation experiments and 150 inference steps. Our results are illustrated and discussed in table 6.2. Concisely, MSDM proves to be very close to the state of the art. Moreover, our newly defined sampling procedure, when used in the weakly supervised flavor, yields results that are better than the competitors on some stems.

6.3.3 Limitations

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. This flexibility paves the way for more advanced music composition tools, where users can easily control and manipulate individual sources within a mixture. However, the amount of available contextual data constrains the performance of our model. To address this, pre-separating mixtures and training on the separations, as demonstrated in [147], may prove beneficial.

6.4 Generalized Multi-source Diffusion Inference

We train a text-conditioned diffusion model (eq. (2.9)) $S_\theta(\mathbf{y}(t), \mathbf{z}, \sigma(t))$, with pairs of 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 $\mathbf{z}_1 \otimes \cdots \otimes \mathbf{z}_K$ (more compactly $\bigotimes_{k=1}^K \mathbf{z}_k$), where each \mathbf{z}_k describes a source \mathbf{x}_k present in \mathbf{y} and \otimes denotes an encoding of concatenated textual information (e.g., $\mathbf{z}_1 \otimes \cdots \otimes \mathbf{z}_K = E_\phi^{\text{text}}(\mathbf{q}_1, \dots, \mathbf{q}_K)$, with $E_\phi^{\text{text}}(\mathbf{q}_k) = \mathbf{z}_k$). The idea is to leverage such text embeddings for parameterizing the individual source score functions:

$$\nabla_{\mathbf{x}_k(t)} \log p(\mathbf{x}_k(t) | \mathbf{z}_k) \approx S_\theta(\mathbf{x}_k(t), \mathbf{z}_k, \sigma(t)), \quad (6.8)$$

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} in the relaxed data setting.

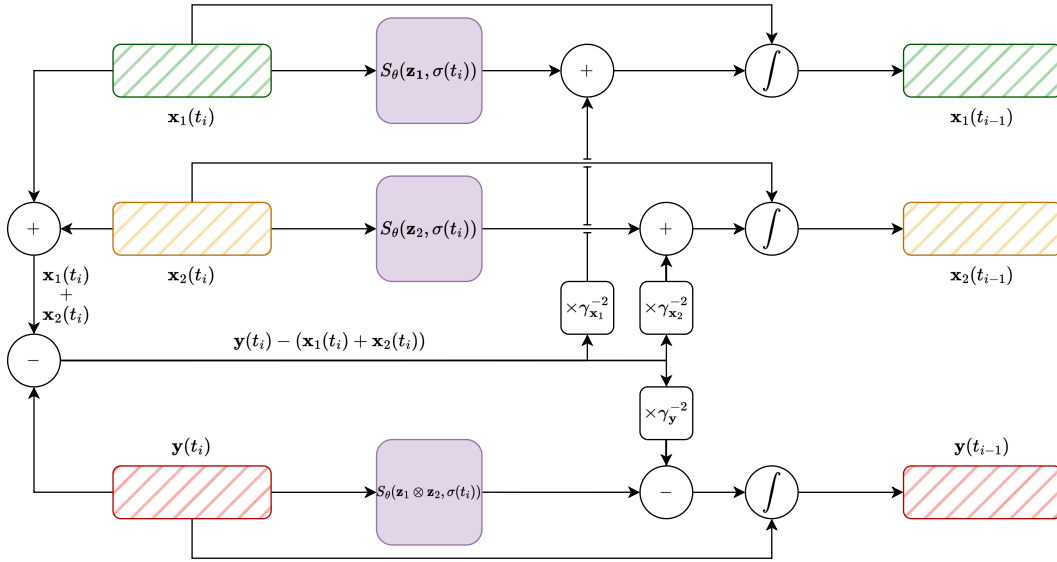


FIGURE 6.3: Diagram for unconditional generation procedure with GMSDI, sampling two coherent sources.

6.4.1 Total generation

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

$$\frac{p(\mathbf{x}(t), \mathbf{y}(t) | \mathbf{z}_1, \dots, \mathbf{z}_K, \mathbf{z}_1 \otimes \cdots \otimes \mathbf{z}_K)}{p(\mathbf{x}_{\bar{k}}(t), \mathbf{y}(t) | \mathbf{z}_{\bar{k}}, \mathbf{z}_1 \otimes \cdots \otimes \mathbf{z}_K)}. \quad (6.9)$$

First, we develop the numerator in eq. (6.9) using the chain rule:

$$\begin{aligned}
& p(\mathbf{x}(t), \mathbf{y}(t) \mid \mathbf{z}_1, \dots, \mathbf{z}_K, \mathbf{z}_1 \otimes \dots \otimes \mathbf{z}_K) \\
&= p(\mathbf{x}_k(t) \mid \mathbf{z}_k) p(\mathbf{y}(t), \mathbf{x}_{\bar{k}}(t) \mid \mathbf{x}_k(t), \mathbf{z}_{\bar{k}}, \mathbf{z}_1 \otimes \dots \otimes \mathbf{z}_K) \\
&= p(\mathbf{x}_k(t) \mid \mathbf{z}_k) p(\mathbf{y}(t) \mid \mathbf{x}(t)) p(\mathbf{x}_{\bar{k}}(t) \mid \mathbf{x}_k(t), \mathbf{z}_{\bar{k}}) \\
&\approx p(\mathbf{x}_k(t) \mid \mathbf{z}_k) p(\mathbf{y}(t) \mid \mathbf{x}(t)). \tag{6.10}
\end{aligned}$$

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 $p(\mathbf{x}_{\bar{k}}(t) \mid \mathbf{x}(t), \mathbf{z}_{\bar{k}})$. We substitute eq. (6.10) in eq. (6.9), take the gradient of the logarithm with respect to $\mathbf{x}_k(t)$ and model the likelihood with isotropic Gaussians [179] depending on a variance $\gamma_{\mathbf{x}_k}^2$:

$$\begin{aligned}
& \nabla_{\mathbf{x}_k(t)} \frac{\log p(\mathbf{x}_k(t) \mid \mathbf{z}_k) p(\mathbf{y}(t) \mid \mathbf{x}(t))}{\log p(\mathbf{x}_{\bar{k}}(t), \mathbf{y}(t) \mid \mathbf{z}_{\bar{k}}, \mathbf{z}_1 \otimes \dots \otimes \mathbf{z}_K)} \\
&= \nabla_{\mathbf{x}_k(t)} \log p(\mathbf{x}_k(t) \mid \mathbf{z}_k) + \nabla_{\mathbf{x}_k(t)} \log p(\mathbf{y}(t) \mid \mathbf{x}(t)) \\
&= \nabla_{\mathbf{x}_k(t)} \log p(\mathbf{x}_k(t) \mid \mathbf{z}_k) + \nabla_{\mathbf{x}_k(t)} \log \mathcal{N}(\mathbf{y}(t) \mid \sum_{l=1}^K \mathbf{x}_l(t), \gamma_{\mathbf{x}_k}^2 \mathbf{I}) \\
&= \nabla_{\mathbf{x}_k(t)} \log p(\mathbf{x}_k(t) \mid \mathbf{z}_k) + \frac{1}{\gamma_{\mathbf{x}_k}^2} (\mathbf{y}(t) - \sum_{l=1}^K \mathbf{x}_l(t)). \tag{6.11}
\end{aligned}$$

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

$$\begin{aligned}
& p(\mathbf{y}(t) \mid \mathbf{x}(t), \mathbf{z}_1, \dots, \mathbf{z}_K, \mathbf{z}_1 \otimes \dots \otimes \mathbf{z}_K) \\
&\approx \nabla_{\mathbf{y}(t)} \log p(\mathbf{y}(t) \mid \bigotimes_{l=1}^K \mathbf{z}_l) + \frac{1}{\gamma_{\mathbf{y}}^2} (\sum_{l=1}^K \mathbf{x}_l(t) - \mathbf{y}(t)). \tag{6.12}
\end{aligned}$$

During inference, we sample from eq. (6.11) and eq. (6.12) in *parallel*, replacing the gradients of the log-densities with score models (eq. (6.8)):

$$\begin{cases} S_{\theta}(\mathbf{x}_k(t), \mathbf{z}_k, \sigma(t)) + \frac{1}{\gamma_{\mathbf{x}_k}^2} (\mathbf{y}(t) - \sum_{l=1}^K \mathbf{x}_l(t)) \\ S_{\theta}(\mathbf{y}(t), \bigotimes_{l=1}^K \mathbf{z}_l, \sigma(t)) + \frac{1}{\gamma_{\mathbf{y}}^2} (\sum_{l=1}^K \mathbf{x}_l(t) - \mathbf{y}(t)). \end{cases} \tag{6.13}$$

A diagram of the method is illustrated in fig. 6.3. Given a partition $\{\mathcal{J}_m\}_{m \in [M]}$ of $[K]$ containing M subsets (i.e., $\cup_{m \in [M]} \mathcal{J}_m = [K]$), we can perform inference more generally with:

$$\begin{cases} S_{\theta}(\sum_{j \in \mathcal{J}_m} \mathbf{x}_j(t), \bigotimes_{j \in \mathcal{J}_m} \mathbf{z}_j, \sigma(t)) + \frac{1}{\gamma_{\mathcal{J}_m}^2} (\mathbf{y}(t) - \sum_{l=1}^K \mathbf{x}_l(t)) \\ S_{\theta}(\mathbf{y}(t), \bigotimes_{l=1}^K \mathbf{z}_l, \sigma(t)) + \frac{1}{\gamma_{\mathbf{y}}^2} (\sum_{l=1}^K \mathbf{x}_l(t) - \mathbf{y}(t)). \end{cases} \tag{6.14}$$

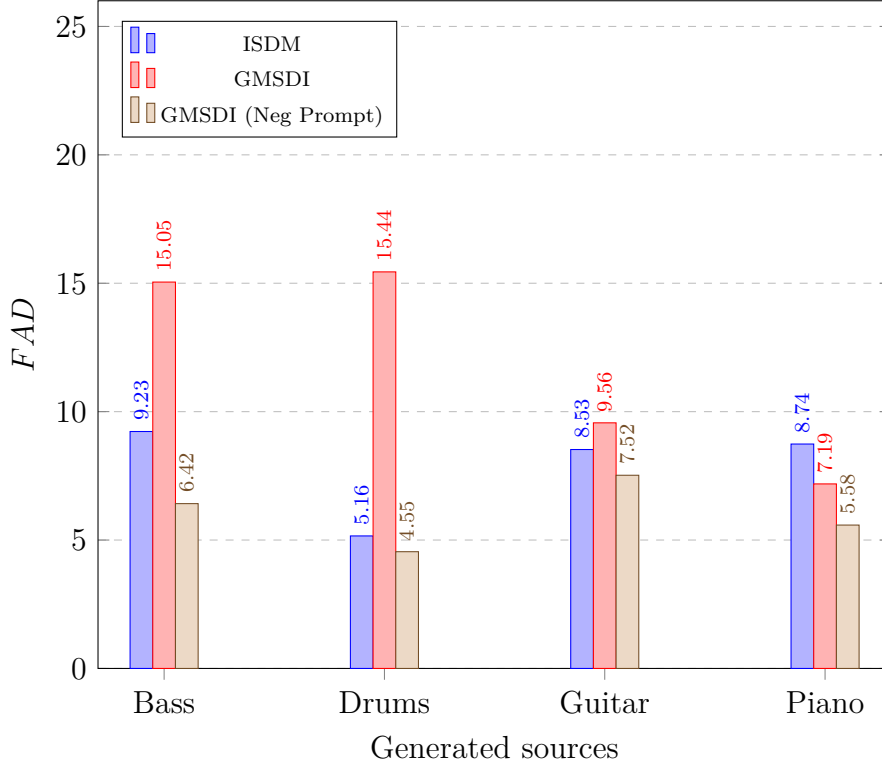


FIGURE 6.4: **FAD between generated sources and Slakh2100 test data (200 chunks, ~12s each).** Neg Prompt indicates the presence of negative prompting.

6.4.2 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 eq. (6.13) for $j \in \mathcal{J}$:

$$\begin{cases} S_{\theta}(\mathbf{x}_j(t), \mathbf{z}_j(t), \sigma(t)) + \frac{1}{\gamma_{\mathbf{x}_j}^2} [\mathbf{y}(t) - (\alpha \sum_{i \in \mathcal{I}} \mathbf{x}_i(t) + \beta \sum_{l \in \mathcal{J}} \mathbf{x}_l(t))] \\ S_{\theta}(\mathbf{y}(t), \bigotimes_{l=1}^K \mathbf{z}_l, \sigma(t)) + \frac{1}{\gamma_{\mathbf{y}}^2} [(\alpha \sum_{i \in \mathcal{I}} \mathbf{x}_i(t) + \beta \sum_{l \in \mathcal{J}} \mathbf{x}_l(t)) - \mathbf{y}(t)] \end{cases}, \quad (6.15)$$

with $\mathbf{x}_i(t)$ ($i \in \mathcal{I}$) sampled from the perturbation kernel in eq. (2.4) conditioned on \mathbf{x}_i and $\alpha, \beta \in \mathbb{R}$ scaling factors. Using eq. (6.14), we can generate the accompaniment mixtures $\sum_{j \in \mathcal{J}} \mathbf{x}_j$ directly.

6.4.3 Source separation

Source separation can be performed by adapting eq. (6.6) to the text-conditioned model. Let an observable mixture $\mathbf{y}(0)$ be composed by sources described by $\{\mathbf{z}_k\}_{k \in [K]}$. We can separate the sources by choosing a constrained source (w.l.o.g. the K -th) and sampling, for $k \in [K - 1]$, with:

$$S_{\theta}(\mathbf{x}_k(t), \mathbf{z}_k, \sigma(t)) - S_{\theta}(\mathbf{y}(0) - \sum_{l=1}^{K-1} \mathbf{x}_l(t), \mathbf{z}_K, \sigma(t)). \quad (6.16)$$

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

$$S_{\theta}(\mathbf{x}_k(t), \mathbf{z}_k, \sigma(t)) - S_{\theta}(\mathbf{y}(0) - \mathbf{x}_k(t), \bigotimes_{l \neq k} \mathbf{z}_l, \sigma(t)), \quad (6.17)$$

constraining the mixture $\sum_{l \neq k} \mathbf{x}_l(t)$, complementary to $\mathbf{x}_k(t)$.

6.4.4 Experimental Setup

To validate our theoretical claims, we train two time-domain Moûsai-like [133] diffusion models. The first model is trained on Slakh2100 [44]. 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 [134], 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 (~ 12 s).

The second model is trained on a more realistic dataset, namely MTG-Jamendo [43]. MTG-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 (~ 24 s) at 22kHz. We condition the model with the pre-trained checkpoint `music_audioset_epoch_15_esc_90.14.pt`⁶ of the LAION CLAP contrastive encoder [137]. 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 ADPM2⁷ [180] with $\rho = 1$ for generation and AEuler² with $s_{\text{churn}} = 20$ for separation.

6.4.5 Experimental Results

First, we want to understand whether the model trained on Slakh2100 mixtures can parametrize single sources well. We sample, for each stem, 200 chunks of ~ 12 s, conditioning with embeddings of single stem labels (e.g., “Bass”). Then, we compute the Fréchet Audio Distance (FAD) [181] with VGGish embeddings between such samples and 200 random Slakh2100 test chunks of the same source. In fig. 6.4, we compare our model against the weakly supervised version of MSDM [9], 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 2.2.2) essential for obtaining non-silent results using “Drums, Guitar, Piano” (Bass), “Bass”

⁶<https://github.com/LAION-AI/CLAP>

⁷<https://github.com/crowsonkb/k-diffusion>

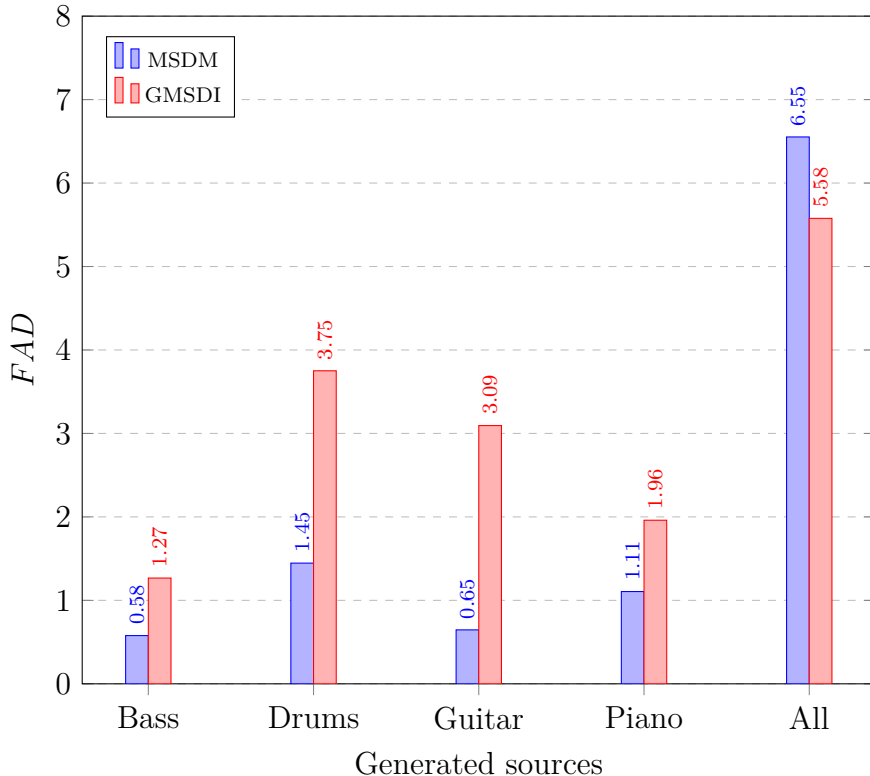


FIGURE 6.5: **FAD results on total and partial generation.** These results were obtained on Slakh2100 test mixtures. In particular, 200 chunks were randomly selected, with ~ 12 seconds of sound each.

(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 fig. 6.5, we compute the FAD between 200 random Slakh2100 test mixture chunks (~ 12 s 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 $\gamma_y = \infty$ and reach ~ 1 lower FAD point, using 600 sampling steps. On partial generation, we sample using 300 steps, setting $\gamma_y \ll \infty$, 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 SI-SDR improvement (SI-SDR_i) [182] as an evaluation metric and follow the evaluation protocol of [9]. First, we perform a grid search (table 6.4) to find a good embedding scale 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 6.5. 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

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

TABLE 6.4: **Grid search over embedding scale w on 100 chunks (~12s each) of Slakh2100 test set.** Results in SI-SDR_i. The source in parenthesis is the constrained source.

Model	Bass	Drums	Guitar	Piano	All
Demucs + Gibbs (512 steps) [140]	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

TABLE 6.5: **Quantitative results for source separation on the Slakh2100 test set.** Results in SI-SDR_i (dB – higher is better).

training [183].

6.4.6 Limitations

While the lack of a requirement of specific stem-separated data is a nice upside of this approach, we still need models that have a good enough “understanding” single-stem distributions for the GMSDI procedure to work effectively. This might be true for probably the most common instruments—such as piano and guitar—it might be more difficult for less popular instruments, or for instruments that rarely can be heard performing by themselves.

Another important drawback of GMSDI is the necessity to stay in the time domain, rather than the latent space of a VAE, like most successful diffusion models [126, 128, 184]. Thus it is required to enhance this method for usage in the latent domain⁸.

⁸It is not doable with the standard approach since the latent space of a VAE might not preserve linearity enough for a correct sampling.

6.5 CompoNet

As a conclusion of our multi-source diffusion work, we propose an improved compositional model for music called *CompoNet* and compare it with our initial MSDM model (section 6.3).

CompoNet uses a pre-trained latent diffusion model ϵ_ϕ , based on U-Net architecture [185] conditioned via cross-attention layers [16], on a large dataset of tuples (\mathbf{m}, \mathbf{c}) comprising audio mixtures \mathbf{m} and relative textual descriptions \mathbf{c} . The mixtures \mathbf{m} are mapped to latent vectors $\mathbf{z} = E_{\text{VAE}}(\mathbf{m})$ using a pre-trained VAE encoder [13], while the text descriptions are mapped to a continuous sequence $\mathbf{s} = E_{\text{TXT}}(\mathbf{c})$ using a text encoder (e.g., [134]). Following the DDPM formulation [28], the model is trained to reverse the forward Gaussian noising process given by:

$$\mathbf{z}_t = \sqrt{\bar{\alpha}_t} \mathbf{z} + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad (6.18)$$

where $t \in [0, T]$ is a time index (with T the maximum time step), $\bar{\alpha}_t$ is defined by integrating a noise schedule [28], and $\boldsymbol{\epsilon}$ is sampled from the standard Gaussian distribution. The model is trained with denoising score matching [186]:

$$\min_{\phi} \mathbb{E}_{\mathbf{z}, \boldsymbol{\epsilon}, \mathbf{s}, t} [\|\boldsymbol{\epsilon} - \epsilon_\phi(\mathbf{z}_t, \mathbf{s}, t)\|_2^2], \quad (6.19)$$

with \mathbf{z}_t obtained via eq. (6.18).

For our downstream task, we fine-tune a ControlNet [120] adapter C_ψ for tackling all compositional musical tasks (fig. 6.6) with a single model.

The U-Net ϵ_ϕ comprises an encoder, bottleneck, decoder structure $\epsilon_\phi = D_{\phi_D} \circ B_{\phi_B} \circ E_{\phi_E}$. The ControlNet adapter is defined as $C_\psi(\mathbf{z}_t, \mathbf{s}, t, \mathbf{w}) = E_{\phi_E}(\mathbf{z}_t + \text{conv}_{\text{in}}(\mathbf{w}), \mathbf{s}, t)$, where conv_{in} is a zero-initialized convolutional layer, and \mathbf{w} is a latent (VAE) embedding of an external audio input. The ControlNet adapter outputs the set of processed features $\{C_\psi^i(\mathbf{z}_t, \mathbf{s}, t, \mathbf{w})\}_{i=1, \dots, I}$ for each layer of E_{ϕ_E} , with I the total number of layers. The full ControlNet conditional architecture is defined as $\epsilon_{\phi, \psi} = D_{\phi_D} \circ B_{\phi_B} \circ E_{\phi_E, \psi}$ with $E_{\phi_E, \psi}$ combining the encoder and ControlNet adapter features at each layer i with zero-initialized convolutional layers conv_i :

$$E_{\phi_E, \psi}^i(\mathbf{z}_t, \mathbf{s}, t, \mathbf{w}) = E_{\phi_E}^i(\mathbf{z}_t, \mathbf{s}, t) + \text{conv}_i(C_\psi^i(\mathbf{z}_t, \mathbf{s}, t, \mathbf{w})).$$

While we have described the general ControlNet architecture, we still have to describe how we train it in CompoNet, namely, the roles of the \mathbf{z} , \mathbf{w} and \mathbf{s} variables. Iterating over a dataset containing tuples (\mathbf{x}, \mathbf{t}) with multi-stem tracks $\mathbf{x} = \{\mathbf{x}_n\}_{n=1, \dots, N}$ and tag descriptions $\mathbf{t} = \{\mathbf{t}_n\}_{n=1, \dots, N}$ for each stem, we sample from \mathbf{x} two arbitrary subsets of stems $Y, X \subseteq \mathbf{x}$, with $|X| > 0$. Y contains input stems while X contains output stems. The topological relationships between such subsets define all possible compositional tasks, as depicted in fig. 6.6. While previous models partially solve some tasks (see tables 6.6 and 6.7), ours is the first to solve all of them simultaneously. We proceed like in eq. (7.1) and mix the

sources in Y and X , obtaining \mathbf{m}_Y and \mathbf{m}_X , respectively. Afterward, we encode them in the VAE latent space, defining $\mathbf{z} = E^{\text{VAE}}(\mathbf{m}_X)$ and $\mathbf{w} = E^{\text{VAE}}(\mathbf{m}_Y)$. We define the following prompt \mathbf{s} :

$$\mathbf{s} = E^{\text{TXT}}(\mathbf{t}_{Y_1}, \dots, \mathbf{t}_{Y_{|Y|}}, \text{SEP}, \mathbf{t}_{X_1}, \dots, \mathbf{t}_{X_{|X|}}), \quad (6.20)$$

specifying input and output mixture tags separated by a special token **SEP**.

Having specified the required inputs and outputs, we train $\epsilon_{\phi, \psi}$ via the usual denoising scoring matching loss, optimizing only the parameters of the ControlNet encoder. The \mathbf{s} prompt instructs the model which task to perform based on the specified stems combination.

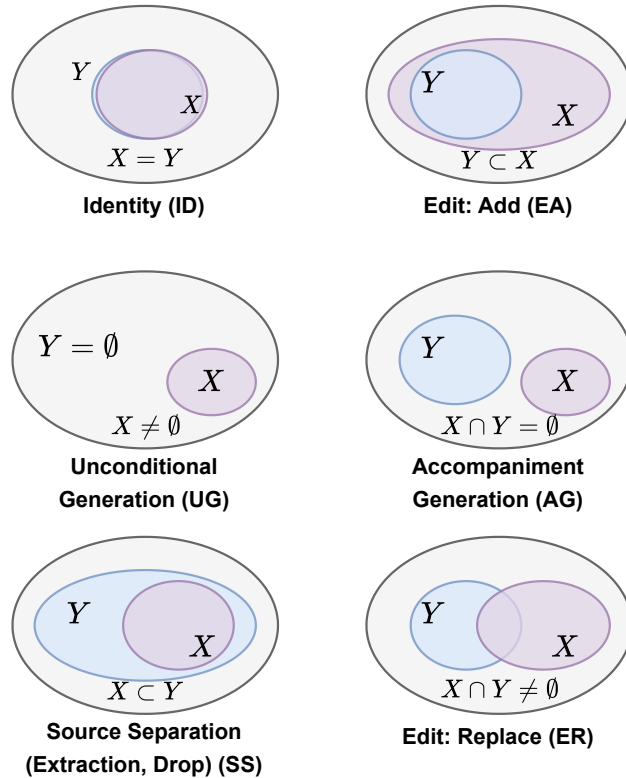


FIGURE 6.6: **Inter-stem compositional generation tasks.** Y and X represent the input and output stem sub-sets, respectively.

6.5.1 Experimental Setup

Model implementation. In CompoNet, we employ the AudioLDM2 [128, 184] architecture. Since the authors pre-train the model on a large array of datasets [187–194], we skip the pretext-task training phase and directly fine-tune a ControlNet adapter based on the AudioLDM 2-Large checkpoint⁹. During fine-tuning, we directly pass the conditioning prompt in eq. (6.20) to the text-embedding mechanism of AudioLDM2 (based on CLAP [136, 137], T5 [134], and GPT2 [72]), conditioning both the U-Net and the ControlNet adapter.

⁹<https://huggingface.co/cvssp/audioldm2-large>

Model	Task				
	UG	AG	SS	EA	ER
MSDM	✓	✓(MS)	✓(MS)	✗	✗
GMSDI	✓	✓(MS)	✓(MS)	✗	✗
StemGen [121]	✓	✓(1S)	✗	✗	✗
Audit [163]	✓	✗	✓(Remove 1S)	✓(1S)	✓(1S)
InstructME [122]	✓	✗	✓(Extract 1S; Remove 1S)	✓(1S)	✓(1S)
CompoNet	✓	✓(MS)	✓(MS)	✓(MS)	✓(MS)

TABLE 6.6: **Compositional audio models comparison.** The various tasks are illustrated in fig. 6.6. **1S vs MS**: the task operates on one vs multiple sources at a time.

Model	Methodology	Input	Output	Coherence
MSDM	Train / Supervised	Multi / Source	Multi / Source	✓
GMSDI	Inf. / Weakly Sup.	Multi / Sub-mix	Multi / Sub-mix	✓
StemGen	Train / Supervised	Single / Sub-mix	Single / Source	✓
Audit	Train / Supervised	Single / Sub-mix	Single / Sub-mix	✗
InstructME	Train / Supervised	Single / Sub-mix	Single / Sub-mix	✓
CompoNet	Train / Fine-tuning	Single / Sub-mix	Single / Sub-mix	✓

TABLE 6.7: **Compositional audio models comparison.** The various tasks are illustrated in fig. 6.6. **Multi vs Single** on Input / Output: the model accepts multiple vs single inputs / outputs. **Source vs Sub-mix** on Input / Output: the model processes single sources or sub-mixes as inputs / outputs.

6.5.2 Experimental Results

For the accompaniment generation evaluation, we compare our MSDM model with the new proposed CompoNet. We train CompoNet on MUSDB18-HQ and Slakh2100 (restricted to Bass, Drums, Guitar, and Piano stems at test time). We also consider a Random baseline, where, for a given input, we output a random sub-mix from a different test track. We generate 200 chunks for both datasets and models, conditioning on random stem subsets of test tracks and querying a subset of the complementary. The chunks are $\sim 6s / 10.24s$ long on MUSDB18-HQ / Slakh2100. Given that MSDM tends to generate silence, we sample 12 candidate tracks for each generated track, selecting the one with the highest L^2 norm. We compare the COCOLA score in eq. (7.4) with the FAD [195, 196] metric (interpreted as a sub-FAD [9, 147]) computed with CLAP [136], EnCodec [197], and VGGish [198] backbones.

We showcase the results in table 6.8. With the FAD metrics, the model assigns the best score to the Random baseline. This behavior can be explained by considering that the Random outputs are real data, and the FAD evaluates well the perceptual quality. At the same time, it fails to assess the coherence

between the tracks, and tends to score MSDM better.

Method	FAD ↓ CLAP	FAD ↓ EnCodec	FAD ↓ VGGish	COCOLA score ↑
Slakh2100				
MSDM	0.23	92.81	2.01	3.31
CompoNet	0.30	106.23	3.20	13.50
Random	0.064	51.44	0.16	0.069
Ground Truth	-	-	-	16.57
MUSDB18-HQ				
MSDM	0.29	148.09	2.36	11.61
CompoNet	0.37	130.04	2.14	11.94
Random	0.11	100.25	0.35	4.40
Ground Truth	-	-	-	16.25

TABLE 6.8: Comparison between MSDM and CompoNet.

6.6 Conclusions

In this chapter, we have discussed a number of diffusion-based approaches to compositional music generation. In particular, we:

- (i) presented MSDM, 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 quantitatively 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.
- (ii) described GMSDI, a compositional music generation method working with any time-domain text-guided diffusion model. This method obtains reasonable generation and separation metrics on Slakh2100, enabling unsupervised compositional music generation for the first time. In future work, we want to extend the technique to latent diffusion models and narrow the gap with supervised methods.
- (iii) introduced CompoNet, a novel compositional model for music based on ControlNet that can simultaneously solve a wide range of tasks. Further, we evaluated its performance against our original baseline MSDM and showed a stark improvement.

Chapter 7

Coherence-Oriented Contrastive Learning for Audio

In this chapter, we introduce *COCOLA* (*Coherence-Oriented Contrastive Learning for Audio*), a contrastive learning method designed to capture the harmonic and rhythmic coherence between musical audio samples. *COCOLA* operates at the level of music track stems: this approach allows for the objective evaluation of compositional models in the task of accompaniment generation, providing a robust framework for assessing musical coherence. To facilitate further research and development, we release all trained models.

7.1 Introduction

In the last couple of years, there have been significant advances in music generation in the continuous domain [18, 131, 132, 138, 199], thanks to the impressive development of generative models [27, 28, 123]. In addition to producing high-quality tracks of increasing length [199], these models offer precise semantic control through textual conditioning [134, 136]. However, they fall short as tools for musical composition since they output a final mix containing all stems. To address this, a diverse range of *compositional generative models* is emerging¹ [9, 121, 122]. These models (*i*) define generative tasks at the stem level and (*ii*) might be used iteratively and interactively. Arguably, the most significant application of these models is accompaniment generation: given multiple conditioning sources (combined or not), the model generates a new set (or a mixture) of coherent stems.

The measuring problem. An important issue with this line of research is the lack of an objective metric for measuring the coherence of the generated accompaniments with respect to the provided input. In section 6.3.2, we proposed the sub-FAD metric as a multi-stem generalization of the FAD [195] protocol proposed in [147], however, this metric is not optimal for assessing coherence, as it tends to focus on global quality rather than the level of harmony and rhythm shared between accompanying stems.

Our solution. To this end, we propose a novel contrastive model called *COCOLA* (*Coherence-Oriented Contrastive Learning for Audio*), which can

¹See chapter 6.

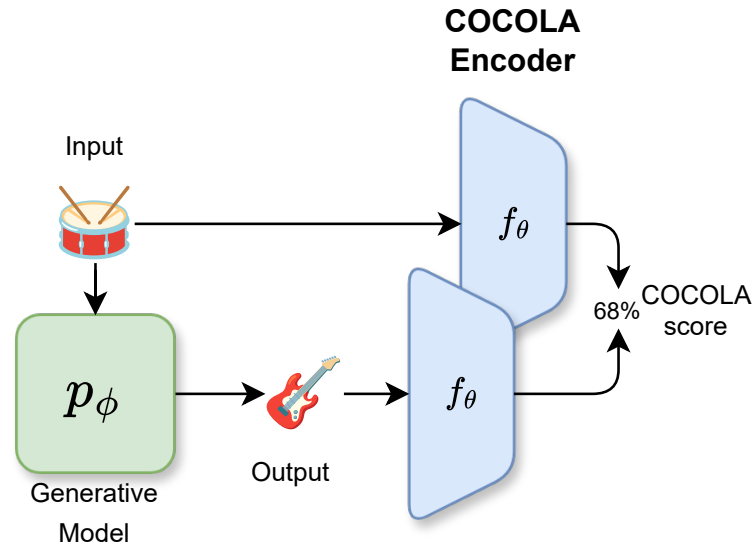


FIGURE 7.1: **Illustration of COCOLA score.** COCOLA is a contrastive model able to estimate the coherence between instrumental tracks and generated accompaniments.

evaluate the coherence between conditioning tracks and generated accompaniments (fig. 7.1). The model is trained by maximizing the agreement between disjoint sub-components of an audio window (sub-mixtures of stems) and minimizing it on sub-components belonging to different windows. With the model, we define a *COCOLA score* as the similarity between conditioning tracks and accompaniments in the embedding space.

7.2 Related Work

7.2.1 Contrastive Methods for Audio

Contrastive learning [200, 201] can be formulated both as a supervised or self-supervised problem.

Supervised approach. Supervised contrastive learning methods are typically cross-modal, requiring labeled information alongside audio data. In early works, the labeled information was in the form of simple tags, while the loss used to align embeddings of audio segments and tags was the triplet loss [202]. Within the same data setting, [203] used the contrastive loss of SimCLR [204]. With the advent of the transformer architecture [16], using complex sentences instead of simple tags became feasible. MuLaP [135] is the first model to train a common representation between audio and sentences in the musical domain. In such work, the audio and text are processed by a joint transformer encoder, conveying information about the two modalities through cross-attention layers. Although it is not a contrastive model per se, an audio-text matching loss uses negative examples to encourage the model to focus on aligned pairs. More recent

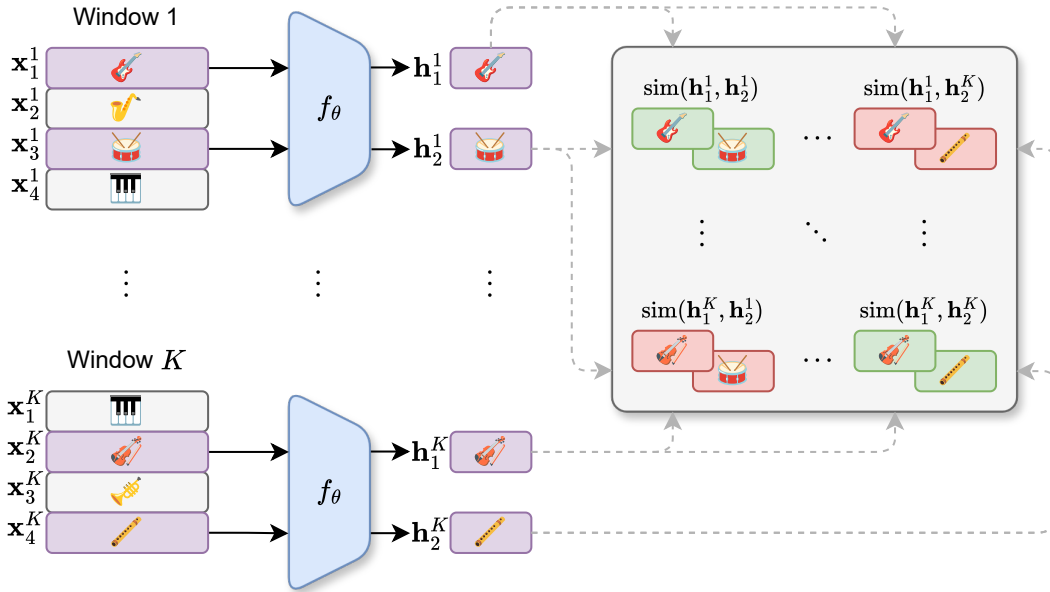


FIGURE 7.2: **The COCOLA training procedure (single stem case).** We first randomly crop windows of size L from a batch of K tracks (depicted on the left). As a second step, we randomly select two distinct stems in each window. For example, in the first window we select x_1^1 (Guitar) and x_3^1 (Drums). Thus, we embed all selected stems with the COCOLA encoder f_θ , obtaining latent representations. For example, we obtain h_1^1 and h_2^1 from the first window. Finally, we compute the contrastive loss (eq. (7.3)) considering embeddings belonging to the same window as positive pairs and combinations of embeddings between different windows as negative pairs.

works [136, 137, 146, 205], consider separate textual and audio encoders, which makes it possible to use the two branches independently at inference time.

Self-supervised approach. Self-supervised representation learning methods [206–209] build embedding spaces targeting structural information extracted from the audio data itself. In [210], the authors build positive examples for a triplet loss by augmenting with Gaussian noise, time and frequency translations, and sampling with time proximity. They also consider example mixing. While we compare coherent mixes in our method (section 7.3.1), in [210], positive pairs are not coherence-related (e.g., mixing siren and dog sounds). As in the supervised case, following [204], multi-class cross-entropy losses are employed [211–213]. In COLA [211], the authors train an embedding model with contrastive loss using the simple criterion of sampling positive pairs only from the same audio track (still employing Gaussian noise), outperforming a fully supervised baseline in a plethora of tasks. [214] pairs mixtures with sources extracted via source separation.

Interestingly, our proposed COCOLA method shares aspects of both supervised and self-supervised approaches. Given that stems are pre-separated, we cannot

consider the method purely self-supervised. At the same time, we process such data with a uni-modal encoder, as is the case for self-supervised methods.

7.3 Method

7.3.1 Stem-Level Contrastive Learning

In our setting, we have access to a dataset $D = \{\bar{\mathbf{x}}^k\}_{k=1,\dots,\bar{K}}$ containing \bar{K} musical tracks $\bar{\mathbf{x}}^k$, each separated into a variable number N of individual stems $\bar{\mathbf{x}}_n^k$, i.e., $\bar{\mathbf{x}}^k = \{\bar{\mathbf{x}}_n^k\}_{n=1,\dots,N}$. As a first step, we sample a batch of $K < \bar{K}$ tracks $\{\bar{\mathbf{x}}^k\}_{k=1,\dots,K}$ from D , with possible repetitions. Following, we slice a window \mathbf{x}^k of size L for each track $\bar{\mathbf{x}}^k$ in the batch (all stems in a window share the same length), such that no window contained in the same track overlaps for more than a ratio r , obtaining a new batch $\{\mathbf{x}^k\}_{k=1,\dots,K}$. Afterward, we select, for each k , two disjoint non-empty stem subsets X_1^k, X_2^k of \mathbf{x}^k . We define the sub-mixes \mathbf{m}_1^k and \mathbf{m}_2^k by summing the stems in X_1^k, X_2^k :

$$\mathbf{m}_1^k = \sum_{\mathbf{x}_n^k \in X_1^k} \mathbf{x}_n^k, \quad \mathbf{m}_2^k = \sum_{\mathbf{x}_n^k \in X_2^k} \mathbf{x}_n^k \quad (7.1)$$

When X_1^k, X_2^k are singletons, the sub-mixes are simply two stems in the window (single stem case). We work with sub-mixes because current compositional music generation methods [121] operate over them. Like in COLA [211], we use a convolutional audio-only encoder² $f_\theta : \mathbb{R}^L \rightarrow \mathbb{R}^d$, mapping \mathbf{m}_1^k and \mathbf{m}_2^k to lower-dimensional embedding vectors $\mathbf{h}_1^k = f_\theta(\mathbf{m}_1^k)$ and $\mathbf{h}_2^k = f_\theta(\mathbf{m}_2^k)$, with d the embedding dimension.

The COCOLA training procedure maximizes the agreement between pairs $\mathbf{h}_1^k, \mathbf{h}_2^k$ of sub-mixes embeddings in the same window. It decreases it for pairs $\mathbf{h}_1^k, \mathbf{h}_2^j$ ($j \neq k$) of sub-mixes embeddings in different windows. As in COLA, we use a bilinear similarity metric:

$$\text{sim}(\mathbf{h}_1^k, \mathbf{h}_2^j) = (\mathbf{h}_1^k)^T \mathbf{W} \mathbf{h}_2^j, \quad (7.2)$$

where \mathbf{W} is a learnable matrix. The loss we optimize is the multi-class cross entropy:

$$\mathcal{L} = - \sum_{k=1}^K \log \frac{\exp(\text{sim}(\mathbf{h}_1^k, \mathbf{h}_2^k))}{\sum_{j=1}^K \exp(\text{sim}(\mathbf{h}_1^k, \mathbf{h}_2^j))}. \quad (7.3)$$

We depict the training procedure of COCOLA in fig. 7.2 for the single stem case.

²In our notation, we incorporate into f_θ any domain transform preceding or following the convolutional network operations, like the (pre) mel-filterbank map and the (post) projection head g in COLA.

NOTE:

In the COLA training procedure, the positive pairs are (fully mixed) windows belonging to the same track. In COCOLA, they are sub-mixes belonging to the same window. As such, we allow for negative pairs belonging to the same track but in different windows. The r ratio has to be chosen well to avoid strong overlaps between windows in the same track. In that case, we could potentially consider (nearly) coherent sub-mixes as negative pairs.

7.3.2 The COCOLA Score

Equipped with the encoder f_θ , we can quantify the coherence of the accompaniments generated by a generative model $p_\phi(\mathbf{x} | \mathbf{y})$, where \mathbf{y} is the conditioning variable and \mathbf{x} is the modeled variable; respectively, input and output of the generative model. The model’s variables can be either a set of stems or sub-mixes. Given the input \mathbf{y} , the model p_ϕ generates an output $\tilde{\mathbf{x}} \sim p_\phi(\mathbf{x} | \mathbf{y})$. We can compute the coherence between \mathbf{y} and $\tilde{\mathbf{x}}$ by first embedding the two vectors $\mathbf{h}_\mathbf{y} = f_\theta(\mathbf{y})$ and $\mathbf{h}_{\tilde{\mathbf{x}}} = f_\theta(\tilde{\mathbf{x}})$ (summing the stems beforehand if considering a set of stems). We define the *COCOLA score* between \mathbf{x} and $\tilde{\mathbf{y}}$ as:

$$\text{COCOLA score}(\mathbf{y}, \tilde{\mathbf{x}}) = \text{sim}(\mathbf{h}_\mathbf{y}, \mathbf{h}_{\tilde{\mathbf{x}}}), \quad (7.4)$$

the similarity (eq. (7.2)) between their embeddings. The described procedure is depicted in fig. 7.1.

7.4 Experimental Setup

Datasets. For training and evaluating COCOLA, we use four public stem-separated datasets: MUSDB18-HQ [40], MoisesDB³ [42], Slakh2100 and Coco-Chorales⁴ [46]. These datasets are quite different from each other; some are synthesized and offer large volumes of clean—albeit less diverse—music, while others consist of real studio recordings but are significantly smaller in size. For a more in-depth discussion see section 2.3.

Model implementation. To implement the COCOLA encoder f_θ , we follow [211] and employ the EfficientNet-B0 [215] convolutional architecture followed by a linear projection layer, operating on the mel-filterbank audio representation. The embedding dimension is 512. With respect to the original baseline, we add a 0.1 dropout on the EfficientNet layers.

Training details. All COCOLA models are trained on an NVIDIA RTX 4070 Super with 12GB of VRAM. Each training batch contains 32 audio chunks of 5s (16kHz). We set the maximum window overlap ratio $r = 50\%$ and train with the Adam optimizer [216] with a 10^{-3} learning rate. We add Gaussian

³Not having pre-computed splits, we set a custom 0.8 (train) / 0.1 (validation) / 0.1 (test) split.

⁴In our experiments we use the tiny version, comprising 4000 tracks.

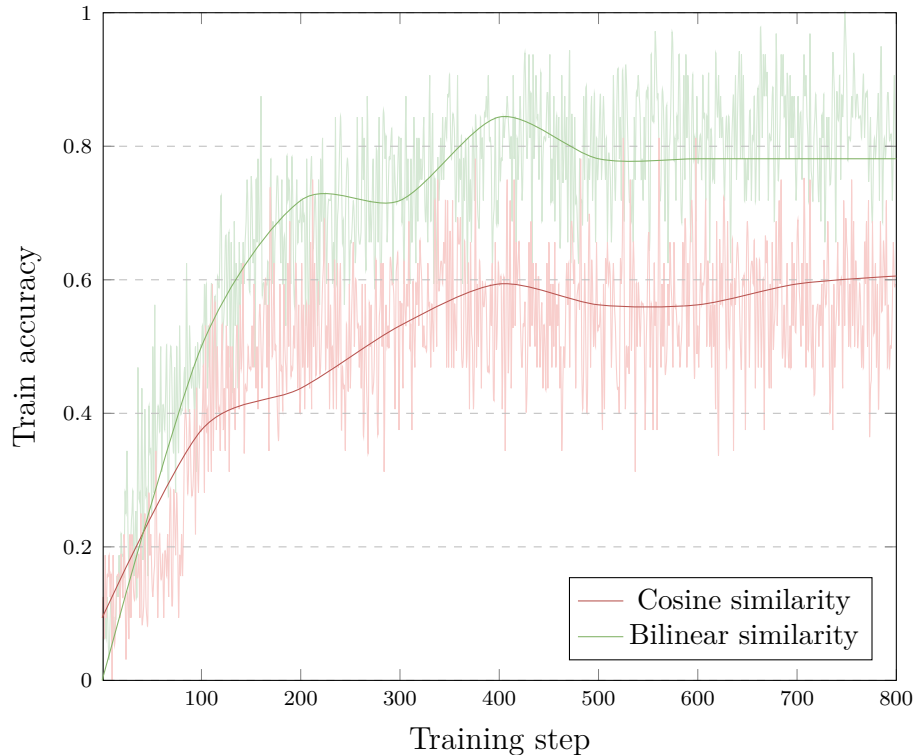


FIGURE 7.3: **Cosine vs bilinear similarity during training.** Training is performed with “COCOLA All”. The accuracy metric is defined in eq. (7.5). We adopt the bilinear similarity in eq. (7.2), given its improved performance.

noise to positive samples as a data augmentation method, with $\sigma = 10^{-3}$. We experimented training with cosine similarity [204] but reported (fig. 7.3) lower performance, corroborating [211].

7.5 Experimental Results

We employ four COCOLA encoder models in our experiments: “COCOLA MoisesDB”, “COCOLA Slakh2100”, “COCOLA CocoChorales” and “COCOLA All”. The first three are trained on the homonym datasets, while the last one is trained on all three combined. For the “COCOLA CocoChorales” we use all ensables while on “COCOLA All” we use only the Random ensemble for a more balanced partitioning, with respect to the other datasets. MUSDB18-HQ, being the smallest dataset, is used as a held-out test dataset for studying generalization.

7.5.1 Coherent Sub-Mix Classification

We cross-test the performance of all COCOLA models, performing classification of coherent pairs on the test split of our datasets. More specifically, given an encoder f_θ , we iterate a test set, collecting at each step a batch of K windows $\mathbf{x}^1, \dots, \mathbf{x}^K$. Following the steps in section 7.3.1 we compute all similarities

$\text{sim}(\mathbf{h}_1^k, \mathbf{h}_2^j)$ for $k, j \in [K]$. We define the accuracy over a batch as:

$$\frac{1}{K} \sum_{k=1}^K \mathbb{1} \left(k = \arg \max_{j \in [K]} \text{sim}(\mathbf{h}_1^k, \mathbf{h}_2^j) \right), \quad (7.5)$$

where $\mathbb{1}$ is the indicator function. We obtain the final accuracy averaging over all batches in the dataset. For our evaluation we use $K = 2$ and depict in table 7.1 the results across all combinations of models and test datasets. While both ‘‘COCOLA MoisesDB’’ and ‘‘COCOLA Slakh2100’’ perform only slightly better than a random choice, ‘‘COCOLA CocoChorales’’ features improved performance. Finally, combining the three dataset, we obtain an accuracy of over 90% on all datasets, showcasing generalization with 90.43% on the held-out MUSDB18-HQ.

Train Dataset	Test Dataset			
	MUSDB18-HQ	MoisesDB	Slakh2100	CocoChorales
MoisesDB [42]	52.56%	53.01%	51.22%	60.32%
Slakh2100 [44]	53.06%	53.58%	53.78%	59.35%
CocoChorales [46]	70.10%	61.48%	67.50%	99.78%
All	90.43%	93.06%	90.06%	99.89%

TABLE 7.1: **Classification accuracy.** Results on various test sets with COCOLA models using $K = 2$ sub-mixture test pairs. MUSDB18-HQ is used as a hold-out test dataset.

7.6 Conclusion

In this chapter, we proposed COCOLA: a contrastive encoder for recognizing the coherence between musical stems. Then, we evaluated different audio datasets with our model COCOLA and found it adept at evaluating accompaniment coherence. Indeed, such a model could help future research on compositional music generation by providing a specialized quantitative metric on accompaniment quality.

Future work. We plan to improve the quality of COCOLA by training on additional stem-level datasets [217] or using data obtained by pre-separating [147] larger realistic music datasets [218]. In future work, we would also like to explore inference-side methods that can guide the diffusion process using COCOLA as a likelihood function, offering an alternative (or additional) loss for the GMSDI method.

Chapter 8

Conclusions

In this thesis, we covered various approaches and architectures to harness the generative powers for solving tasks from different domains: reasoning, images, and music. The common ground among all these fields is the exploration of various approaches and procedures to harness (and sometimes measure) the generative capabilities of the current state-of-the-art generative models.

In the reasoning and language domain, we have seen how the collaborative effort of a generative model with a discriminative interpreter can be used to improve a property deduction task. This was developed in conjunction with a toy reasoning environment composed of simple sequences of geometric shapes and logical properties. This line of research was further explored with our participation in a collaborative benchmark on Large Language Models: for this benchmark, we proposed a task inspired by our aforementioned reasoning environment.

In source separation, we developed several procedures that use (possibly pre-trained) generative models. In particular, we have seen: (i) an approach that is able to separate two mixed continuous signals using a sparse estimated joint probability over latent discrete codes. (ii) A diffusion-based method that separates musical mixture by performing constrained generation over the sources' joint probability. (iii) A fine-tuned approach that learns—amid other things—how to disentangle the single sources from a provided mixture.

For music production, we have investigated several approaches to tackle the generation of coherent audio sources, possibly conditioned on some provided musical tracks. We were amongst the first researchers that approached this compositional music direction in the waveform domain, thanks to our multi-source diffusion architecture [9]. We further explored this line of research by developing an inference time procedure to exploit pre-trained diffusion models on music. Finally, we approached the task of compositional music generation in a semi-supervised approach by fine-tuning an existing audio diffusion architecture over several possible music accompaniment generation tasks.

In conclusion, this thesis spans several methodologies and useful tools to aid the utilization of (often pre-trained) generative architecture. This is a valuable effort since generative artificial intelligence utilization is rapidly increasing in both the industry and academic domains.

Bibliography

- [1] K. Schwab, *The Fourth Industrial Revolution*. USA: Crown Publishing Group, 2017.
- [2] A. R. Chow and B. Perrigo, “The ai arms race is changing everything,” 20203.
- [3] G. Krogh, “Artificial intelligence in organizations: New opportunities for phenomenon-based theorizing,” *Academy of Management Discoveries*, vol. 4, pp. 404–409, 12 2018.
- [4] X. Yang, Z. Song, I. King, and Z. Xu, “A survey on deep semi-supervised learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, p. 8934–8954, Sept. 2023.
- [5] J. E. van Engelen and H. H. Hoos, “A survey on semi-supervised learning,” *Machine Learning*, vol. 109, pp. 373 – 440, 2019.
- [6] A. Norelli, G. Mariani, L. Moschella, A. Santilli, G. Parascandolo, S. Melzi, and E. Rodolà, “Explanatory learning: Beyond empiricism in neural networks,” *CoRR*, vol. abs/2201.10222, 2022.
- [7] A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso, A. Kluska, A. Lewkowycz, A. Agarwal, A. Power, A. Ray, A. Warstadt, A. W. Kocurek, A. Safaya, A. Tazarv, A. Xiang, A. Parrish, A. Nie, A. Hussain, A. Askeel, A. Dsouza, A. Slone, A. Rahane, A. S. Iyer, A. Andreassen, A. Madotto, A. Santilli, A. Stuhlmüller, A. Dai, A. La, A. Lampinen, A. Zou, A. Jiang, A. Chen, A. Vuong, A. Gupta, A. Gottardi, A. Norelli, A. Venkatesh, A. Gholamidavoodi, A. Tabassum, A. Menezes, A. Kirubaranjan, A. Mullokandov, A. Sabharwal, A. Herrick, A. Efrat, A. Erdem, A. Karakaş, B. R. Roberts, B. S. Loe, B. Zoph, B. Bojanowski, B. Özyurt, B. Hedayatnia, B. Neyshabur, B. Inden, B. Stein, B. Ekmekci, B. Y. Lin, B. Howald, B. Orinion, C. Diao, C. Dour, C. Stinson, C. Argueta, C. F. Ramírez, C. Singh, C. Rathkopf, C. Meng, C. Baral, C. Wu, C. Callison-Burch, C. Waites, C. Voigt, C. D. Manning, C. Potts, C. Ramirez, C. E. Rivera, C. Siro, C. Raffel, C. Ashcraft, C. Garbacea, D. Sileo, D. Garrette, D. Hendrycks, D. Kilman, D. Roth, D. Freeman, D. Khashabi, D. Levy, D. M. González, D. Perszyk, D. Hernandez, D. Chen, D. Ippolito, D. Gilboa, D. Dohan, D. Drakard, D. Jurgens, D. Datta, D. Ganguli, D. Emelin, D. Kleyko, D. Yuret, D. Chen, D. Tam, D. Hupkes, D. Misra, D. Buzan, D. C. Mollo, D. Yang, D.-H. Lee, D. Schrader, E. Shutova, E. D. Cubuk, E. Segal, E. Hagerman, E. Barnes, E. Donoway, E. Pavlick,

E. Rodola, E. Lam, E. Chu, E. Tang, E. Erdem, E. Chang, E. A. Chi, E. Dyer, E. Jerzak, E. Kim, E. E. Manyasi, E. Zheltonozhskii, F. Xia, F. Siar, F. Martínez-Plumed, F. Happé, F. Chollet, F. Rong, G. Mishra, G. I. Winata, G. de Melo, G. Kruszewski, G. Parascandolo, G. Mariani, G. Wang, G. Jaimovitch-López, G. Betz, G. Gur-Ari, H. Galijasevic, H. Kim, H. Rashkin, H. Hajishirzi, H. Mehta, H. Bogar, H. Shevlin, H. Schütze, H. Yakura, H. Zhang, H. M. Wong, I. Ng, I. Noble, J. Jumelet, J. Geissinger, J. Kernion, J. Hilton, J. Lee, J. F. Fisac, J. B. Simon, J. Koppel, J. Zheng, J. Zou, J. Kocoń, J. Thompson, J. Wingfield, J. Kaplan, J. Radom, J. Sohl-Dickstein, J. Phang, J. Wei, J. Yosinski, J. Novikova, J. Bosscher, J. Marsh, J. Kim, J. Taal, J. Engel, J. Alabi, J. Xu, J. Song, J. Tang, J. Waweru, J. Burden, J. Miller, J. U. Balis, J. Batchelder, J. Berant, J. Frohberg, J. Rozen, J. Hernandez-Orallo, J. Boudeman, J. Guerr, J. Jones, J. B. Tenenbaum, J. S. Rule, J. Chua, K. Kanclerz, K. Livescu, K. Krauth, K. Gopalakrishnan, K. Ignatyeva, K. Markert, K. D. Dhole, K. Gimpel, K. Omondi, K. Mathewson, K. Chiafullo, K. Shkaruta, K. Shridhar, K. McDonell, K. Richardson, L. Reynolds, L. Gao, L. Zhang, L. Dugan, L. Qin, L. Contreras-Ochando, L.-P. Morency, L. Moschella, L. Lam, L. Noble, L. Schmidt, L. He, L. O. Colón, L. Metz, L. K. Şenel, M. Bosma, M. Sap, M. ter Hoeve, M. Farooqi, M. Faruqui, M. Mazeika, M. Baturan, M. Marelli, M. Maru, M. J. R. Quintana, M. Tolkiehn, M. Giulianelli, M. Lewis, M. Potthast, M. L. Leavitt, M. Hagen, M. Schubert, M. O. Baitemirova, M. Arnaud, M. McElrath, M. A. Yee, M. Cohen, M. Gu, M. Ivanitskiy, M. Starritt, M. Strube, M. Swędrowski, M. Bevilacqua, M. Yasunaga, M. Kale, M. Cain, M. Xu, M. Suzgun, M. Walker, M. Tiwari, M. Bansal, M. Aminnaseri, M. Geva, M. Gheini, M. V. T, N. Peng, N. A. Chi, N. Lee, N. G.-A. Krakover, N. Cameron, N. Roberts, N. Doiron, N. Martinez, N. Nangia, N. Deckers, N. Muennighoff, N. S. Keskar, N. S. Iyer, N. Constant, N. Fiedel, N. Wen, O. Zhang, O. Agha, O. Elbaghdadi, O. Levy, O. Evans, P. A. M. Casares, P. Doshi, P. Fung, P. P. Liang, P. Vicol, P. Alipoormolabashi, P. Liao, P. Liang, P. Chang, P. Eckersley, P. M. Htut, P. Hwang, P. Miłkowski, P. Patil, P. Pezeshkpour, P. Oli, Q. Mei, Q. Lyu, Q. Chen, R. Banjade, R. E. Rudolph, R. Gabriel, R. Habacker, R. Risco, R. Millièrè, R. Garg, R. Barnes, R. A. Saurous, R. Arakawa, R. Raymaekers, R. Frank, R. Sikand, R. Novak, R. Sitelew, R. LeBras, R. Liu, R. Jacobs, R. Zhang, R. Salakhutdinov, R. Chi, R. Lee, R. Stovall, R. Teehan, R. Yang, S. Singh, S. M. Mohammad, S. Anand, S. Dillavou, S. Shleifer, S. Wiseman, S. Gruetter, S. R. Bowman, S. S. Schoenholz, S. Han, S. Kwatra, S. A. Rous, S. Ghazarian, S. Ghosh, S. Casey, S. Bischoff, S. Gehrmann, S. Schuster, S. Sadeghi, S. Hamdan, S. Zhou, S. Srivastava, S. Shi, S. Singh, S. Asaadi, S. S. Gu, S. Pachchigar, S. Toshniwal, S. Upadhyay, Shyamolima, Debnath, S. Shakeri, S. Thormeyer, S. Melzi, S. Reddy, S. P. Makini, S.-H. Lee, S. Torene, S. Hatwar, S. Dehaene, S. Divic, S. Ermon, S. Biderman, S. Lin, S. Prasad, S. T. Piantadosi, S. M. Shieber, S. Mishnerghi, S. Kiritchenko, S. Mishra, T. Linzen, T. Schuster, T. Li, T. Yu, T. Ali, T. Hashimoto, T.-L. Wu, T. Desbordes, T. Rothschild,

- T. Phan, T. Wang, T. Nkinyili, T. Schick, T. Kornev, T. Tunduny, T. Gerstenberg, T. Chang, T. Neeraj, T. Khot, T. Shultz, U. Shaham, V. Misra, V. Demberg, V. Nyamai, V. Raunak, V. Ramasesh, V. U. Prabhu, V. Padmakumar, V. Srikumar, W. Fedus, W. Saunders, W. Zhang, W. Vossen, X. Ren, X. Tong, X. Zhao, X. Wu, X. Shen, Y. Yaghoobzadeh, Y. Lakretz, Y. Song, Y. Bahri, Y. Choi, Y. Yang, Y. Hao, Y. Chen, Y. Belinkov, Y. Hou, Y. Hou, Y. Bai, Z. Seid, Z. Zhao, Z. Wang, Z. J. Wang, Z. Wang, and Z. Wu, “Beyond the imitation game: Quantifying and extrapolating the capabilities of language models,” 2023.
- [8] E. Postolache, G. Mariani, M. Mancusi, A. Santilli, L. Cosmo, and E. Rodolà, “Latent autoregressive source separation,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 9444–9452, Jun. 2023.
- [9] G. Mariani, I. Tallini, E. Postolache, M. Mancusi, L. Cosmo, and E. Rodolà, “Multi-source diffusion models for simultaneous music generation and separation,” *arXiv preprint arXiv:2302.02257*, 2023.
- [10] E. Postolache, G. Mariani, L. Cosmo, E. Benetos, and E. 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)*, pp. 6980–6984, 2024.
- [11] R. Ciranni, E. Postolache, G. Mariani, M. Mancusi, L. Cosmo, and E. Rodolà, “Cocola: Coherence-oriented contrastive learning of musical audio representations,” 2024.
- [12] G. Kim and J. C. Ye, “Diffusionclip: Text-guided image manipulation using diffusion models,” *CoRR*, vol. abs/2110.02711, 2021.
- [13] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *Proc. ICLR*, 2014.
- [14] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, “Neural discrete representation learning,” in *Proc. NeurIPS*, 2017.
- [15] P. Esser, R. Rombach, and B. Ommer, “Taming transformers for high-resolution image synthesis,” in *Proc. CVPR*, pp. 12873–12883, 2021.
- [16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *Proc. NeurIPS*, vol. 30, 2017.
- [17] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, “Pixel recurrent neural networks,” *CoRR*, vol. abs/1601.06759, 2016.
- [18] A. Agostinelli, T. I. Denk, Z. Borsos, J. Engel, M. Verzetti, A. Caillon, Q. Huang, A. Jansen, A. Roberts, M. Tagliasacchi, *et al.*, “Musiclm: Generating music from text,” *arXiv preprint arXiv:2301.11325*, 2023.

- [19] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” 2016.
- [20] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, “Jukebox: A generative model for music,” *arXiv preprint arXiv:2005.00341*, 2020.
- [21] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, *et al.*, “Conditional image generation with pixelcnn decoders,” *Proc. NeurIPS*, vol. 29, 2016.
- [22] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, “Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications,” *arXiv preprint arXiv:1701.05517*, 2017.
- [23] W. Kool, H. van Hoof, and M. Welling, “Ancestral gumbel-top-k sampling for sampling without replacement,” *Journal of Machine Learning Research*, vol. 21, no. 47, pp. 1–36, 2020.
- [24] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, “The curious case of neural text degeneration,” in *Proc. ICLR*, 2020.
- [25] D. R. Reddy *et al.*, “Speech understanding systems: A summary of results of the five-year research effort,” *Department of Computer Science. Carnegie-Mell University, Pittsburgh, PA*, vol. 17, p. 138, 1977.
- [26] Y. Song and S. Ermon, “Generative modeling by estimating gradients of the data distribution,” in *Advances in Neural Information Processing Systems*, pp. 11895–11907, 2019.
- [27] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, “Score-based generative modeling through stochastic differential equations,” in *Proc. ICLR*, 2021.
- [28] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” *Proc. NeurIPS*, vol. 33, pp. 6840–6851, 2020.
- [29] T. Karras, M. Aittala, T. Aila, and S. Laine, “Elucidating the design space of diffusion-based generative models,” in *Advances in Neural Information Processing Systems*, 2022.
- [30] A. Hyvärinen, “Estimation of non-normalized statistical models by score matching,” *Journal of Machine Learning Research*, vol. 6, no. 24, pp. 695–709, 2005.
- [31] D. P. Kingma and Y. LeCun, “Regularized estimation of image statistics by score matching,” in *Advances in Neural Information Processing Systems* (J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, eds.), vol. 23, Curran Associates, Inc., 2010.

- [32] P. Vincent, “A connection between score matching and denoising autoencoders,” *Neural Computation*, vol. 23, no. 7, pp. 1661–1674, 2011.
- [33] J. Ho and T. Salimans, “Classifier-free diffusion guidance,” in *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.
- [34] G. Sanchez, H. Fan, A. Spangher, E. Levi, P. S. Ammanamanchi, and S. Biderman, “Stay on topic with classifier-free guidance,” *arXiv preprint arXiv:2306.17806*, 2023.
- [35] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [36] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep learning face attributes in the wild,” in *Proc. ICCV*, December 2015.
- [37] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of GANs for improved quality, stability, and variation,” in *Proc. ICLR*, 2018.
- [38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *Proc. CVPR*, pp. 248–255, 2009.
- [39] G. A. Miller, “WordNet: A lexical database for English,” in *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.
- [40] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, “Musdb18-hq - an uncompressed version of musdb18,” Aug. 2019.
- [41] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, “The MUSDB18 corpus for music separation,” Dec. 2017.
- [42] I. G. Pereira, F. Araujo, F. Korzeniowski, and R. Vogl, “Moisesdb: A dataset for source separation beyond 4 stems,” in *Ismir 2023 Hybrid Conference*, 2023.
- [43] D. Bogdanov, M. Won, P. Tovstogan, A. Porter, and X. Serra, “The mtg-jamendo dataset for automatic music tagging,” in *International Conference on Machine Learning*, 2019.
- [44] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, “Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity,” in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, IEEE, 2019.
- [45] C. Raffel, *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, Columbia University, USA, 2016.

- [46] Y. Wu, J. Gardner, E. Manilow, I. Simon, C. Hawthorne, and J. Engel, “The chamber ensemble generator: Limitless high-quality mir data via generative modeling,” *arXiv preprint arXiv:2209.14458*, 2022.
- [47] Y. Wu, E. Manilow, Y. Deng, R. Swavely, K. Kastner, T. Cooijmans, A. Courville, C.-Z. A. Huang, and J. Engel, “Midi-ddsp: Detailed control of musical performance via hierarchical modeling,” in *International Conference on Learning Representations*, 2021.
- [48] J. McConnell, “Memory transfer through cannibalism in planarians,” *J. Neuropsychiat.*, vol. 3, pp. 542–548, 1962.
- [49] A. H. Taylor, R. Miller, and R. D. Gray, “New caledonian crows reason about hidden causal agents,” *PNAS*, vol. 109, no. 40, pp. 16389–16391, 2012.
- [50] K. Heath, “Zendo,” 2001.
- [51] K. Popper, *The Logic of Scientific Discovery*. Julius Springer, Hutchinson & Co, 1935.
- [52] L. E. Schulz, A. Gopnik, and C. Glymour, “Preschool children learn about causal structure from conditional interventions,” *Developmental science*, vol. 10, no. 3, pp. 322–332, 2007.
- [53] D. Angluin, “Learning regular sets from queries and counterexamples,” *Inf. Comput.*, vol. 75, p. 87–106, Nov. 1987.
- [54] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick, “CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning,” in *Proc. CVPR*, pp. 1988–1997, 2017.
- [55] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, “Learning transferable visual models from natural language supervision,” in *Proc. ICML*, 2021.
- [56] T. M. Hospedales, A. Antoniou, P. Micaelli, and A. J. Storkey, “Meta-learning in neural networks: A survey,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, may 2020.
- [57] E. Y. Shapiro, *Inductive inference of theories from facts*. Yale University, Department of Computer Science, 1981.
- [58] M. Balog, A. Gaunt, M. Brockschmidt, S. Nowozin, and D. Tarlow, “Deepcoder: Learning to write programs,” in *5th International Conference on Learning Representations, ICLR 2017-Conference Track Proceedings*, 2017.
- [59] M. Blum, P. Feldman, and S. Micali, “Non-interactive zero-knowledge and its applications,” in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC ’88, (New York, NY, USA), p. 103–112, Association for Computing Machinery, 1988.

- [60] J. Pearl, “Radical empiricism and machine learning research,” *Journal of Causal Inference*, vol. 9, no. 1, pp. 78–82, 2021.
- [61] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS’17, (Red Hook, NY, USA), p. 6000–6010, Curran Associates Inc., 2017.
- [62] D. Hofstadter, *Gödel, Escher, Bach: an eternal golden braid*, vol. 13. Basic books New York, 1979.
- [63] G. Rota, “The pernicious influence of mathematics upon philosophy,” *Synthese*, vol. 88, no. 2, pp. 165–178, 1991.
- [64] T. M. Mitchell, *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research, 1980.
- [65] G. Goh, N. Cammarata, C. Voss, S. Carter, M. Petrov, L. Schubert, A. Radford, and C. Olah, “Multimodal neurons in artificial neural networks,” *Distill*, 2021. <https://distill.pub/2021/multimodal-neurons>.
- [66] L. A. Hendricks, Z. Akata, M. Rohrbach, J. Donahue, B. Schiele, and T. Darrell, “Generating visual explanations,” in *European conference on computer vision*, pp. 3–19, Springer, 2016.
- [67] M. Hind, D. Wei, M. Campbell, N. C. Codella, A. Dhurandhar, A. Majsilović, K. Natesan Ramamurthy, and K. R. Varshney, “Ted: Teaching ai to explain its decisions,” in *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 123–129, 2019.
- [68] C. Rudin, “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead,” *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.
- [69] A. Meinke and M. Hein, “Towards neural networks that provably know when they don’t know,” *arXiv preprint arXiv:1909.12180*, 2019.
- [70] Y. LeCun, “The epistemology of deep learning.” *Institute for Advanced Studies* <https://www.ias.edu/sites/default/files/video/lecun-ias-20190222.pdf>; <https://youtu.be/gG5NckMerHU>, 2019. Accessed: 2021–10–04.
- [71] R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” *CoRR*, vol. abs/1508.07909, 2015.
- [72] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, “Language models are few-shot learners,” *Proc. NeurIPS*, vol. 33, pp. 1877–1901, 2020.
- [73] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, “Jukebox: A generative model for music,” 2020.

- [74] A. Razavi, A. van den Oord, and O. Vinyals, “Generating diverse high-fidelity images with VQ-VAE-2,” in *Proc. NeurIPS*, 2019.
- [75] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, “Zero-shot text-to-image generation,” in *Proc. ICML*, pp. 8821–8831, PMLR, 2021.
- [76] J. Yu, Y. Xu, J. Y. Koh, T. Luong, G. Baid, Z. Wang, V. Vasudevan, A. Ku, Y. Yang, B. K. Ayan, *et al.*, “Scaling autoregressive models for content-rich text-to-image generation,” *arXiv preprint arXiv:2206.10789*, 2022.
- [77] R. Castellon, C. Donahue, and P. Liang, “Codified audio language modeling learns useful representations for music information retrieval,” *arXiv preprint arXiv:2107.05677*, 2021.
- [78] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, “Finetuned language models are zero-shot learners,” *CoRR*, vol. abs/2109.01652, 2021.
- [79] V. Sanh, A. Webson, C. Raffel, *et al.*, “Multitask prompted training enables zero-shot task generalization,” in *Proc. ICLR*, 2022.
- [80] H. Yang, J. Lin, A. Yang, P. Wang, C. Zhou, and H. Yang, “Prompt tuning for generative multimodal pretrained models,” *arXiv preprint arXiv:2208.02532*, 2022.
- [81] A. Hertz, R. Mokady, J. Tenenbaum, K. Aberman, Y. Pritch, and D. Cohen-Or, “Prompt-to-prompt image editing with cross attention control,” *arXiv preprint arXiv:2208.01626*, 2022.
- [82] S. Dovrat, E. Nachmani, and L. Wolf, “Many-speakers single channel speech separation with optimal permutation training,” in *Interspeech*, 2021.
- [83] A. Défossez, “Hybrid spectrogram and waveform source separation,” in *Proceedings of the ISMIR 2021 Workshop on Music Source Separation*, 2021.
- [84] S. Wisdom, H. Erdogan, D. P. W. Ellis, R. Serizel, N. Turpault, E. Fonseca, J. Salamon, P. Seetharaman, and J. R. Hershey, “What’s all the fuss about free universal sound separation data?,” in *Proc. ICASSP*, pp. 186–190, 2021.
- [85] E. Postolache, J. Pons, S. Pascual, and J. Serrà, “Adversarial permutation invariant training for universal sound separation,” *arXiv preprint arXiv:2210.12108*, 2022.
- [86] T. Halperin, A. Ephrat, and Y. Hoshen, “Neural separation of observed and unobserved distributions,” *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 4548–4557, 2019.

- [87] Y. Luo and N. Mesgarani, “Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation,” *IEEE/ACM transactions on audio, speech, and language processing*, vol. 27, no. 8, pp. 1256–1266, 2019.
- [88] A. Défossez, N. Usunier, L. Bottou, and F. Bach, “Music Source Separation in the Waveform Domain,” *arXiv:1911.13254 [cs, eess, stat]*, 2019. arXiv: 1911.13254.
- [89] S. Wisdom, E. Tzinis, H. Erdogan, R. Weiss, K. Wilson, and J. Hershey, “Unsupervised sound separation using mixture invariant training,” in *Proc. NeurIPS*, vol. 33, pp. 3846–3857, 2020.
- [90] P. Comon, “Independent Component Analysis, a new concept?,” *Signal Processing*, 1994.
- [91] A. Hyvärinen and E. Oja, “Independent component analysis: algorithms and applications,” *Neural networks*, vol. 13, no. 4-5, pp. 411–430, 2000.
- [92] P.-S. Huang, S. D. Chen, P. Smaragdis, and M. Hasegawa-Johnson, “Singing-voice separation from monaural recordings using robust principal component analysis,” in *Proc. ICASSP*, pp. 57–60, IEEE, 2012.
- [93] P. Smaragdis, C. Févotte, G. J. Mysore, N. Mohammadiha, and M. Hoffman, “Static and dynamic source separation using nonnegative factorizations: A unified view,” *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 66–75, 2014.
- [94] Z. Raffi and B. Pardo, “Repeating pattern extraction technique (repet): A simple method for music/voice separation,” *IEEE transactions on audio, speech, and language processing*, vol. 21, no. 1, pp. 73–84, 2012.
- [95] E. Gusó, J. Pons, S. Pascual, and J. Serrà, “On loss functions and evaluation metrics for music source separation,” in *Proc. ICASSP*, pp. 306–310, 2022.
- [96] S. T. Roweis, “One microphone source separation,” in *Proc. NIPS*, 2000.
- [97] S. Uhlich, F. Giron, and Y. Mitsufuji, “Deep neural network based instrument extraction from music,” in *Proc. ICASSP*, 2015.
- [98] P.-S. Huang, M. Kim, M. A. Hasegawa-Johnson, and P. Smaragdis, “Singing-voice separation from monaural recordings using deep recurrent neural networks,” in *Proc. ISMIR*, 2014.
- [99] A. A. Nugraha, A. Liutkus, and E. Vincent, “Multichannel audio source separation with deep neural networks,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 9, pp. 1652–1664, 2016.
- [100] J.-Y. Liu and Y.-H. Yang, “Denoising auto-encoder with recurrent skip connections and residual regression for music source separation,” 2018.

- [101] N. Takahashi, N. Goswami, and Y. Mitsufuji, “Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation,” in *Proc. IWAENC*, pp. 106–110, 2018.
- [102] F. Lluís, J. Pons, and X. Serra, “End-to-end music source separation: Is it possible in the waveform domain?,” in *INTERSPEECH*, pp. 4619–4623, 2019.
- [103] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” *Proc. NIPS*, vol. 27, 2014.
- [104] Y. C. Subakan and P. Smaragdis, “Generative adversarial source separation,” in *Proc. ICASSP*, pp. 26–30, IEEE, 2018.
- [105] Q. Kong, Y. Xu, W. Wang, P. J. B. Jackson, and M. D. Plumbley, “Single-channel signal separation and deconvolution with generative adversarial networks,” in *Proc. IJCAI*, p. 2747–2753, AAAI Press, 2019.
- [106] V. Narayanaswamy, J. J. Thiagarajan, R. Anirudh, and A. Spanias, “Unsupervised audio source separation using generative priors,” 2020.
- [107] V. Jayaram and J. Thickstun, “Source separation with deep generative priors,” in *Proc. ICML*, PMLR, 2020.
- [108] G. Parisi, “Correlation functions and computer simulations,” *Nuclear Physics B*, vol. 180, no. 3, pp. 378–384, 1981.
- [109] V. Jayaram and J. Thickstun, “Parallel and flexible sampling from autoregressive models via langevin dynamics,” in *Proc. ICML*, pp. 4807–4818, PMLR, 2021.
- [110] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [111] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, “Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity,” in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, IEEE, 2019.
- [112] A. Horé and D. Ziou, “Image quality metrics: Psnr vs. ssim,” in *Proc. ICPR*, pp. 2366–2369, 2010.
- [113] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” in *Proc. NeurIPS*, vol. 30, 2017.
- [114] T. Dockhorn, A. Vahdat, and K. Kreis, “Score-based generative modeling with critically-damped langevin diffusion,” *ArXiv*, vol. abs/2112.07068, 2021.

- [115] P. Seetharaman, F. Pishdadian, and B. Pardo, “Music/voice separation using the 2d fourier transform,” in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 36–40, IEEE, 2017.
- [116] F.-R. Stöter, A. Liutkus, and N. Ito, “The 2018 signal separation evaluation campaign,” in *Proc. LVA/ICA*, pp. 293–305, 2018.
- [117] N. Takahashi, S. Parthasaarathy, N. Goswami, and Y. Mitsufuji, “Recursive speech separation for unknown number of speakers,” *arXiv preprint arXiv:1904.03065*, 2019.
- [118] J. Yu, X. Li, J. Y. Koh, H. Zhang, R. Pang, J. Qin, A. Ku, Y. Xu, J. Baldrige, and Y. Wu, “Vector-quantized image modeling with improved vqgan,” *arXiv preprint arXiv:2110.04627*, 2021.
- [119] Y. Xu, Y. Song, S. Garg, L. Gong, R. Shu, A. Grover, and S. Ermon, “Anytime sampling for autoregressive models via ordered autoencoding,” *arXiv preprint arXiv:2102.11495*, 2021.
- [120] L. Zhang, A. Rao, and M. Agrawala, “Adding conditional control to text-to-image diffusion models,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3836–3847, 2023.
- [121] J. D. Parker, J. Spijkervet, K. Kosta, F. Yesiler, B. Kuznetsov, J.-C. Wang, M. Avent, J. Chen, and D. Le, “Stemgen: A music generation model that listens,” in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1116–1120, IEEE, 2024.
- [122] B. Han, J. Dai, X. Song, W. Hao, X. He, D. Guo, J. Chen, Y. Wang, and Y. Qian, “Instructme: An instruction guided music edit and remix framework with latent diffusion models,” *arXiv preprint arXiv:2308.14360*, 2023.
- [123] OpenAI, “Gpt-4 technical report,” 2023.
- [124] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, *et al.*, “Llama: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023.
- [125] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, “Hierarchical text-conditional image generation with clip latents,” *arXiv preprint arXiv:2204.06125*, 2022.
- [126] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-resolution image synthesis with latent diffusion models,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

- [127] J.-E. Shin, A. J. Riesselman, A. W. Kollasch, C. McMahon, E. Simon, C. Sander, A. Manglik, A. C. Kruse, and D. S. Marks, “Protein design and variant prediction using autoregressive generative models,” *Nature communications*, vol. 12, no. 1, p. 2403, 2021.
- [128] H. Liu, Z. Chen, Y. Yuan, X. Mei, X. Liu, D. Mandic, W. Wang, and M. D. Plumbley, “Audioldm: Text-to-audio generation with latent diffusion models,” *arXiv preprint arXiv:2301.12503*, 2023.
- [129] A. van den Oord, O. Vinyals, *et al.*, “Neural discrete representation learning,” *Advances in neural information processing systems*, vol. 30, 2017.
- [130] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, “Score-based generative modeling through stochastic differential equations,” in *International Conference on Learning Representations*, 2020.
- [131] J. Copet, F. Kreuk, I. Gat, T. Remez, D. Kant, G. Synnaeve, Y. Adi, and A. Défossez, “Simple and controllable music generation,” *arXiv preprint arXiv:2306.05284*, 2023.
- [132] H. F. Garcia, P. Seetharaman, R. Kumar, and B. Pardo, “Vampnet: Music generation via masked acoustic token modeling,” *arXiv preprint arXiv:2307.04686*, 2023.
- [133] F. Schneider, Z. Jin, and B. Schölkopf, “Moûsai: Text-to-music generation with long-context latent diffusion,” *arXiv preprint arXiv:2301.11757*, 2023.
- [134] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.
- [135] I. Manco, E. Benetos, E. Quenton, and G. Fazekas, “Learning music audio representations via weak language supervision,” in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 456–460, IEEE, 2022.
- [136] B. Elizalde, S. Deshmukh, M. Al Ismail, and H. Wang, “Clap learning audio concepts from natural language supervision,” in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, IEEE, 2023.
- [137] Y. Wu, K. Chen, T. Zhang, Y. Hui, T. Berg-Kirkpatrick, and S. 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)*, pp. 1–5, IEEE, 2023.
- [138] F. Schneider, Z. Jin, and B. Schölkopf, “Moûsai: Text-to-music generation with long-context latent diffusion,” *arXiv preprint arXiv:2301.11757*, 2023.

- [139] A. Défossez, N. Usunier, L. Bottou, and F. Bach, “Music source separation in the waveform domain,” *arXiv preprint arXiv:1911.13254*, 2019.
- [140] E. Manilow, C. Hawthorne, C.-Z. A. Huang, B. Pardo, and J. Engel, “Improving source separation by explicitly modeling dependencies between sources,” in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 291–295, IEEE, 2022.
- [141] C. Donahue, J. McAuley, and M. Puckette, “Adversarial audio synthesis,” in *International Conference on Learning Representations*, 2019.
- [142] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “WaveNet: A Generative Model for Raw Audio,” in *Proc. 9th ISCA Workshop on Speech Synthesis Workshop (SSW 9)*, p. 125, 2016.
- [143] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, “Soundstream: An end-to-end neural audio codec,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 495–507, 2021.
- [144] Z. Borsos, R. Marinier, D. Vincent, E. Kharitonov, O. Pietquin, M. Sharifi, D. Roblek, O. Teboul, D. Grangier, M. Tagliasacchi, *et al.*, “Audiolm: a language modeling approach to audio generation,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.
- [145] F. Kreuk, G. Synnaeve, A. Polyak, U. Singer, A. Défossez, J. Copet, D. Parikh, Y. Taigman, and Y. Adi, “Audiogen: Textually guided audio generation,” in *The Eleventh International Conference on Learning Representations*, 2023.
- [146] Q. Huang, A. Jansen, J. Lee, R. Ganti, J. Y. Li, and D. P. W. Ellis, “Mulan: A joint embedding of music audio and natural language,” in *International Society for Music Information Retrieval Conference*, 2022.
- [147] C. Donahue, A. Caillon, A. Roberts, E. Manilow, P. Esling, A. Agostinelli, M. Verzettti, I. Simon, O. Pietquin, N. Zeghidour, *et al.*, “Singsong: Generating musical accompaniments from singing,” *arXiv preprint arXiv:2301.12662*, 2023.
- [148] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, “Diffwave: A versatile diffusion model for audio synthesis,” in *International Conference on Learning Representations*, 2021.
- [149] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan, “Wavegrad: Estimating gradients for waveform generation,” in *International Conference on Learning Representations*, 2021.
- [150] Y.-J. Lu, Y. Tsao, and S. Watanabe, “A study on speech enhancement based on diffusion probabilistic model,” in *2021 Asia-Pacific Signal and*

- Information Processing Association Annual Summit and Conference (AP-SIPA ASC)*, pp. 659–666, IEEE, 2021.
- [151] J. Serrà, S. Pascual, J. Pons, R. O. Araz, and D. Scaini, “Universal speech enhancement with score-based diffusion,” *arXiv preprint arXiv:2206.03065*, 2022.
- [152] R. Sawata, N. Murata, Y. Takida, T. Uesaka, T. Shibuya, S. Takahashi, and Y. Mitsufuji, “Diffiner: A Versatile Diffusion-based Generative Refiner for Speech Enhancement,” in *Proc. INTERSPEECH 2023*, pp. 3824–3828, 2023.
- [153] K. Saito, N. Murata, T. Uesaka, C.-H. Lai, Y. Takida, T. Fukui, and Y. 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.
- [154] C.-Y. Yu, S.-L. Yeh, G. Fazekas, and H. 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.
- [155] G. Mittal, J. Engel, C. Hawthorne, and I. Simon, “Symbolic music generation with diffusion models,” in *Proceedings of the 22nd International Society for Music Information Retrieval Conference*, 2021.
- [156] C. Hawthorne, I. Simon, A. Roberts, N. Zeghidour, J. Gardner, E. Manilow, and J. Engel, “Multi-instrument music synthesis with spectrogram diffusion,” in *International Society for Music Information Retrieval Conference*, 2022.
- [157] K. W. Cheuk, R. Sawata, T. Uesaka, N. Murata, N. Takahashi, S. Takahashi, D. Herremans, and Y. 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)*, pp. 1–5, IEEE, 2023.
- [158] S. Rouard and G. 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*, pp. 579–585, 2021.
- [159] D. Yang, J. Yu, H. Wang, W. Wang, C. Weng, Y. Zou, and D. Yu, “Diff-sound: Discrete diffusion model for text-to-sound generation,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.
- [160] S. Pascual, G. Bhattacharya, C. Yeh, J. Pons, and J. 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.

- [161] S. Forsgren and H. Martiros, “Riffusion - Stable diffusion for real-time music generation,” 2022.
- [162] R. Yuan, H. Lin, Y. Wang, Z. Tian, S. Wu, T. Shen, G. Zhang, Y. Wu, C. Liu, Z. Zhou, *et al.*, “Chatmusician: Understanding and generating music intrinsically with llm,” *arXiv preprint arXiv:2402.16153*, 2024.
- [163] Y. Wang, Z. Ju, X. Tan, L. He, Z. Wu, J. Bian, *et al.*, “Audit: Audio editing by following instructions with latent diffusion models,” *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [164] E. Gusó, J. Pons, S. Pascual, and J. 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)*, pp. 306–310, IEEE, 2022.
- [165] F. Lluís, J. Pons, and X. Serra, “End-to-end music source separation: Is it possible in the waveform domain?,” in *INTERSPEECH*, pp. 4619–4623, 2019.
- [166] N. Takahashi, N. Goswami, and Y. Mitsufuji, “Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation,” in *Proc. IWAENC*, pp. 106–110, 2018.
- [167] W. Choi, M. Kim, J. Chung, and S. Jung, “Lasoft: Latent source attentive frequency transformation for conditioned source separation,” in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 171–175, IEEE, 2021.
- [168] Q. Kong, Y. Xu, W. Wang, P. J. B. Jackson, and M. D. Plumbley, “Single-channel signal separation and deconvolution with generative adversarial networks,” in *Proc. IJCAI*, p. 2747–2753, AAAI Press, 2019.
- [169] G. Zhu, J. Darefsky, F. Jiang, A. Selitskiy, and Z. Duan, “Music source separation with generative flow,” *IEEE Signal Processing Letters*, vol. 29, pp. 2288–2292, 2022.
- [170] E. Postolache, J. Pons, S. Pascual, and J. 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.
- [171] I. Kavalerov, S. Wisdom, H. Erdogan, B. Patton, K. Wilson, J. Le Roux, and J. R. Hershey, “Universal sound separation,” in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 175–179, IEEE, 2019.
- [172] S. Wisdom, E. Tzinis, H. Erdogan, R. Weiss, K. Wilson, and J. Hershey, “Unsupervised sound separation using mixture invariant training,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 3846–3857, 2020.

- [173] R. Scheibler, Y. Ji, S.-W. Chung, J. Byun, S. Choe, and M.-S. Choi, “Diffusion-based generative speech source separation,” in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2023.
- [174] S. Lutati, E. Nachmani, and L. Wolf, “Separate and diffuse: Using a pretrained diffusion model for improving source separation,” *arXiv preprint arXiv:2301.10752*, 2023.
- [175] G. Plaja-Roglans, M. Marius, and X. 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.
- [176] P. Dhariwal and A. Nichol, “Diffusion models beat gans on image synthesis,” in *Advances in Neural Information Processing Systems* (M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds.), vol. 34, pp. 8780–8794, Curran Associates, Inc., 2021.
- [177] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, “The MUSDB18 corpus for music separation,” 2017.
- [178] J. L. Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, “Sdr – half-baked or well done?,” in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 626–630, 2019.
- [179] V. Jayaram and J. Thickstun, “Source separation with deep generative priors,” in *Proceedings of the 37th International Conference on Machine Learning*, pp. 4724–4735, 2020.
- [180] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu, “Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 5775–5787, 2022.
- [181] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, “Fréchet Audio Distance: A Reference-Free Metric for Evaluating Music Enhancement Algorithms,” in *Proc. Interspeech 2019*, pp. 2350–2354, 2019.
- [182] J. Le Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, “Sdr-half-baked or well done?,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 626–630, IEEE, 2019.
- [183] J. Pons, X. Liu, S. Pascual, and J. Serrà, “Gass: Generalizing audio source separation with large-scale data,” *arXiv preprint arXiv:2310.00140*, 2023.
- [184] H. Liu, Q. Tian, Y. Yuan, X. Liu, X. Mei, Q. Kong, Y. Wang, W. Wang, Y. Wang, and M. D. Plumbley, “Audioldm 2: Learning holistic audio generation with self-supervised pretraining,” *arXiv preprint arXiv:2308.05734*, 2023.

- [185] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241, Springer, 2015.
- [186] Y. Song and S. Ermon, “Generative modeling by estimating gradients of the data distribution,” in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada* (H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox, and R. Garnett, eds.), pp. 11895–11907, 2019.
- [187] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, “Audio set: An ontology and human-labeled dataset for audio events,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 776–780, 2017.
- [188] X. Mei, C. Meng, H. Liu, Q. Kong, T. Ko, C. Zhao, M. D. Plumbley, Y. Zou, and W. Wang, “Wavcaps: A chatgpt-assisted weakly-labelled audio captioning dataset for audio-language multimodal research,” *arXiv preprint arXiv:2303.17395*, 2023.
- [189] C. D. Kim, B. Kim, H. Lee, and G. Kim, “AudioCaps: Generating captions for audios in the wild,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (J. Burstein, C. Doran, and T. Solorio, eds.), (Minneapolis, Minnesota), pp. 119–132, Association for Computational Linguistics, June 2019.
- [190] H. Chen, W. Xie, A. Vedaldi, and A. Zisserman, “Vggsound: A large-scale audio-visual dataset,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 721–725, IEEE, 2020.
- [191] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, “Fma: A dataset for music analysis,” in *International Society for Music Information Retrieval Conference*, 2016.
- [192] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, “The million song dataset,” in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.
- [193] G. Chen, S. Chai, G. Wang, J. Du, W.-Q. Zhang, C. Weng, D. Su, D. Povey, J. Trmal, J. Zhang, *et al.*, “Gigaspeech: An evolving, multi-domain asr corpus with 10,000 hours of transcribed audio,” *arXiv preprint arXiv:2106.06909*, 2021.
- [194] K. Ito and L. Johnson, “The lj speech dataset.” <https://keithito.com/LJ-Speech-Dataset/>, 2017.

- [195] D. Roblek, K. Kilgour, M. Sharifi, and M. Zuluaga, “Fr\`echet audio distance: A reference-free metric for evaluating music enhancement algorithms,” in *Proc. Interspeech*, pp. 2350–2354, 2019.
- [196] A. Gui, H. Gamper, S. Braun, and D. Emmanouilidou, “Adapting frechet audio distance for generative music evaluation,” in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1331–1335, IEEE, 2024.
- [197] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, “High fidelity neural audio compression,” *Transactions on Machine Learning Research*, 2023.
- [198] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, *et al.*, “Cnn architectures for large-scale audio classification,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 131–135, IEEE, 2017.
- [199] Z. Evans, C. Carr, J. Taylor, S. H. Hawley, and J. Pons, “Fast timing-conditioned latent audio diffusion,” *arXiv preprint arXiv:2402.04825*, 2024.
- [200] S. Chopra, R. Hadsell, and Y. LeCun, “Learning a similarity metric discriminatively, with application to face verification,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, pp. 539–546 vol. 1, 2005.
- [201] A. v. d. Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” *arXiv preprint arXiv:1807.03748*, 2018.
- [202] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.
- [203] X. Favory, K. Drossos, T. Virtanen, and X. Serra, “COALA: Co-aligned autoencoders for learning semantically enriched audio representations,” in *ICML 2020 Workshop on Self-supervision in Audio and Speech*, 2020.
- [204] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *International conference on machine learning*, pp. 1597–1607, PMLR, 2020.
- [205] I. Manco, E. Benetos, E. Quinton, and G. Fazekas, “Contrastive audio-language learning for music,” in *Ismir 2022 Hybrid Conference*, 2022.
- [206] S. Pascual, M. Ravanelli, J. Serrà, A. Bonafonte, and Y. Bengio, “Learning problem-agnostic speech representations from multiple self-supervised tasks,” *Interspeech 2019*, 2019.
- [207] M. Tagliasacchi, B. Gfeller, F. d. C. Quitry, and D. Roblek, “Pre-training audio representations with self-supervision,” *IEEE Signal Processing Letters*, vol. 27, pp. 600–604, 2020.

- [208] H.-H. Wu, C.-C. Kao, Q. Tang, M. Sun, B. McFee, J. P. Bello, and C. Wang, “Multi-task self-supervised pre-training for music classification,” in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 556–560, IEEE, 2021.
- [209] P.-Y. Huang, H. Xu, J. Li, A. Baevski, M. Auli, W. Galuba, F. Metze, and C. Feichtenhofer, “Masked autoencoders that listen,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 28708–28720, 2022.
- [210] A. Jansen, M. Plakal, R. Pandya, D. P. Ellis, S. Hershey, J. Liu, R. C. Moore, and R. A. Saurous, “Unsupervised learning of semantic audio representations,” in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 126–130, IEEE, 2018.
- [211] A. Saeed, D. Grangier, and N. Zeghidour, “Contrastive learning of general-purpose audio representations,” in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3875–3879, IEEE, 2021.
- [212] H. Al-Tahan and Y. Mohsenzadeh, “Clar: Contrastive learning of auditory representations,” in *International Conference on Artificial Intelligence and Statistics*, pp. 2530–2538, PMLR, 2021.
- [213] J. Spijkervet and J. A. Burgoyne, “Contrastive learning of musical representations,” *CoRR*, vol. abs/2103.09410, 2021.
- [214] C. Garoufis, A. Zlatintsi, and P. Maragos, “Multi-source contrastive learning from musical audio,” *arXiv preprint arXiv:2302.07077*, 2023.
- [215] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in *International conference on machine learning*, pp. 6105–6114, PMLR, 2019.
- [216] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.
- [217] S. Sarkar, E. Benetos, and M. Sandler, “EnsembleSet: a new high quality synthesised dataset for chamber ensemble separation,” in *Ismir 2022 Hybrid Conference*, 2022.
- [218] D. Bogdanov, M. Won, P. Tovstogan, A. Porter, and X. Serra, “The mtg-jamendo dataset for automatic music tagging,” in *Machine Learning for Music Discovery Workshop, International Conference on Machine Learning (ICML 2019)*, (Long Beach, CA, United States), 2019.