Score Estimation for Generative Modeling

by

Tejas Kumar Jayashankar

S.B., University of Illinois at Urbana-Champaign, 2019
S.M., Massachusetts Institute of Technology, 2022

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2025

Authored by: Tejas Kumar Jayashankar
Department of Electrical Engineering and Computer Science
May 14, 2025

Certified by: Gregory W. Wornell
Sumitomo Professor of Engineering
Thesis Supervisor

Accepted by: Leslie Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Score Estimation for Generative Modeling

by

Tejas Kumar Jayashankar

Submitted to the Department of Electrical Engineering and Computer Science
on May 14, 2025 in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Abstract

Recent advances in score-based (diffusion) generative models have achieved state-of-the-art sample quality across standard benchmarks. Building on the remarkable property of these models in estimating scores, this thesis presents three core contributions: 1) new objectives to reduce score estimation error, 2) a novel Bayesian-inspired optimization framework for solving inverse problems, and 3) a fast one-step generative modeling framework that is based on a novel amortized score estimation framework.

In the first part of this thesis, we introduce two new score estimation objectives with applications to both implicit and diffusion-based generative models. To improve spectral-based non-parametric estimators, we propose a theoretically optimal parametric framework that learns the score by projecting it onto its top-L principal directions. Additionally, inspired by matrix-valued kernel methods, we present a second approach that lifts the score into the space of outer products, and minimizes the distance between the estimated and true scores in this higher-order space.

In the second part, we shift focus from score estimation to leveraging diffusion models as data-driven priors for solving inverse problems. Centering our development around the problem of source separation, we introduce a novel algorithm inspired by maximum a posteriori estimation. This approach combines multiple levels of Gaussian smoothing with an α-posterior, enabling effective signal separation using only independent priors for the sources. We demonstrate the effectiveness of this method through its application to interference mitigation in digital communication signals. Finally, we outline how this framework can be naturally extended to tackle a broader class of inverse problems.

In the final part, we return to the fundamental challenge of efficient sampling, which is critical for enabling practical data-driven engineering systems. We propose a novel generative modeling framework that enables training a one-step neural sampler from scratch. At the core of this method is a new objective based on multi-divergence minimization, guided by a novel approach for score estimation of mixture distributions. Our framework is simple to implement, stable during training, unifies several existing approaches, and achieves state-of-the-art performance in image generation tasks. Furthermore, we discuss how this framework can be naturally extended to multi-step neural sampling and adapted for fast posterior sampling—an essential component in simulation-based inverse problem solvers.

Thesis supervisor: Gregory W. Wornell
Title: Sumitomo Professor of Engineering

Acknowledgments

I owe my deepest gratitude to my advisor, Professor Gregory Wornell. Throughout my time at MIT, his unwavering support, insightful guidance, and belief in my research vision have been vital to my growth as a researcher. His technical mentorship, combined with a deep investment in my development as a creative thinker, made this journey not only possible but profoundly meaningful. When I first arrived at MIT, the sheer range of possibilities was both exhilarating and, at times, overwhelming. I am especially grateful for the steady encouragement and timely nudges that helped me navigate those early moments of uncertainty. Under his mentorship, I have gained skills that I will carry on for the rest of my life—the ability to situate my ideas within the broader scientific landscape, to bridge the gap between theory and practice, and to embrace the mindset that no idea is inherently bad and that persistence is key. For all of this, and so much more, I am sincerely thankful.

I am also grateful to my thesis committee for their valuable guidance and insightful feedback. Professor Yury Polyanskiy challenged me to approach generative modeling and inverse problems with greater theoretical rigor, and he helped sharpen my analytical thinking. His ability to bridge deep theory with practical insights continues to inspire my approach to research. I also thank Professor Lizhong Zheng for his thoughtful questions and for helping me bring coherence to the diverse topics in this thesis.

The PhD journey is never a solo endeavor, and I am deeply grateful to all my labmates, past and present, in the Signals, Information, and Algorithms (SIA) group. The countless discussions, collaborations, and group outings we have shared have not only enriched my academic experience at MIT but have also made this place feel like home. I owe special thanks to Jongha Ryu, whose mentorship, unwavering support, and constant willingness to engage in technical and thoughtful conversations have been invaluable. More than a mentor, he has been a true friend, and this thesis would not have been possible without him—I sincerely hope our collaboration and friendship continue for years to come. I am also grateful to Gary Lee, Amir Weiss, and Alejandro Lancho for bringing me into the RF source separation project. Their patience, guidance, and encouragement helped me navigate and find excitement in a new and initially unfamiliar domain. I would also like to express my heartfelt thanks to Tricia O'Donnell for her steady support and care for me and my labmates throughout our time at MIT.

I was fortunate to complete several internships during my PhD, each of which contributed meaningfully to the practical aspects of this thesis. I am especially grateful to Qing He for giving me the opportunity to intern at Meta AI on two occasions. Her support allowed me to pursue research in variable bitrate neural speech compression and self-supervised learning for generative modeling. I would also like to thank Fabian Mentzer for hosting me at Google Research, where I expanded my interests in neural compression by designing transformer-based architectures for video compression and training these models at scale. Finally, I am deeply thankful to Nikos Vlassis, whose mentorship at Adobe Research provided the opportunity to explore score estimation and one-step generative modeling, laying the foundation for the final part of this thesis.

This thesis reflects only a part of my academic journey, and it is important to acknowledge where it began. As an undergraduate, I was fortunate to be mentored by Professor Pierre

Moulin and Professor Chandrasekhar Radhakrishnan. Professor Moulin introduced me to the foundations of Electrical Engineering and offered me my first experience in signal processing research. That opportunity helped shape my interests and encouraged me to explore the field more deeply at MIT. Professor Radhakrishnan has continued to be a thoughtful mentor, and our conversations about the history and development of signal processing gave me a better appreciation for the field and its current tools.

During my earlier years at MIT, I became interested in the use of generative models for neural compression. I am especially thankful to Ying-zong Huang for his generous guidance and support. He helped me build a broader understanding of the compression landscape and identify meaningful directions to explore. Our collaboration led to a Master's thesis that I found both formative and fulfilling, and it remains one of the most memorable projects from my time at MIT.

I am deeply grateful to the friends and colleagues I met throughout this journey. My time serving on the board and later as co-President of the EECS Graduate Student Association brought many meaningful connections and lasting memories. I especially appreciated the opportunity to engage with the student body and the graduate office. Janet Fisher's support and kindness were particularly memorable during this period. Outside of academics, the MIT Sports Taekwondo Club provided a supportive and energetic environment where I could grow physically and represent MIT in competitions across New England. The friendships I formed and the positive spirit of the team made even the most challenging training sessions enjoyable.

My friends at MIT and in the greater Boston area have truly been my family away from home. Mark Saad, Johan Pereira, Karthiga Mahalingam and Pavani Majety, my fellow *Beach Buddies*, have stood by me since the day I arrived in Massachusetts. Their steady support, thoughtful humor, and kind encouragement have meant more to me than words can express. Whether it was entertaining my ideas, lifting my spirits, or helping me through difficult times, they were always there. I will always cherish our time together and look forward to the many adventures that lie ahead, perhaps even one that takes us back to a beach someday. Vivienne Zhang brought a sense of joy and spontaneity to my journey. From learning squash and enjoying great meals to spending time with my dog Qiqi, the moments we shared are ones I will always hold close. I am also deeply thankful to the many other friends who made this time so meaningful: Siddharth Muralidaran, Rushik Desai, Aniket Patel, Francis Roxas, Christian Chow, Riyasat Ohib, Zhutian Yang, Shyan Akmal, Laura Brandt, Michail Ouroutzoglou, Safa Medin, Abhin Shah, and Sohil Shah. Their presence helped shape this experience in unforgettable ways.

Finally, I owe my deepest gratitude to my parents, sister, and grandparents, whose unwavering love and belief in me have been the foundation of everything I have achieved. Their sacrifices, encouragement, and quiet strength have carried me through every challenge, even from afar. I am constantly inspired by their resilience and generosity, and it is their support that has made this journey possible. This work is a reflection of their enduring presence in my life, and I carry their love with me in all that I do.

to my parents
for their endless love and support

Score Estimation for Generative Modeling

Contents

List of Figures

List of Tables

1 Introduction

The volume of digital data consumed daily is currently estimated at several million terabytes (2^{60} bytes) and continues to grow at an exponential rate. This data is constantly accessed, stored, and shared, not only between users but also across various interoperating modules within engineering systems. For instance, self-driving cars often rely on visual measurements such as infrared images or high-temporal-resolution videos. These measurements are critical, as the information extracted from them plays a key role in downstream processes that drive corrective maneuvers and ensure safety decisions are made in real-time.

Traditionally, engineering systems have been designed by experts, relying on hand-crafted methods for data modeling and processing. However, with the sheer abundance of data in today's world, we are witnessing a shift towards data-driven engineering systems, e.g., intelligent chatbots (OpenAI, 2022) and next generation 6G wireless systems (Nokia, 2023). This shift allows us to harness the powerful capabilities of machine learning and artificial intelligence, enabling systems that not only simplify and streamline processes but also outperform traditional approaches.

At the core of these systems are generative models — parametric models that capture the rich statistical structures of a signal, often solely from data. Often parametrized via neural network architectures, once trained, these models allow for sampling and making probabilistic queries from rich and high-dimensional data.

1.1 Motivation

We start by motivating the need for generative modeling through real-world problems.

1.1.1 Signal Reconstruction and Modeling

Let $\mathbf{x} \in \mathbb{R}^D$ represent a D-dimensional signal of interest, and consider the corruption process given by

$$\mathbf{y} = \mathsf{A}\mathbf{x} + \mathbf{n}, \quad \mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_D), \tag{1.1}$$

where $\mathsf{A} \in \mathbb{R}^{d \times D}$ is a linear forward operator. This model can describe systems such as image blurring, superresolution, or filtering, where A introduces a (convolutional) distortion.

Given a measurement $\mathbf{y}$, the task is to recover $\mathbf{x}$. In the absence of noise, a naïve approach would be to use the pseudoinverse, $\hat{\mathbf{x}} = \mathsf{A}^\dagger \mathbf{y}$. However, this often leads to poor estimates for several reasons. First, multiple pseudoinverses exist, and not all are suitable for recovering the

specific structures inherent in $\mathbf{x}$. Second, matrix inversion can be unstable and ill-conditioned in high-dimensional settings.

Bayesian approaches offer a more principled and expressive framework for signal recovery by casting the problem as the estimation of the *maximum a posteriori* (MAP) solution. Specifically, given a noisy observation $\mathbf{y}$, the goal is to recover the clean signal $\hat{\mathbf{x}}$ by solving:

$$\begin{aligned}\hat{\mathbf{x}} &= \arg\max_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{y}) \\ &= \arg\max_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x}) + \log p(\mathbf{x}),\end{aligned} \tag{1.2}$$

where $p(\mathbf{x})$ denotes the prior distribution over the clean signals, and $p(\mathbf{y}|\mathbf{x})$ is the likelihood model. This can be specialized for the setting where the likelihood is known to be Gaussian (which is an assumption often made in practice) and the MAP estimation problem can be rewritten as a minimization problem. This gives rise to the following formulation:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - \mathsf{A}\mathbf{x}\|_2^2 - \log p(\mathbf{x}), \tag{1.3}$$

which resembles Eq. (1.4) but replaces the hand-crafted regularizer with a prior-driven term $p(\mathbf{x})$ that captures richer statistical structure in the signals.

In practice, however, the true prior $p(\mathbf{x})$ is rarely known. Traditional approaches approximate it using tractable models such as Gaussian mixture models (GMMs) or Markov random fields (MRFs). More broadly, classical methods typically solve:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - \mathsf{A}\mathbf{x}\|_2^2 + \mathcal{R}(\mathbf{x}), \tag{1.4}$$

where $\mathcal{R}(\mathbf{x})$ is a regularization term that encodes prior knowledge about the signal. In image deblurring, for example, the Tikhonov regularizer $\mathcal{R}(\mathbf{x}) \triangleq \|\mathsf{L}\mathbf{x}\|_2^2$ is widely used, where L is a linear operator estimating image gradients via first-order differences. While such regularizers simplify the optimization and improve robustness, they often lack the capacity to capture the complex structures present in real-world signals. As a result, they can overly constrain the solution and limit recovery performance, especially under severe distortions.

This highlights the need for learning expressive and computationally tractable priors, which can significantly enhance the performance of signal recovery systems. In the chapters that follow, we will explore how generative models can be leveraged to learn such priors, opening the door to more powerful and flexible reconstruction techniques.

1.1.2 Simulated Bayesian Inference

In the earlier setup, we assumed access to a known likelihood model. But what if the likelihood is unknown or intractable? Simulation-Based Inference (SBI) (Cranmer et al., 2020), also referred to as likelihood-free inference, offers a powerful Bayesian framework for solving such inverse problems. In SBI, although the likelihood function is not available, a forward simulator is assumed. This allows for the generation of multiple $(\mathbf{x}, \mathbf{y})$ pairs in parallel, enabling the approximation of the posterior distribution $p(\mathbf{x}|\mathbf{y})$ through simulation.

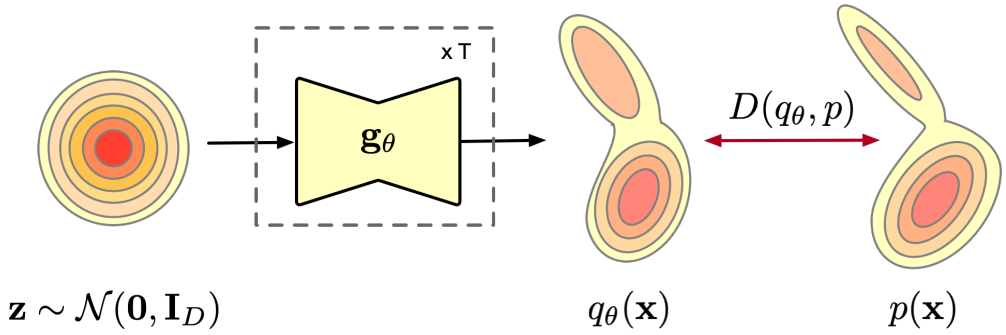


Figure 1.1.: A generative neural sampler maps a sample from a tractable Gaussian distribution (over T model evaluations) to the generated sample distribution $q_\theta(\mathbf{x})$. The latter distribution is aligned with the data distribution typically by minimizing a statistical distance.

By leveraging forward simulations, SBI sidesteps the need for an explicit likelihood model while still supporting principled uncertainty quantification. Key methodologies in this framework include Approximate Bayesian Computation (ABC), neural posterior estimation, and neural likelihood or ratio estimation, each of which aims to recover or approximate the posterior using only simulated data (Gloeckler et al., 2024).

A major strength of SBI is its flexibility in handling complex models where traditional Bayesian methods fail. With recent advances in deep learning, SBI can use neural networks to learn mappings from data to posteriors, making it efficient for repeated inference tasks. This is especially useful in high-dimensional or ill-posed inverse problems, where many solutions may explain the data equally well. By capturing full posterior distributions rather than single point estimates, SBI naturally accounts for uncertainty and ambiguity—making it a valuable tool when reliable and interpretable predictions are needed.

In this thesis, we explore how neural samplers can be trained from scratch to enable rapid inference, and how these methods can be extended to learn efficient posterior samplers tailored for SBI tasks.

1.1.3 Data-driven Signal Priors and Neural Samplers

This previous sections highlight the simplifying assumptions and practical constraints that existing systems often rely on to address key engineering challenges. While hand-crafted priors and regularizers have proven effective in many cases, scaling these approaches to handle novel data modalities—especially in an efficient and scalable manner—remains a significant challenge. At the same time, supervised learning with nonlinear models such as neural networks has shown strong performance in modeling the joint statistics of paired $(\mathbf{x}, \mathbf{y})$ datasets for discriminative tasks. While effective, such methods often require extensive data and the learned model may not be easily leveraged for solving other downstream signal processing tasks

Generative neural samplers offer a powerful solution in this context. At a high level, given samples from a data distribution $p(\mathbf{x})$, a generative neural sampler $\mathbf{g}_\theta$ parametrized by a

neural network with weights θ maps samples from a tractable distribution (e.g., Gaussian) to samples from a distribution $q_\theta(\mathbf{x})$. The sampler is is trained to align q_θ with the data distribution, as illustrated in Figure 1.1. This alignment can be enforced in various ways, with one of the most common approaches being the minimization of a statistical distance between the two distributions, such as through an f-divergence (Csiszár et al., 2004).

Rather than relying on hand-crafted regularizers, we can instead leverage the power of generative models to implicitly impose structural constraints on the solutions. By utilizing samples generated from a trained neural sampler, we can solve for Eq. (1.4) under a suitable differentiable regularizer. This approach enables the model to inherently learn and enforce the underlying structure of the data, allowing for a more flexible and data-driven solution to the inverse problem. Moreover, the generative model can be re-used as a regularizer for solving other inverse problems as well.

However, solving Eq. (1.2) is not feasible with a neural sampler alone. To make this work, we need an estimate of the probability density of the sample, or its gradient if using gradient-based optimization methods. As an example, score-based generative models (Ho et al., 2020; Song et al., 2021b) that estimate the score $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ can be leveraged in the latter setting. Understanding which type of generative model is most suited to a given practical application is crucial, as it provides a versatile tool for solving complex problems in a data-driven manner.

1.1.4 Other Motivating Applications

Generative models have far reaching applications beyond inverse problems. We list some applications below, several of which serve as motivations for the methods introduced in this thesis.

- *Environment Simulation and Synthetic Data Generation:* Generative models can be employed to simulate a wide range of physical phenomena using only sample data, such as non-linearities in digital communication systems or reverberation effects in underwater acoustic environments. To streamline and accelerate the development of new systems in these settings, generative models can also be used to synthetically generate novel training data, facilitating the advancement and proliferation of emerging technologies.

- *Efficient Audio-visual and Text Generation:* Generative models have made remarkable strides in synthesizing novel multimedia content in recent years. Architectures such as transformers (Vaswani, 2017), which power cutting-edge chatbots like ChatGPT (OpenAI, 2022), and diffusion models (Ho et al., 2020), which have set new benchmarks for sample quality across audio, image, and video modalities, are prime examples of these advances. However, these models often come with high inference costs. As multimedia sharing across multiple modalities increasingly shapes the future of communication, developing more efficient algorithms for training generative neural samplers becomes essential. These faster samplers have the potential to impact a wide range of fields, including education, entertainment, and medicine.

- *Medical Diagnosis and Imaging:* Generative models play a crucial role in the medical field. They have already shown great promise in tasks like protein folding, which can significantly accelerate drug discovery (Jumper et al., 2021). Additionally, generative models can be utilized for data augmentation, supporting the training of new doctors and enhancing disease diagnosis and treatment under limited data accessibility constraints.

- *Data Compression and Representation Learning:* Instead of relying on traditional hand-crafted codecs (encoder-quantizer-decoder), generative models can be used as density estimators to constrain the information content in learned representations in a data-driven way. This approach leads to a new class of data compression methods known as non-linear transform coding (Ballé et al., 2020). Generative models also enable universal compression with model-free encoders, paving the way for future-proof data storage techniques (Jayashankar, 2022). Furthermore, generative models can be used for superresolution of the decoded signal, reducing distortion in existing compression systems by more effectively leveraging information from compressed bitstreams in a data-driven manner. This has the potential to enable efficient communication and storage of novel formats of data including 3D and augmented reality (AR) content.

- *Probabilistic Modeling and Uncertainty Quantification:* By modeling the underlying probability distribution of data, generative models can provide valuable insights into the uncertainty inherent in predictions. For instance, they can be used to approximate posterior distributions in Bayesian inference, allowing for more accurate uncertainty quantification in tasks such as risk assessment and decision-making. The ability to generate samples from uncertain or unknown distributions also allows generative models to improve the robustness of simulations, particularly in fields like engineering, finance, and climate modeling, where accurate uncertainty quantification is crucial for reliable decision-making.

1.2 Generative Models

The previous section highlighted the practical advantages of using generative models to address key challenges in engineering. With several generative modeling frameworks introduced in recent years, a natural question arises — which framework is best suited for the task at hand? As we will explore, each generative model comes with its own distinct properties and learning frameworks.

1.2.1 Explicit and Implicit Generative Models

For over a decade generative models typically fell into two classes based on their ability to model the likelihood of data:

- *Explicit Models*: These models are typically trained by either estimating the density of samples or minimizing a variational bound on the likelihood of the data. A well-known example is the variational autoencoder (VAE) (Kingma and Welling, 2014), which employs a stochastic encoder to map inputs to a Gaussian-regularized latent space

Table 1.1.: Overview of popular generative models, their training characteristics, and the inherent trade-offs involved.

Model	Training Idea	Tradeoff
VAE	minimize variational upper bound on NLL	one-step sampling, stable training, good density estimation low-fidelity and blurry samples
Normalizing Flow	minimize NLL via chain rule of probability	excellent density estimation invertibility constraint and poor samples
GAN	minimize JSD with discriminator	high fidelity one-step sampling unstable training, no density estimation
Diffusion Model	multi-noise-level denoising score matching	high-fidelity samples, stable training, density estimation expensive sampling

and a decoder to reconstruct the data. The model is trained by minimizing an upper bound on the negative log likelihood (NLL). Another example is the normalizing flow (Rezende and Mohamed, 2015), which utilizes an invertible neural network to transform a tractable distribution into data samples, estimating the likelihood through the change of variables formula.

- *Implicit Models*: Unlike explicit models, where the neural sampler is a by-product of the density estimation process, implicit models directly learn a neural sampler, which is trained by minimizing a statistical distance, as illustrated in Figure 1.1. A prominent example in this category is the Generative Adversarial Network (GAN) (Goodfellow et al., 2014), which alternately trains a generator and an auxiliary discriminator network. The discriminator's objective is to estimate the density ratio between the generated and true distributions, which is used to compute a statistical distance known as the Jensen-Shannon Divergence (JSD), to align the two distributions and train the generator.

The Generative Tradeoff. At first glance, explicit models may seem appealing due to their ability to model the likelihood and train a one-step neural sampler. However, as with most things in machine learning, there is no free lunch—excelling in one aspect often requires trade-offs in others as summarized in Table 1.1. Normalizing flows, for example, are excellent density estimators but tend to produce blurry samples. Additionally, their training can be computationally expensive due to the constraints imposed by the invertibility requirement. On the other hand, GANs produce high-quality samples and set the state-of-the-art in several domains. However, training GANs is notoriously challenging because of the underlying min-max objective, and we cannot easily compute density estimates, often resorting to Monte Carlo methods instead.

1.2.2 Score-based Generative Models: Diffusion

Recently a new paradigm for generative modeling has emerged that bridges both explicit and implicit modeling techniques. Rather than trading off density estimation for sampling, these

models trade off sampling efficiency, i.e., sampling is achieved via T successive evaluations of the underlying model rather than a single evaluation.

Instead of learning a one-step neural sampler, score-based generative models or as they are known via their more popular moniker, diffusion models, (Ho et al., 2020; Karras et al., 2022; Sohl-Dickstein et al., 2015; Song et al., 2021b) are characterized by their ability to estimate the score of the *noisy* data distribution defined as,

$$\mathbf{s}(\mathbf{x}_t) := \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t),$$

where $\mathbf{x}_t := \mathbf{x} + \sigma_t \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_D)$. Here t denotes a timestep in $[0, 1]$ and it parametrizes a corresponding noise level σ_t. As mentioned in Table 1.1, by sacrificing sampling efficiency, score-based models produce high-quality samples with stable training dynamics. This is achieved, by defining a forward process that slowly corrupts the signal with increasing amounts of Gaussian noise such that $\mathbf{x}_1 \sim \mathcal{N}(\mathbf{0}, \sigma_1^2 \mathbf{I}_D)$. The generative process reverses this process by using the score of intermediate distributions to synthesize samples from Gaussian noise in T steps. Both of these processes can be expressed as stochastic differential equations (SDEs) and hence its theoretical properties can be analyzed in practice. Additionally, an ODE based on the Fokker-Planck equation governs the distributional evolution (Song et al., 2021b) and allows for likelihood computation.

As alluded to in Section 1.1.1, with such a learned score it should now be more clear how a gradient-based algorithm can be practically implemented to solve Eq. (1.2) by utilizing the score of the prior distribution. This connection will be further explored and clarified throughout this thesis.

1.2.3 Efficient and Stable One-Step Generation: A New Paradigm

Modern learning-based solutions are often data-hungry. For instance, transformer architectures (Vaswani, 2017) are well-known to significantly underperform in data-scarce scenarios, and diffusion models frequently benefit from data augmentation to "virtually" expand the size of their training datasets (Karras et al., 2022). But what happens when we encounter a new problem with limited data? How can we scale and adapt existing generative modeling techniques to suit these constraints?

One common solution is synthetic data generation. Since generative models can capture the underlying properties of a data distribution, it makes sense to leverage their capabilities to generate new data. However, achieving this at scale requires an efficient and easy-to-train neural sampler. GANs provide efficiency but are notoriously difficult to train, while diffusion models are stable but challenging to sample from.

Now, imagine an alternative to GANs that offers fast, one-step generation without the training instabilities. This model would generate data in a single step, but to achieve this, we would need to replace the unstable min-max discriminator learning objective. Given that the diffusion training objective is stable and practical, a novel score estimation approach could be introduced as an auxiliary task—rather than the primary focus—to help define a new, stable one-step generative modeling paradigm. Such an approach would enhance the capabilities of score estimation in training high-quality generative models. This thesis culminates in the development of such a framework.

1.3 Thesis Guide

1.3.1 Overview

This thesis is focused on questions of effectiveness, utility, quality and efficiency surrounding score estimation for generative modeling.

In the first part, we begin from first principles to explore score estimation, drawing on existing literature to develop novel frameworks. We specifically examine the approximation errors inherent to non-parametric methods and introduce new parametric score estimation techniques from the optimal scalar and matrix kernel regression point of view. The quality of the estimator is assessed by training generative models with the learned score models. A preliminary version of this part of the thesis can be found in (Jayashankar et al., 2024) and a full version is under review.

In the second part of this thesis, we shift our focus to evaluating the utility of score estimators learned by diffusion models in addressing practical signal processing challenges. While much of the existing research in diffusion modeling emphasizes downstream sample quality typically assessed using metrics like the Fréchet Inception Distance (FID) (Heusel et al., 2017)—we take a different approach. Specifically, we explore the signal source separation problem, utilizing score estimators as statistical priors for the signals. This enables us to assess the quality of the score estimation by its ability to accurately model complex statistical structures inherent in the signals via a novel Bayesian algorithm. A published version of this part of the thesis can be found in (Jayashankar et al., 2023).

In real-world engineering applications, there exists a critical trade-off between quality and efficiency. Sampling from existing diffusion models can be prohibitively expensive, with costs rising steeply as the dimensionality of the data increases. To address this issue, the last part of this thesis explores methods to enhance the efficiency and stability of generative samplers. Specifically, we propose novel score estimation strategies for training one-step generative models based purely on statistical divergence minimization, building on the technical tools developed earlier in the thesis. A published version of this part of the thesis can be found at (Jayashankar et al., 2025).

1.3.2 Organization

This thesis starts by an introduction of necessary tools, notiation and prerequisites surrounding score estimation and diffusion models in Chapter 2 (Score-based Generative Modeling Preliminaries). The thesis is then sectioned in three parts.

The main technical development for improving score estimation by addressing the suboptimality of non-parametric estimators is in Chapter 3 (Principal Direction Score Estimation). Our new parametric score estimator inspired by matrix-kernel-based non-parametric score estimation is introduced in Chapter 4 (Lifted Residual Score Estimation).

Chapter 5 (Score-based Source Separation) introduces a novel Bayesian method for source separation using independent score models and Chapter 6 (RF Source Separation) applies the proposed method for source separation of digital communication signals.

We turn to the problem of efficient and fast generative sampling in the last part of this thesis.

Chapter 7 (Overview of One-Step Generative Modeling Techniques) provides background of several recent one-step generative modeling frameworks along with their shortcomings and tradeoffs. Chapter 8 (Stable and Efficient Generative Modeling with Score-of-Mixture Training) introduces our new proposed method for one-step generative model training based on multi-divergence minimization and score of mixture estimation.

Each chapter concludes with appendices that provide supplementary material, including additional results, deferred proofs, and background information to support a deeper understanding of the main content. We also include special sections marked with an asterisk (*), which present preliminary analyses and early results for potential extensions and future research directions.

We finally summarize our contributions and discuss potential new research directions in Chapter 9 (Concluding Remarks and Future Directions).

2 Score-based Generative Modeling Preliminaries

The goal of this thesis is twofold — to advance the capabilities of generative models through score estimation, and to explore novel applications of parametric score estimators in practical engineering contexts. To achieve this, we draw inspiration from both theoretical foundations and real-world applications. This chapter serves as an introduction to the key concepts and techniques that underpin this thesis, namely score matching and diffusion-based generative modeling.

2.1 Notation

We write e.g., x for a scalar-valued variable and e.g., $\mathbf{x}$ for a vector-valued variable with x_i denoting the i'th element in the vector. When necessary we will use e.g., x to denote a random variable.

We use e.g., B to denote a matrix and $B_{i,j}$ to denote the element in row i and column j.

We use e.g., $\mathsf{J}_{\mathbf{x}} f(\mathbf{x}, \mathbf{y})$ to denote the Jacobian of the function with respect to $\mathbf{x}$.

We use e.g., $\mathcal{S}$ to denote a set of objects with cardinality $|\mathcal{S}|$.

We use e.g., $\mathsf{routine}$ to denote a standard computational routine or function used in practice.

For a continuous valued random variable x we denote its probability density function by $p(x)$ or more explicitly as $p_{\mathsf{x}}(x)$ when needed. Likewise for a discrete valued random variable we denote its probability mass function by $P(x)$ or more explicitly as $P_{\mathsf{x}}(x)$ when needed.

Analogously, for a continuous valued random vector x we denote its probability density function by $p(\mathbf{x})$ or more explicitly as $p_{\mathsf{x}}(\mathbf{x})$ when needed. Likewise for a random vector drawn from a finite set $\mathcal{X}$ we denote its probability mass function by $P(\mathbf{x})$ or more explicitly as $P_{\mathsf{x}}(\mathbf{x})$ when needed.

2.2 Score Estimation

Assume that there exists an underlying distribution with a continuously differentiable density $p(\mathbf{x})$ over $\mathcal{X} \subset \mathbb{R}^D$, where the score function is defined as

$$\mathbf{s}(\mathbf{x}) := \nabla_{\mathbf{x}} \log p(\mathbf{x}) \in \mathbb{R}^D.$$

The goal of score estimation is to construct an estimator $\hat{\mathbf{s}}(\mathbf{x})$ from data $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ drawn from $p(\mathbf{x})$. In this thesis, we particularly assume a class of parametric score models

$\{\mathbf{s}_\theta(\mathbf{x}) : \theta \in \Theta\}$ such as neural networks, and aim to find the best hypothesis from the class as a proxy to the underlying score function $\mathbf{s}(\mathbf{x})$. In this section, we provide a brief overview on the literature of both parametric and nonparametric score estimation.

2.2.1 Parametric Score Estimation

2.2.1.1 Exact Score Matching

Hyvärinen (2005) initially proposed the score matching (SM) objective for training unnormalized density models which can expressed simply as,

$$\mathcal{L}_{\mathrm{SM}}(\mathbf{s}; \mathbf{s}_\theta) := \mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|^2]. \tag{2.1}$$

This objective is not realizable in practice as it depends on the ground truth score $\mathbf{s}(\mathbf{x})$ which is typically unavailable in practice. Hyvärinen (2005) showed that an equivalent sample-only objective exists, which we call exact SM to distinguish it from the variants that follow,

$$\begin{aligned}
\mathcal{L}_{\mathrm{ESM}}(\mathbf{s}; \mathbf{s}_\theta) &:= \frac{1}{2}(\mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|^2] - \mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}(\mathbf{x})\|^2]) \\
&= -\mathbb{E}_{p(\mathbf{x})}[\mathbf{s}(\mathbf{x})^\intercal \mathbf{s}_\theta(\mathbf{x})] + \frac{1}{2}\mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}_\theta(\mathbf{x})\|^2] && (2.2) \\
&\overset{(a)}{=} \mathbb{E}_{p(\mathbf{x})}[\mathrm{tr}(\mathsf{J}_\mathbf{x}\mathbf{s}_\theta(\mathbf{x}))] + \frac{1}{2}\mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}_\theta(\mathbf{x})\|^2]. && (2.3)
\end{aligned}$$

Here, (a) follows by integration by parts, under some mild regularity conditions on the density $p(\mathbf{x})$ and score function $\mathbf{s}(\mathbf{x})$. While the SM framework provides a computable objective function, the computation often becomes demanding due to the trace of the Jacobian $\mathrm{tr}(\mathsf{J}_\mathbf{x}\mathbf{s}_\theta(\mathbf{x}))$. While the latter is easy to compute at low dimensions using automatic differentiation tools, it can be very expensive to compute for high-dimensional data.

2.2.1.2 Sliced Score Matching

As a solution to the computational burden of ESM, Song et al. (2020) proposed an equivalent objective

$$\mathcal{L}_{\mathrm{SSM}}(\mathbf{s}; \mathbf{s}_\theta) := \mathbb{E}_{p(\mathbf{x})p(\mathbf{v})}[\mathbf{v}^\intercal \mathsf{J}_\mathbf{x}\mathbf{s}_\theta(\mathbf{x})\mathbf{v}] + \frac{1}{2}\mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}_\theta(\mathbf{x})\|^2]. \tag{2.4}$$

Here, Hutchinson (1989)'s trick is applied to the first term in Eq. (2.3),

$$\begin{aligned}
\mathrm{tr}(\mathsf{J}_\mathbf{x}\mathbf{s}_\theta(\mathbf{x})) &= \mathrm{tr}(\mathsf{J}_\mathbf{x}\mathbf{s}_\theta(\mathbf{x})\mathbb{E}_{p(\mathbf{v})}[\mathbf{v}\mathbf{v}^\intercal]) \\
&= \mathbb{E}_{p(\mathbf{v})}[\mathbf{v}^\intercal \mathsf{J}_\mathbf{x}\mathbf{s}_\theta(\mathbf{x})\mathbf{v}], && (2.5)
\end{aligned}$$

where $p(\mathbf{v})$ is a noise distribution over $\mathbb{R}^D$ such that $\mathbb{E}_{p(\mathbf{v})}[\mathbf{v}\mathbf{v}^\intercal] = \mathsf{I}_D$. In practice, the noise distribution is often chosen as a standard normal distribution or a Rademacher distribution. They call this computational trick slicing and call the objective the sliced SM (SSM) objective.

It is important to note that SSM also requires the computation of the Jacobian of the estimator. The primary computational advantage of this approach lies in replacing the trace

computation with a Jacobian-vector product, which can be efficiently implemented using slicing vectors in modern deep learning frameworks. However, in practice, the estimate in Eq. (2.4) may become biased when only a limited number of slicing vectors are used. As a result, SSM tends to perform worse in high-dimensional problem settings.

2.2.1.3 Denoising Score Matching

To avoid the computational complexity of derivatives altogether, Vincent (2011) proposed to estimate the score of a noisy version of the underlying distribution. The resulting objective known as denoising score matching (DSM) lies at the heart of applied statistics and state-of-the-art generative models,

Formally, denote a noisy version of $\mathbf{x}$ as $\mathbf{x}_\sigma = \mathbf{x} + \sigma\boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_D)$, where σ is some noise level. The induced Gaussian conditional distribution is $p(\mathbf{x}_\sigma|\mathbf{x}) = \mathcal{N}(\mathbf{x}_\sigma; \mathbf{x}; \sigma^2\mathbf{I}_D)$ and the induced marginal distribution is $p(\mathbf{x}_\sigma) = \mathbb{E}_{p(\mathbf{x})}[p(\mathbf{x}_\sigma|\mathbf{x})]$. We can then define $\mathbf{s}(\mathbf{x}_\sigma) := \nabla_{\mathbf{x}_\sigma} \log p(\mathbf{x}_\sigma)$ as the score of $p(\mathbf{x}_\sigma)$ and $\mathbf{s}(\mathbf{x}_\sigma|\mathbf{x}) := \nabla_{\mathbf{x}_\sigma} \log p(\mathbf{x}_\sigma|\mathbf{x})$ as the conditional score function. Then, applying Eq. (2.2) on a parametric score model $\mathbf{x}_\sigma \mapsto \mathbf{s}_\theta(\mathbf{x}_\sigma)$ and the noisy score $\mathbf{s}(\mathbf{x}_\sigma)$, the DSM objective is derived as

$$\begin{aligned}\mathcal{L}_{\text{DSM}}(\mathbf{s}; \mathbf{s}_\theta) &:= \frac{1}{2}(\mathbb{E}_{p(\mathbf{x}_\sigma)}[\|\mathbf{s}(\mathbf{x}_\sigma) - \mathbf{s}_\theta(\mathbf{x}_\sigma)\|^2] - \mathbb{E}_{p(\mathbf{x}_\sigma)}[\|\mathbf{s}(\mathbf{x}_\sigma)\|^2]) \\ &= \mathbb{E}_{p(\mathbf{x}_\sigma)}\left[-\mathbf{s}(\mathbf{x}_\sigma)^\intercal \mathbf{s}_\theta(\mathbf{x}_\sigma) + \frac{1}{2}\|\mathbf{s}_\theta(\mathbf{x}_\sigma)\|^2\right] \\ &\overset{(b)}{=} \mathbb{E}_{p(\mathbf{x})p(\mathbf{x}_\sigma|\mathbf{x})}\left[-\mathbf{s}(\mathbf{x}_\sigma|\mathbf{x})^\intercal \mathbf{s}_\theta(\mathbf{x}_\sigma) + \frac{1}{2}\|\mathbf{s}_\theta(\mathbf{x}_\sigma)\|^2\right],\end{aligned} \tag{2.6}$$

where the (generalized) Tweedie's formula, introduced in more detail in Section 2.2.2 , $\mathbb{E}_{p(\mathbf{x}|\mathbf{x}_\sigma)}[\mathbf{s}(\mathbf{x}_\sigma|\mathbf{x})] = \mathbf{s}(\mathbf{x}_\sigma)$ is used in (b). When $\mathbf{s}(\mathbf{x}_\sigma|\mathbf{x})$ is easy to compute for a given choice of $p(\mathbf{x}_\sigma|\mathbf{x})$ such as a Gaussian, the final objective can be computed without derivatives of the estimator. Particularly, the DSM objective takes the form,

$$\begin{aligned}\mathcal{L}_{\text{DSM}}(\mathbf{s}; \mathbf{s}_\theta) &\overset{(a)}{=} \mathbb{E}_{p(\mathbf{x})p(\mathbf{x}_\sigma|\mathbf{x})}\left[-\left(\frac{\mathbf{x}_\sigma - \mathbf{x}}{\sigma^2}\right)^\intercal \mathbf{s}_\theta(\mathbf{x}_\sigma) + \frac{1}{2}\|\mathbf{s}_\theta(\mathbf{x}_\sigma)\|^2\right] \\ &\overset{(b)}{=} \mathbb{E}_{p(\mathbf{x})q(\boldsymbol{\epsilon})}\left[-\left(\frac{\boldsymbol{\epsilon}}{\sigma}\right)^\intercal \mathbf{s}_\theta(\mathbf{x}_\sigma) + \frac{1}{2}\|\mathbf{s}_\theta(\mathbf{x}_\sigma)\|^2\right] \\ &\overset{(c)}{=} \frac{1}{2}\mathbb{E}_{p(\mathbf{x})q(\boldsymbol{\epsilon})}\left[\left\|\mathbf{s}_\theta(\mathbf{x}_\sigma) + \frac{\boldsymbol{\epsilon}}{\sigma}\right\|^2\right] + C,\end{aligned} \tag{2.7}$$

where the conditional score of the Gaussian likelihood is expanded in (a), the expression is rewritten in terms of the additive noise in (b) and a closed form expression where $C := \mathbb{E}_{q(\boldsymbol{\epsilon})}[\|\boldsymbol{\epsilon}/\sigma\|^2]$ is independent of θ is derived in (c). Thus, DSM allows for computing the noisy score in a computationally efficient manner via a closed form regression objective which is easy to implement and stable in practice. It should be noted that while DSM was originally proposed as an approximate solution for direct score estimation, i.e., estimating $\mathbf{s}(\mathbf{x})$, it is not widely used for the same as the denoising parameter is hard to tune. Instead, the DSM has played a key role in the recent development of diffusion models.

2.2.2 Tweedie's Formula

Tweedie's formula (Robbins, 1956) is a remarkable result that admits a computational tool to estimate the marginal score of a noisy distribution, utilizing only the conditional score and samples from the posterior. We first state it below as a theorem and then discuss its practical significance.

Theorem 2.1 (Generalized Tweedie's formula)**.** *For* $(\mathbf{x}, \mathbf{x}_\sigma) \sim p(\mathbf{x})p(\mathbf{x}_\sigma|\mathbf{x})$*,*

$$\mathbb{E}_{p(\mathbf{x}|\mathbf{x}_\sigma)}[\mathbf{s}(\mathbf{x}_\sigma|\mathbf{x})] = \mathbf{s}(\mathbf{x}_\sigma),$$
$$\mathbb{E}_{p(\mathbf{x}|\mathbf{x}_\sigma)}[\mathbf{s}(\mathbf{x}_\sigma|\mathbf{x})\mathbf{s}(\mathbf{x}_\sigma|\mathbf{x})^\intercal + \mathbf{s}^{(2)}(\mathbf{x}_\sigma|\mathbf{x})] = \mathbf{s}(\mathbf{x}_\sigma)\mathbf{s}(\mathbf{x}_\sigma)^\intercal + \mathbf{s}^{(2)}(\mathbf{x}_\sigma).$$

Here, $\mathbf{s}^{(2)}(\mathbf{x}_\sigma) := \nabla^2_{\mathbf{x}_\sigma} \log p(\mathbf{x}_\sigma)$ *and* $\mathbf{s}^{(2)}(\mathbf{x}_\sigma|\mathbf{x}) := \nabla^2_{\mathbf{x}_\sigma} \log p(\mathbf{x}_\sigma|\mathbf{x})$*.*

The first relationship is of great practical significance. Considering a Gaussian noise corrupted signal with noise standard deviation σ, Tweedie's formula gives the following relationship,

$$\begin{aligned}\mathbf{s}(\mathbf{x}_\sigma) &= -\mathbb{E}_{p(\mathbf{x}|\mathbf{x}_\sigma)}\left[\frac{\mathbf{x}_\sigma - \mathbf{x}}{\sigma^2}\right] \\ &= -\frac{1}{\sigma^2}\mathbf{x}_\sigma + \frac{1}{\sigma^2}\mathbb{E}[\mathbf{x}|\mathbf{x}_\sigma].\end{aligned}$$

In practice $\mathbb{E}[\mathbf{x}|\mathbf{x}_\sigma]$ is the optimal Gaussian noise denoiser. Thus, the marginal score estimator can be parametrized by a denoiser $\mathbf{f}_\theta(\mathbf{x}_\sigma) \approx \mathbb{E}[\mathbf{x}|\mathbf{x}_\sigma]$, leading to the following expression for the score:

$$\mathbf{s}_\theta(\mathbf{x}_\sigma) = -\frac{1}{\sigma^2}\mathbf{x}_\sigma + \frac{1}{\sigma^2}\mathbf{f}_\theta(\mathbf{x}_\sigma). \tag{2.8}$$

Denoising is a fundamental task in signal processing, and significant research has been devoted to developing powerful parametric models for this purpose (Milanfar and Delbracio, 2024). These models can, in turn, be leveraged for estimating the marginal score.

2.2.3 Nonparametric Score Estimation

In this section, we provide a brief overview of non-parametric score estimation methods. Unlike parametric approaches, non-parametric methods often estimate the score by employing kernel regression, linear estimation, and regularized objectives that yield closed-form expressions. Although these methods are generally more limited in terms of expressiveness, they play a crucial role in the theoretical development of score estimation frameworks.

2.2.3.1 Stein Gradient Estimator

Li and Turner (2018) proposed estimating $\mathbf{s}(\mathbf{x})$ using the generalized Stein's identity (Gorham and Mackey, 2015; Stein, 1981)

$$\mathbb{E}_{p(\mathbf{x})}[\mathbf{h}(\mathbf{x})\mathbf{s}(\mathbf{x})^\intercal + \mathsf{J}_\mathbf{x}\mathbf{h}(\mathbf{x})] = 0$$

for a choice of test function $\mathbf{h}\colon \mathcal{X} \to \mathbb{R}^D$ with certain regularity conditions. Note that the identity is essentially equivalent to integration by parts used in SM. Li and Turner (2018) considered an empirical version of the identity and proposed to solve the resulting linear system with a regularizer. Given a set of N samples $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\}$, they construct the matrix $\mathsf{H} := [\mathbf{h}(\mathbf{x}^{(1)})\, \mathbf{h}(\mathbf{x}^{(2)}) \ldots \mathbf{h}(\mathbf{x}^{(N)})] \in \mathbb{R}^{D \times N}$. Then the matrix of scores at the sample points S must (approximately) satisfy,

$$-\frac{1}{N}\mathsf{H}\mathsf{S}^\intercal \approx \nabla_{\mathbf{x}}\mathsf{H},$$

where $\nabla_{\mathbf{x}}\mathsf{H} := \sum_{i=1}^{N} \mathsf{J}_{\mathbf{x}}\mathbf{h}(\mathbf{x}^{(i)})/N$. In practice this is implemented by solving a regularized problem,

$$\min_{\mathsf{S}} \left\| \nabla_{\mathbf{x}}\mathsf{H} + \frac{1}{N}\mathsf{H}\mathsf{S}^\intercal \right\|_F^2 + \eta \|\mathsf{S}\|_F^2,$$

for some positive scalar η. They call the resulting estimator the Stein gradient estimator (SGE). SGE has several drawbacks. First, it can only estimate the score function at the given data points, without principled way for extrapolation. Second, the choice of $\mathbf{h}(\mathbf{x})$ governs the quality of approximation, as the method can be understood as learning with kernel $k(\mathbf{x}, \mathbf{x}') = \mathbf{h}(\mathbf{x})^\intercal \mathbf{h}(\mathbf{x}')$; however, choosing a good test function is highly nontrivial.

2.2.3.2 Spectral Stein Gradient Estimator

To resolve the latter downsides, Shi et al. (2018) proposed the spectral Stein gradient estimator (SSGE), where the idea is to use the stack of the top-L eigenfuctions of a kernel as the test functions in the SGE framework. More precisely, consider the top-L orthonormal eigenbasis $\{\phi_i(\mathbf{x})\}_{i=1}^{\infty}$ of a fixed choice of kernel $k(\mathbf{x}, \mathbf{x}')$ with respect to the given distribution $p(\mathbf{x})$. It then aims to estimate the score function $\mathbf{s}(\mathbf{x}) := \nabla_{\mathbf{x}} \log p(\mathbf{x})$ of a given distribution $p(\mathbf{x})$, by considering the order-L expansion

$$\mathbf{s}^{(L)}(\mathbf{x}) := \sum_{i=1}^{L} \mathbb{E}_{p(\mathbf{x})}[\phi_i(\mathbf{x})\mathbf{s}(\mathbf{x})]\phi_i(\mathbf{x}) = \sum_{i=1}^{L} \mathbf{b}_i \phi_i(\mathbf{x}),$$

where $\mathbf{b}_i := \mathbb{E}_{p(\mathbf{x})}[\mathbf{s}(\mathbf{x})\phi_i(\mathbf{x})]$. Compared to SGE, SSGE has a principled formula for extrapolation based on the Nyström method and the error was theoretically analyzed.

2.2.3.3 Nonparametric Score Estimators

Zhou et al. (2020) proposed a unifying framework for nonparametric score estimation methods based on a regularized vector-regression formulation (Baldassarre et al., 2012). For a matrix-valued kernel Γ, let $\mathcal{H}_\Gamma$ denote the reproducing kernel Hilbert space. Then, they defined the score estimator as

$$\hat{\mathbf{s}}_\lambda := \arg\min_{\mathbf{s} \in \mathcal{H}_\Gamma} \mathbb{E}_{\hat{p}(\mathbf{x})}[\|\mathbf{s}(\mathbf{x}) - \hat{\mathbf{s}}(\mathbf{x})\|_2^2] + \frac{\lambda}{2}\|\mathbf{s}\|_{\mathcal{H}_\Gamma}^2,$$

where $\hat{p}(\mathbf{x})$ is the empirical distribution constructed from samples. Based on its closed-form solution (or its variant with other spectral regularization), Zhou et al. (2020) applied a more general version of Stein's identity (essentially integration by parts), and introduced a general

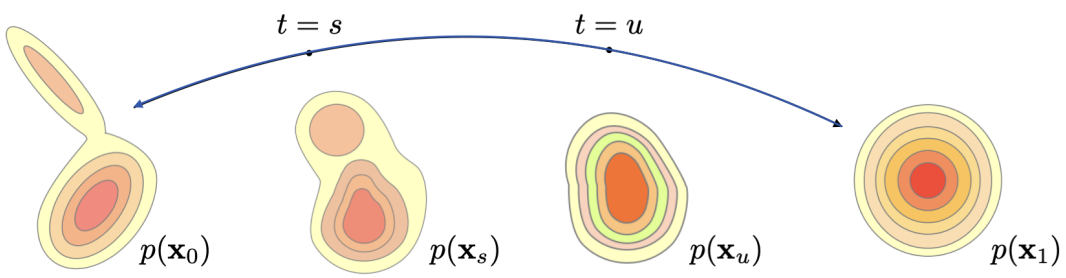


Figure 2.1.: A diffusion model is parametrized by a noise-destructive process running from left to right and a generative process running in the opposite direction. As time evolves the data distribution is gradually corrupted with increasing levels of Gaussian noise.

nonparametric estimator that can be computed based on linear system of size $MN \times MN$, for a matrix-valued kernel of size $M \times M$ and data size N. They showed that this estimator subsumes both SGE and SSGE as special cases, when the underlying matrix-valued kernel is essentially scalar-valued. They also demonstrated that the nonparametric score estimator using truly matrix-valued kernels such as curl-free kernels can substantially improve the performance of SSGE.

2.3 Diffusion Generative Models

Score-based generative models were briefly introduced in Section 1.2.2 as models that admit high-quality sampling and density estimation by trading off sampling efficiency. In this section we provide a technical overview of the most popular instance within this class — diffusion models.

2.3.1 Technical Formulation

We take the following unified view in our definition of DPMs as inspired by (Karras et al., 2022). Let $p(\mathbf{x})$ be the data distribution and let $\lambda(t)$ define a *variance exploding* noise schedule with distribution $p(t)$ where $t \sim \mathcal{U}(0, 1)$. Under this noise schedule we can define a noisy version of $\mathbf{x}$ at noise level σ_t as

$$\mathbf{x}_t := \mathbf{x} + \sigma_t \boldsymbol{\epsilon} \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}). \tag{2.9}$$

Rather than sampling from the data distribution via a one-step map from a tractable noise distribution as in other implicit models (see Section 1.2.1), as shown in Figure 2.1, diffusion models first define a user-design forward process governed by a stochastic differential equation that gradually destroys structure in the signal,

$$\mathrm{d}\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w},$$

where $\mathbf{w}$ is a standard Wiener process and $\mathbf{x}_0 = \mathbf{x}$. The term $\mathbf{f}(\mathbf{x}_t, t)$ is known as the drift coefficient and $g(t)$ is known as the diffusion coefficient. For the variance exploding processes considered in this thesis, $\mathbf{f}(\mathbf{x}_t, t) = \mathbf{0}$ and $g(t) = \sqrt{2\dot{\sigma}_t \sigma_t}$.

The time reversal of this process (i.e., the generative process) is known to follow the reverse SDE,

$$\mathrm{d}\mathbf{x}_t = \left[\mathbf{f}(\mathbf{x}_t, t) - g^2(t)\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)\right] \mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}.$$

Notice that the sampling requires knowledge of the score. In practice $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$ would be estimated by the score function $\mathbf{s}_\theta(\mathbf{x}_t; t)$ from a variant of DSM.

Sampling can be simulated through techniques such as annealed Langevin dynamics or ancestral sampling (Song et al., 2021b). While the above reverse SDE is stochastic in nature, there also exists a deterministic process known as the *probability flow ODE* that satisfies the same intermediate marginal distributions,

$$\mathrm{d}\mathbf{x}_t = \left[\mathbf{f}(\mathbf{x}_t, t) - \frac{1}{2}g^2(t)\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)\right] \mathrm{d}t. \tag{2.10}$$

The benefit of the ODE formulation is that it can discretized more coarsely and hence sampling can done in fewer timesteps. Furthermore, sampling is possible by plugging in the updates from Eq. (2.10) into black-box ODE solvers, e.g., the Heun 2$^{\text{nd}}$ order solver (Karras et al., 2022). Sampling can be sped even further if Eq. (2.10) can be solved exactly. Lu et al. (2022) show that the exact solution to Eq. (2.10) at timestep t given an initial value at timestep $s < t$ is,

$$\mathbf{x}_t = \mathbf{x}_s + 2\int_{\sigma_s}^{\sigma_t} \sigma_u \boldsymbol{\epsilon}_\theta\left(\mathbf{x}_u; u\right) \mathrm{d}\sigma_u. \tag{2.11}$$

Various samplers can be derived by approximating the exponentially weighted integral in different ways. For example, the widely used DDIM sampler (Song et al., 2021a) is an example of a first-order Taylor expansion of the integral term. At the core of all these algorithms is a score estimator/denoiser, which if learned accurately could improve the quality of samples produced.

2.3.2 Training Objective

Given noisy samples of data, the diffusion objective can be reduced to a weighted denoising objective,

$$\mathcal{L}_{\text{DPM}}(\boldsymbol{\epsilon}_\theta) = \mathbb{E}_{p(t)p(\mathbf{x})q(\boldsymbol{\epsilon})}\left[w(t)\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t; t)\|^2\right], \tag{2.12}$$

where $w(t)$ is a positive scalar-valued weighting function. Note that for the forward process defined in Eq. (2.9), the conditional score is $\mathbf{s}(\mathbf{x}_t|\mathbf{x}) = -\boldsymbol{\epsilon}/\sigma_t$. Thus, Eq. (2.12) can be interpreted as a weighted DSM loss over multiple noise levels,

$$\mathcal{L}_{\text{DPM}}(\boldsymbol{\epsilon}_\theta) = \mathbb{E}_{p(t)p(\mathbf{x})p(\mathbf{x}_t|\mathbf{x})}\left[w'(t)\left\|\mathbf{s}(\mathbf{x}_t|\mathbf{x}) + \frac{\boldsymbol{\epsilon}_\theta(\mathbf{x}_t; t)}{\sigma_t}\right\|^2\right], \tag{2.13}$$

where $w'(t) := \sigma_t^2 w(t)$ and the marginal score estimator is $\mathbf{s}_\theta(\mathbf{x}_t; t) := -\boldsymbol{\epsilon}_\theta(\mathbf{x}_t; t)/\sigma_t$.

Equivalently, some diffusion models train a denoiser to learn the conditional mean via MMSE estimation,

$$\min_\theta \mathbb{E}_{p(t)p(\mathbf{x})q(\boldsymbol{\epsilon})}\left[w(t)\|\mathbf{x} - \mathbf{f}_\theta(\mathbf{x}_t; t)\|^2\right].$$

By applying Tweedie's formula (see Section 2.2.2), the marginal score estimator can be derived using Eq. (2.8). Different models utilize distinct noise schedules and weighting functions, and there is no universally optimal framework. In our experiments, we will clearly outline our design choices.

2.3.3 Prior Work

Sohl-Dickstein et al. (2015) first introduced diffusion probabilistic models (DPMs) as deep variational autoencoders (Kingma and Welling, 2014) based on the principles of thermodynamic diffusion with a Markov-chain variational posterior that maximizes the evidence lower bound (ELBO). Several years later, Ho et al. (2020) re-introduced DPMs (DDPMs) with modern neural network architectures and a simplified loss function that set a new state-of-the-art in image generation. Since then, numerous connections to existing literature in statistics, information theory and stochastic differential equations (SDEs) have helped bolster the quality of these models. For example, Song and Ermon (2019) illustrate the equivalence between DDPMs and DSM at multiple noise levels, thus bridging the areas of diffusion-based models and score-based models. Subsequently, Song et al. (2021b) showed that in continuous time, DPMs can be appropriately interpreted as solving for the reverse of a noising process that evolves as an SDE while Kingma et al. (2021) demonstrated that continuous-time DPMs can interpreted as VAEs and that the variational lower bound is invariant to the noise schedule except for its endpoints, thus bolstering its density estimation capabilities. Following the latter discovery, Kong et al. (2023) show that DPMs can in-fact be used for *exact* likelihood computation by leveraging techniques from information theory. To further improve DPMs, extensive research has gone into the choice of noise schedules, network architectures and loss functions (Hoogeboom et al., 2023; Karras et al., 2022; Kingma and Gao, 2024; Nichol and Dhariwal, 2021). Many tangentially discovered frameworks such as rectified flows (Liu et al., 2023) and conditional normalizing flows trained with Gaussian conditional flow matching (Lipman et al., 2023), are also particular instances of (Gaussian) diffusion models with specialized noise schedules and weighted loss functions, as shown in (Kingma and Gao, 2024).

2.4 Summary

Score estimation is central to modern generative models, with a variety of both parametric and non-parametric techniques developed to address this task, each possessing unique advantages. Among parametric methods, SSM is widely used for learning the score of the base distribution, while DSM is ideal for learning the score of a noisy distribution and forms the foundation of diffusion models. On the non-parametric side, SSGE provides a straightforward proposal for test functions, and the nonparametric score method encompasses a range of related approaches.

We now have all the necessary tools to start our investigation into exploiting score estimation techniques for the purposes of generative modeling.

Appendix

2.A Proof of Tweedie's Formula

Proof of Theorem 2.1. Consider

$$\begin{aligned}
\mathbf{s}(\mathbf{x}_\sigma) &= \nabla_{\mathbf{x}_\sigma} \log p(\mathbf{x}_\sigma) \\
&= \frac{\nabla_{\mathbf{x}_\sigma} p(\mathbf{x}_\sigma)}{p(\mathbf{x}_\sigma)} \\
&= \int \nabla_{\mathbf{x}_\sigma} p(\mathbf{x}_\sigma|\mathbf{x}) \frac{p(\mathbf{x})}{p(\mathbf{x}_\sigma)} \, \mathrm{d}\mathbf{x} \\
&= \int \frac{\nabla_{\mathbf{x}_\sigma} p(\mathbf{x}_\sigma|\mathbf{x})}{p(\mathbf{x}_\sigma|\mathbf{x})} \frac{p(\mathbf{x}) p(\mathbf{x}_\sigma|\mathbf{x})}{p(\mathbf{x}_\sigma)} \, \mathrm{d}\mathbf{x} \\
&= \int \nabla_{\mathbf{x}_\sigma} \log p(\mathbf{x}_\sigma|\mathbf{x}) p(\mathbf{x}|\mathbf{x}_\sigma) \, \mathrm{d}\mathbf{x} \\
&= \mathbb{E}_{p(\mathbf{x}|\mathbf{x}_\sigma)}[\nabla_{\mathbf{x}_\sigma} \log p(\mathbf{x}_\sigma|\mathbf{x})] \\
&= \mathbb{E}_{p(\mathbf{x}|\mathbf{x}_\sigma)}[\mathbf{s}(\mathbf{x}_\sigma|\mathbf{x})].
\end{aligned}$$

To show the second identity, note that

$$\begin{aligned}
\mathbf{s}^{(2)}(\mathbf{x}_\sigma) &= \nabla^2_{\mathbf{x}_\sigma} \log p(\mathbf{x}_\sigma) \\
&= \frac{\nabla^2_{\mathbf{x}_\sigma} p(\mathbf{x}_\sigma)}{p(\mathbf{x}_\sigma)} - \frac{\nabla_{\mathbf{x}_\sigma} p(\mathbf{x}_\sigma) \nabla_{\mathbf{x}_\sigma} p(\mathbf{x}_\sigma)^\intercal}{p(\mathbf{x}_\sigma)^2} \\
&= \frac{\nabla^2_{\mathbf{x}_\sigma} p(\mathbf{x}_\sigma)}{p(\mathbf{x}_\sigma)} - \mathbf{s}(\mathbf{x}_\sigma)\mathbf{s}(\mathbf{x}_\sigma)^\intercal. \qquad (2.14)
\end{aligned}$$

Applying the same logic from above on the first term, we have

$$\frac{\nabla^2_{\mathbf{x}_\sigma} p(\mathbf{x}_\sigma)}{p(\mathbf{x}_\sigma)} = \mathbb{E}_{p(\mathbf{x}|\mathbf{x}_\sigma)}\Big[\frac{\nabla^2_{\mathbf{x}_\sigma} p(\mathbf{x}_\sigma|\mathbf{x})}{p(\mathbf{x}_\sigma|\mathbf{x})}\Big].$$

However, since

$$\frac{\nabla^2_{\mathbf{x}_\sigma} p(\mathbf{x}_\sigma|\mathbf{x})}{p(\mathbf{x}_\sigma|\mathbf{x})} = \mathbf{s}(\mathbf{x}_\sigma)\mathbf{s}(\mathbf{x}_\sigma)^\intercal + \mathbf{s}^{(2)}(\mathbf{x}_\sigma),$$

rearranging the terms in Eq. (2.14) leads to the desired relation. □

Part I.

Improved Score Estimation

3 Principal Direction Score Estimation

In the previous chapter, we discussed how non-parametric score estimation methods often rely on the careful selection of test functions to apply the Stein gradient estimation framework. However, choosing a suboptimal test function can significantly increase the approximation error of the estimated score, rendering the method ineffective in practical applications. In this chapter, we focus on this limitation within the Spectral Stein Gradient Estimator (SSGE) framework introduced in Section 2.2.3.2. We begin by analyzing the approximation error in SSGE and demonstrate how an ill-suited test function impacts performance. To address this, we draw on tools from elementary linear algebra—specifically, principal component analysis (PCA)—to guide the construction of a more effective estimation strategy. Building on these insights, we introduce a novel extension to SSGE that learns the test functions as the basis of an optimal kernel that minimizes the ℓ_2 approximation error.

3.1 Suboptimality of SSGE

Instead of estimating the test function at the given sample points as in the SGE framework (see Section 2.2.3.1), SSGE considers an orthonormal eigenbasis $\{\phi_i(\mathbf{x})\}_{i=1}^{\infty}$ of a fixed choice of kernel $k(\mathbf{x}, \mathbf{x}')$ with respect to the given distribution $p(\mathbf{x})$. It then aims to estimate the score function by considering the expansion

$$\mathbf{s}(\mathbf{x}) = \sum_{i=1}^{\infty} \mathbf{b}_i \phi_i(\mathbf{x}),$$

where $\mathbf{b}_i := \mathbb{E}_{p(\mathbf{x})}[\phi_i(\mathbf{x})\mathbf{s}(\mathbf{x})]$.

Given N samples drawn from $p(\mathbf{x})$, let $\{(\hat{\lambda}_n, \mathbf{v}_n)\}_{n=1}^{N}$ be the eigenvalue-vector pair of the normalized empirical kernel matrix K/N, where $K_{nj} = k(\mathbf{x}_n, \mathbf{x}_j)$. Then, the SSGE with L components is defined as

$$\hat{\mathbf{s}}^{(L)}(\mathbf{x}) := \sum_{\ell=1}^{L} \hat{\mathbf{b}}_\ell \hat{\phi}_\ell(\mathbf{x}),$$

where we define based on the Nyström method (Baker, 1977; Williams and Seeger, 2000),

$$\hat{\phi}_\ell(\mathbf{x}) := \frac{1}{\hat{\lambda}_\ell} \frac{1}{N} \sum_{n=1}^{N} k(\mathbf{x}, \mathbf{x}_n) v_{\ell n}$$

and

$$\hat{\mathbf{b}}_\ell := -\hat{\mathbb{E}}_{q(\mathbf{x})}[\hat{\nabla}_{\mathbf{x}}\phi_\ell(\mathbf{x})] = -\frac{1}{N}\sum_{n=1}^{N}\hat{\nabla}_{\mathbf{x}}\phi_\ell(\mathbf{x}_n).$$

Here, the gradient $\nabla_{\mathbf{x}}\phi_i(\mathbf{x})$ is similarly estimated as

$$\hat{\nabla}_{\mathbf{x}}\phi_\ell(\mathbf{x}) := \nabla_{\mathbf{x}}\hat{\phi}_\ell(\mathbf{x}) = \frac{1}{\hat{\lambda}_\ell}\frac{1}{N}\sum_{n=1}^{N}\nabla_{\mathbf{x}}k(\mathbf{x},\mathbf{x}_n)v_{\ell n}.$$

The following is the theoretical guarantee for SSGE from the original paper [1]

Theorem 3.1 (Error bound of SSGE)**.** *Assume that $k(\mathbf{x},\cdot)$ and $k(\cdot,\mathbf{x})$ are in the Stein class of $p(\mathbf{x})$. Further, assume that*

$$\max_{i\in[D]}\mathbb{E}_{p(\mathbf{x})}[s_i^2(\mathbf{x})] = \max_{i\in[D]}\sum_{j=1}^{\infty}b_{ij}^2 \le C < \infty,$$

where $s_i(\mathbf{x}) \in L^2(\mathcal{X},p)$. Then if $\lambda_1 > \lambda_2 > \ldots > \lambda_L > 0$, for each $i \in [D]$,

$$\mathbb{E}_{p(\mathbf{x})}[(\hat{s}_i^{(L)}(\mathbf{x}) - s_i(\mathbf{x}))^2] \le \underbrace{L^2\Big(O_p\Big(\frac{1}{N}\Big) + O_p\Big(\frac{C}{\lambda_L\Delta_L^2 N}\Big)\Big) + LO_p\Big(\frac{C}{\lambda_L\Delta_L^2 N}\Big)}_{\textit{estimation error}} + \underbrace{\sum_{\ell=L+1}^{\infty}b_i^2}_{\textit{approximation error}}.$$

Here, $\Delta_L := \min_{1\le\ell\le L}|\lambda_\ell - \lambda_{\ell+1}|$ and $O_p(\cdot)$ is the big-O notation in probability.

Intuitively, if the top-L eigensubspace of a given fixed kernel is not aligned with the target score function, the approximation error cannot be controlled.

3.2 Optimal Kernel Characterization

The estimation error of SSGE is difficult to optimize further, but with an optimal test function the approximation error can be minimized. Let $\{\phi_\ell\}_{\ell\ge1}$ be an orthonormal basis of $\mathcal{L}^2(\mathcal{X})$. In order to overcome the suboptimality of SSGE we wish to minimize the approximation error

$$\begin{aligned}\mathcal{L}(\{\phi_\ell\}_{\ell=1}^L) &:= \sum_{i=1}^{D}\mathbb{E}_{p(\mathbf{x})}[|s_i^{(L)}(\mathbf{x}) - s_i(\mathbf{x})|^2] \\ &= \sum_{\ell\ge L+1}\sum_{i=1}^{D}\mathbb{E}_{p(\mathbf{x})}[s_d(\mathbf{x})\phi_\ell(\mathbf{x})].\end{aligned}$$

Define an operator $\mathcal{T}(\mathbf{x},\mathbf{x}') := \sum_{i=1}^{D}s_i(\mathbf{x})s_i(\mathbf{x}')$. Then, minimizing $\mathcal{L}(\{\phi_\ell\}_{\ell=1}^L)$ becomes equivalent to

$$\max_{\{\phi_\ell\}_{\ell=1}^L}\sum_{\ell=1}^{L}\mathbb{E}_{p(\mathbf{x})}[\phi_\ell(\mathbf{x})\mathcal{T}(\mathbf{x},\mathbf{x}')\phi_\ell(\mathbf{x}')],$$

[1] We found a bug in the guarantee in the original paper and made corrections.

This optimization problem is equivalent to characterizing the top-L eigenfunctions of the operator $\mathcal{T}$. Note that this linear operator is the integral kernel operator induced by the kernel

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{s}(\mathbf{x})^\intercal \mathbf{s}(\mathbf{x}') = \sum_{d=1}^{D} s_d(\mathbf{x}) s_d(\mathbf{x}').$$

The EVD of this kernel is equivalent to the EVD of its dual, which is the second moment matrix [2]

$$\mathsf{C} := \mathbb{E}_{p(\mathbf{x})}[\mathbf{s}(\mathbf{x})\mathbf{s}(\mathbf{x})^\intercal].$$

It is easy to show that the best orthonormal eigenbasis is equivalent to the normalized principal components. With this, the best order-L approximation of $\mathbf{s}(\mathbf{x})$ becomes

$$\mathbf{s}^{(L)}(\mathbf{x}) = \sum_{\ell=1}^{L} \sqrt{\lambda_\ell} \mathbf{u}_\ell \frac{\mathbf{u}_\ell^\intercal \mathbf{s}(\mathbf{x})}{\sqrt{\lambda_\ell}} = \mathsf{U}_{1:L} \mathsf{U}_{1:L}^\intercal \mathbf{s}(\mathbf{x}), \tag{3.1}$$

where $\mathsf{U}_{1:L}$ is a stack of the top-L normalized principal directions. Note, however, we cannot find the principal directions of $\mathbf{s}(\mathbf{x})$ via solving a linear system unlike SSGE, which works with a fixed choice of kernel. The standard streaming PCA algorithms are not applicable, as we only observe samples $\{\mathbf{x}_n\}_{n=1}^N$, without having explicit access to $\mathbf{s}(\mathbf{x}_n)$'s.

3.3 Score Estimation with Principal Directions

To learn the optimal kernel and minimize the approximation error we need to learn the top-L prinicipal directions. To this end, we choose to learn $\mathsf{U} = \mathsf{U}_{1:D}$ with a parametric score model $\mathbf{s}_\theta(\mathbf{x})$. For $\mathsf{V}_{1:L} = [\mathbf{v}_1, \ldots, \mathbf{v}_L] \in \mathbb{R}^{D \times L}$ and a parametric model $\mathbf{g}_\phi \colon \mathcal{X} \to \mathbb{R}^D$, we can consider $\mathbf{s}_\theta(\mathbf{x}) = \mathsf{V}\mathsf{V}^\intercal \mathbf{g}_\phi(\mathbf{x})$ as another score model. Note that $\theta = \{\phi, \mathsf{V}\}$. The resulting SM objective with the "low-rank" score model is

$$\mathcal{L}(\mathsf{V}, \mathbf{g}_\phi) := \frac{1}{2}(\mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}(\mathbf{x}) - \mathsf{V}\mathsf{V}^\intercal \mathbf{g}_\phi(\mathbf{x})\|^2] - \mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}(\mathbf{x})\|^2]).$$

It can be shown that this parameterization suffices to learn the best order-L approximation Eq. (3.1). Let $a \wedge b := \min\{a, b\}$. We can then show the following:

Theorem 3.2. *For $L \geq 1$, if*

$$(\mathsf{V}^\star, \mathbf{g}^\star) \in \arg \min_{\substack{\mathsf{V} \in \mathbb{R}^{D \times L} \\ \mathbf{g} : \mathcal{X} \to \mathbb{R}^D}} \mathcal{L}(\mathsf{V}, \mathbf{g}),$$

we have $\mathsf{V}^\star(\mathsf{V}^\star)^\intercal \mathbf{g}^\star(\mathbf{x}) \equiv \mathsf{U}_{1:L \wedge r} \mathsf{U}_{1:L \wedge r}^\intercal \mathbf{s}(\mathbf{x})$.

[2] Since $\mathbb{E}_{p(\mathbf{x})}[\mathbf{s}(\mathbf{x})] = 0$ under a mild regularity condition, C is exactly the covariance matrix of $\mathbf{s}(\mathbf{x})$, and the top-L EVD of C is exactly equivalent to the top-L PCA of $\mathbf{s}(\mathbf{x})$.

3.3.1 Practical Implementation

In practice, the integration-by-part trick from SM (see Section 2.2.1.1) can be used to rewrite the objective function as

$$\begin{aligned}\mathcal{L}(\mathsf{V}, \mathbf{g}_\phi) &= \mathbb{E}_{p(\mathbf{x})}[\mathrm{tr}(\mathsf{V}\mathsf{V}^\intercal \mathsf{J}_{\mathbf{x}}\mathbf{g}_\phi(\mathbf{x}))] + \frac{1}{2}\mathbb{E}_{p(\mathbf{x})}[\|\mathsf{V}\mathsf{V}^\intercal \mathbf{g}_\phi(\mathbf{x})\|_2^2] \\ &= \sum_{\ell=1}^{L} \mathbb{E}_{p(\mathbf{x})}[\mathbf{v}_\ell^\intercal \mathsf{J}_{\mathbf{x}}\mathbf{g}_\phi(\mathbf{x})\mathbf{v}_\ell] + \frac{1}{2}\mathbb{E}_{p(\mathbf{x})}[\|\mathsf{V}\mathsf{V}^\intercal \mathbf{g}_\phi(\mathbf{x})\|_2^2].\end{aligned}$$

Note that this score model does not require the slicing trick for computational feasibility as in SSM due to the built-in matrix V. This objective function can be computed efficiently via the Jacobian-vector product (JVP) with off-the-shelf autograd packages. Furthermore, to learn the ordered principal directions, we can apply the following trick called *joint nesting*. If we define a new objective

$$\mathcal{L}_{\text{Nested}}(\mathsf{V}_{1:L}, \mathbf{g}_\phi; \mathbf{w}) := \sum_{\ell=1}^{L} w_\ell \mathcal{L}(\mathsf{V}_{1:\ell}, \mathbf{g}_\phi)$$

with some positive weights $\mathbf{w} = (w_1, \ldots, w_L) \in \mathbb{R}^L_{>0}$, minimizing this single objective characterizes the ordered principal directions as the global minimizer.

Theorem 3.3 (Joint nesting)**.** *For $L \geq 1$, if*

$$(\mathsf{V}^\star, \mathbf{g}^\star) \in \arg\min_{\substack{\mathsf{V}\in\mathbb{R}^{D\times L} \\ \mathbf{g}:\mathcal{X}\to\mathbb{R}^D}} \mathcal{L}_{Nested}(\mathsf{V}, \mathbf{g}; \mathbf{w}),$$

we have $\mathbf{v}_\ell^\star(\mathbf{v}_\ell^\star)^\intercal \mathbf{g}^\star(\mathbf{x}) \equiv \mathbf{u}_\ell\mathbf{u}_\ell^\intercal \mathbf{s}(\mathbf{x})$ for each $1 \leq \ell \leq L \wedge r$. If $L > r$, $\mathbf{v}_\ell^\star = 0$ for $r < \ell \leq L$.

3.3.2 Interpretation

The proposed score estimation framework introduces an inductive bias by assuming that the underlying score function $\mathbf{s}(\mathbf{x})$ can be effectively captured by the top-L principal directions. If $\mathbf{s}(\mathbf{x})$ lies approximately in a low-dimensional subspace, the score model $\mathbf{x} \mapsto \mathsf{V}_{1:L}\mathsf{V}_{1:L}^\intercal \mathbf{g}_\phi(\mathbf{x})$ can leverage this structure. However, if $\mathbf{s}(\mathbf{x})$ lacks such low-dimensional structure, then setting $L = D$ would be necessary to capture the full signal, resulting in a model similar to the standard SSM.

Despite the inclusion of additional parameters that capture the principal directions, the model $\mathbf{g}_\theta$ does not need to perfectly capture the base score. It is sufficient for the model to capture only the non-null component of the score, which may be easier to learn in certain scenarios. The learned principal directions do encapsulate inherent structures of the data distribution, which can be leveraged as useful representations for downstream tasks.

3.4 Experiments

We evaluated the proposed method by training implicit autoencoders, a type of implicit generative model. In this section, we describe the experimental setup and present results.

3.4.1 Implicit Autoencoders

3.4.1.1 Variational Autoencoder (VAE)

Traditional VAEs (Kingma and Welling, 2014) are autoencoders with an encoder $\mathbf{f}_\phi : \mathbb{R}^D \to \mathbb{R}^d$ and a decoder $\mathbf{g}_\theta$ mapping in the opposite direction. The decoder induces a likelihood distribution, $p_\theta(\mathbf{x}|\mathbf{z})$ and the latent space is assumed to be regularized with a standard normal prior, $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. The decoder induces a posterior distribution and in practice the encoder is designed to parametrize a variational Gaussian posterior, $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\phi(\mathbf{x}), \sigma^2_\phi(\mathbf{x})\mathbf{I}_d)$, where the mean and covariance are defined by deterministic functions.

In practice, for generative modeling, the Gaussian posterior is restrictive in its expressivity and one can instead define an implicit VAE that leverages a parametric stochastic encoder to sample latent variables as $\mathbf{z} = \mathbf{f}_\phi(\mathbf{x}, \boldsymbol{\epsilon})$, where $p(\boldsymbol{\epsilon})$ is an auxiliary noise distribution. The encoder now prescribes a stochastic procedure to generate latent variables by compromising on an explicit posterior model. The overall training objective remains unchanged and these models are still trained by maximizing the evidence lower bound (ELBO) (Kingma and Welling, 2014),

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})\right] + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[-\log q_\phi(\mathbf{z}|\mathbf{x})\right]}_{\triangleq h(q_\phi(\mathbf{z}|\mathbf{x}))}.$$

Maximization of the ELBO via gradient-based methods involves the computation of the derivative of the differential entropy. Li and Turner (2018) show that this can be readily computed given an estimate of the conditional score and by the fact that the stochasticity is induced by the auxiliary noise,

$$\begin{aligned}\nabla_\phi h(q_\phi(\mathbf{z}|\mathbf{x})) &= -\nabla_\phi \mathbb{E}_{p(\boldsymbol{\epsilon})}[\log q_\phi(\mathbf{f}_\phi(\mathbf{x}, \boldsymbol{\epsilon})|\mathbf{x})] \\ &= -\mathbb{E}_{p(\boldsymbol{\epsilon})}[\mathsf{J}_\phi \mathbf{f}_\phi(\mathbf{x}, \boldsymbol{\epsilon})^\intercal \nabla_{\mathbf{z}} \log q_\phi(\mathbf{z}|\mathbf{x})|_{\mathbf{z}=\mathbf{f}_\phi(\mathbf{x},\boldsymbol{\epsilon})}].\end{aligned}$$

Thus, with a good conditional score estimator $\hat{\mathbf{s}}_\psi(\mathbf{z}|\mathbf{x}) \approx \nabla_{\mathbf{z}} \log q_\phi(\mathbf{z}|\mathbf{x})$, we can approximate the gradient as

$$\nabla_\phi h(q_\phi(\mathbf{z}|\mathbf{x})) \approx -\mathbb{E}_{p(\boldsymbol{\epsilon})}[\mathsf{J}_\phi \mathbf{f}_\phi(\mathbf{x}, \boldsymbol{\epsilon})^\intercal \hat{\mathbf{s}}_\psi(\mathbf{f}_\phi(\mathbf{x}, \boldsymbol{\epsilon})|\mathbf{x})],$$

which in practice can be computed using automatic differentiation techniques.

3.4.1.2 Wasserstein Autoencoder (WAE)

Let $p(\mathbf{z})$ be a fixed prior distribution and $p_\theta(\mathbf{x}|\mathbf{z})$ a likelihood distribution induced by a deterministic decoder $\mathbf{x} = \mathbf{g}_\theta(\mathbf{z})$. Let $\boldsymbol{\mu}_\phi(\mathbf{x})$ model the mean of a Gaussian posterior distribution with variance σ^2 and let $q_\phi(\mathbf{z}) = \mathbb{E}_{p(\mathbf{x})}[q_\phi(\mathbf{z}|\mathbf{x})]$. A WAE (Tolstikhin et al., 2018) seeks to solve the optimal transport (OT) problem,

$$\inf_{q_\phi : q_\phi(\mathbf{z}) = p(\mathbf{z})} \mathbb{E}_{p(\mathbf{x})q_\phi(\mathbf{z}|\mathbf{x})}\left[c(\mathbf{x}, \mathbf{g}_\theta(\mathbf{z}))\right],$$

where $c : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a distance function on a metric space, e.g., the 2-Wasserstein distance $c(x, y) = \|x - y\|^2$. While solving the above optimization problem is hard, a relaxed version can be solved with gradient-based methods,

$$\mathbb{E}_{p(\mathbf{x})q_\phi(\mathbf{z}|\mathbf{x})}\left[c(\mathbf{x}, \mathbf{g}_\theta(\mathbf{z}))\right] + D(q_\phi(\mathbf{z}), p(\mathbf{z})). \tag{3.2}$$

Above, the regularizer is an arbitrary divergence that forces the approximate and true posterior to coincide. In our experiments, we choose the KL divergence $D_{\mathrm{KL}}(q_\phi(\mathbf{z})\|p(\mathbf{z}))$ and leverage an implicit encoder defined via the re-parametrization trick (Kingma and Welling, 2014), $\mathbf{z} = \boldsymbol{\mu}_\theta(\mathbf{x}) + \sigma\boldsymbol{\epsilon}$ for $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. As in implicit VAEs, gradient-based optimization of Eq. (3.2) requires computing the entropy term as,

$$\begin{aligned}\nabla_\phi h(q_\phi(\mathbf{z})) &= -\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}[\log q_\phi(\mathbf{z})] \\ &= -\nabla_\phi \mathbb{E}_{p(\mathbf{x})p(\boldsymbol{\epsilon})}[\log q_\phi(\boldsymbol{\mu}_\phi(\mathbf{x}) + \sigma\boldsymbol{\epsilon})] \\ &= -\mathbb{E}_{p(\mathbf{x})p(\boldsymbol{\epsilon})}[\mathsf{J}_\phi \boldsymbol{\mu}_\phi(\mathbf{x})^\intercal \nabla_{\mathbf{z}} \log q_\phi(\mathbf{z})|_{\mathbf{z}=\boldsymbol{\mu}_\phi(\mathbf{x})+\sigma\boldsymbol{\epsilon}}].\end{aligned}$$

With a good conditional score estimator $\hat{\mathbf{s}}_\psi(\mathbf{z}) \approx \nabla_{\mathbf{z}} \log q_\phi(\mathbf{z})$, we can approximate the gradient as

$$\nabla_\phi h(q_\phi(\mathbf{z})) \approx -\mathbb{E}_{p(\mathbf{x})p(\boldsymbol{\epsilon})}[\mathsf{J}_\phi \boldsymbol{\mu}_\phi(\mathbf{x})^\intercal \hat{\mathbf{s}}_\psi(\boldsymbol{\mu}_\phi(\mathbf{x}) + \sigma\boldsymbol{\epsilon})].$$

3.4.2 Experimental Setup

We trained implicit autoencoders on the MNIST (LeCun, 1998) and the CelebA (Liu et al., 2015) datasets. The MNIST dataset consists of grayscale images of handwritten digits at a resolution of 28×28 whereas the CelebA dataset consists of photos of celebrity faces at a resolution of 64×64.

For our experiments, we used the network architectures for the encoder and decoder from (Shi et al., 2018), replacing all ReLU activations with Swish activations (Ramachandran et al., 2018), which we found to improve performance. We used the provided score backbone architectures[3] to parametrize the model $\mathbf{g}_\theta$. We employed an MLP backbone for MNIST experiments and a CNN backbone for CelebA . We parametrized the principal directions as a matrix and initialized it randomly in the beginning. The principal directions and the model were optimized jointly.

On MNIST, we varied the latent space dimensions and evaluated the model using negative log-likelihood (NLL), while on CelebA, we used FID (Heusel et al., 2017) to assess sample quality. We trained all models on a single NVIDIA 3090 GPU with the Adam optimizer. MNIST models were trained for 200k steps with a learning rate of 1e–3 while CelebA models were trained for 400k steps with a learning rate of 1e–4.

3.4.3 Results

We compared our method against non-parametric baselines like SSGE and Stein, as well as parametric methods such as SSM and vanilla VAE/WAE. As shown in Table 3.1, our approach consistently outperforms SSGE, demonstrating that optimal eigenbasis characterization improves results. Notably, at lower dimensions, our method surpasses SSM, as learning the optimal eigenbasis is easier and provides a practical alternative to Gaussian slicing.

As we scale experiments to 64×64 images, we continue to observe significant performance improvements over SSGE, as indicated by lower FID values in Table 4.3. While SSGE

[3] https://github.com/ermongroup/sliced_score_matching

Table 3.1.: Negative log-likelihoods on the MNIST dataset for conditional and unconditional entropy modeling with different score estimation methods for implicit VAE and WAE training respectively. Results are reported for latent dimension sizes of 8 and 32.

Method	VAE		WAE	
	NLL (8)↓	NLL (32)↓	NLL (8)↓	NLL (32)↓
Gaussian Posterior	96.54	87.73	-	-
Stein	95.95	93.23	98.13	90.66
SSGE	96.14	94.54	98.05	91.47
SSM	95.30	87.60	98.00	**89.39**
PDSE (ours)	**93.91**	**87.40**	**97.01**	91.31

Table 3.2.: FID and ELBO/WAE loss on the test set obtained using different score estimation methods.

Method	VAE		WAE	
	FID↓	ELBO↓	FID↓	WAE↓
Guassian Posterior	52.61	4758	-	-
Stein	91.06	**4553**	49.82	635
SSGE	95.06	4555	49.72	639
SSM	**50.06**	4678	**47.44**	482
PDSE (ours)	52.80	4651	49.72	**319**

performs competitively with SSM in WAE architectures, we see no substantial improvements beyond test loss, suggesting the SSGE basis is well-matched in this case. At higher dimensions, learning the eigenbasis requires careful initialization, and we hypothesize that the latent score becomes less low-rank, which may explain why SSM still outperforms our method in these settings.

3.5 Summary

We introduced principal direction score estimation, a parametric approach that reduces the theoretical approximation error found in non-parametric, spectral-based score estimation methods. The proposed approach is straightforward to implement and builds upon the slicing trick from SSM, but with more structured slicing vectors derived from the principal directions of the optimal kernel operator. Through multiple experiments, we demonstrated that the proposed method outperforms SSGE, highlighting the benefits of trading off less expressive models in favor of more expressive ones that leverage higher computational power.

Appendix

3.A Proof of Joint Nesting

To prove Theorem 3.3 we first present a Lemma and state a more general statement from which the proof will follow. Recall that we assume that $\mathsf{C} := \mathbb{E}_{p(\mathbf{x})}[\mathbf{s}(\mathbf{x})\mathbf{s}(\mathbf{x})^\intercal] = \mathsf{U}\Lambda\mathsf{U}^\intercal = \sum_{\ell=1}^{D} \lambda_\ell \mathbf{u}_\ell \mathbf{u}_\ell^\intercal$ with $\mathsf{U}^\intercal\mathsf{U} = \mathsf{I}$ and $\lambda_1 \geq \ldots \geq \lambda_D \geq 0$. Moreover, we assume that C has rank $r \leq D$.

Lemma 3.4. *Let $p(\mathbf{z})$ be a distribution over $\mathbb{R}^D$ with unit covariance matrix, i.e., $\mathbb{E}_{p(\mathbf{z})}[\mathbf{z}\mathbf{z}^\intercal] = \mathsf{I}$. Then, the singular value expansion of the kernel $k(\mathbf{x}, \mathbf{z}) = \mathbf{z}^\intercal \mathbf{s}(\mathbf{x})$ is*

$$k(\mathbf{x}, \mathbf{z}) = \sum_{\ell=1}^{D} \sigma_\ell \phi_\ell(\mathbf{x}) \psi_\ell(\mathbf{z}),$$

where $\psi_\ell(\mathbf{z}) = \mathbf{u}_\ell^\intercal \mathbf{z}$ and $\phi_\ell(\mathbf{x}) = \frac{1}{\sqrt{\lambda_\ell}} \mathbf{u}_\ell^\intercal \mathbf{s}(\mathbf{x})$ are the orthonormal singular functions and $\sigma_\ell = \sqrt{\lambda_\ell}$ is the ℓ-th singular value for $\ell \in [D]$.

Proof. Consider the symmetrized kernel $k(\mathbf{x}, \mathbf{x}') := \mathbb{E}_{p(\mathbf{z})}[k(\mathbf{x}, \mathbf{z})k(\mathbf{x}', \mathbf{z})] = \mathbf{s}(\mathbf{x})^\intercal \mathbf{s}(\mathbf{x}')$. By the duality, this symmetrized kernel's eigenfunctions has an one-to-one correspondence to those of the matrix $\mathsf{C} = \mathbb{E}_{p(\mathbf{x})}[\mathbf{s}(\mathbf{x})\mathbf{s}(\mathbf{x})^\intercal]$. That is, the eigenvalue expansion of $k(\mathbf{x}, \mathbf{x}')$ is

$$k(\mathbf{x}, \mathbf{x}') = \sum_{\ell=1}^{L} \lambda_\ell \phi_\ell(\mathbf{x}) \phi_\ell(\mathbf{x}'),$$

where $\phi_\ell(\mathbf{x}) := \frac{1}{\sqrt{\lambda_\ell}} \mathbf{u}_\ell^\intercal \mathbf{s}(\mathbf{x})$. To verify that $\phi_\ell(\mathbf{x})$ is the ℓ-th eigenfunction of $k(\mathbf{x}, \mathbf{x}')$ with eigenvalue λ_ℓ, consider $\mathbb{E}_{p(\mathbf{x}')}[k(\mathbf{x}, \mathbf{x}')\phi_\ell(\mathbf{x}')] = \mathbf{s}(\mathbf{x})^\intercal \mathbb{E}_{p(\mathbf{x}')}[\mathbf{s}(\mathbf{x}')\phi_\ell(\mathbf{x}')] = \mathbf{s}(\mathbf{x})^\intercal \frac{1}{\sqrt{\lambda_\ell}} \mathbb{E}_{p(\mathbf{x}')}[\mathbf{s}(\mathbf{x}')\mathbf{s}(\mathbf{x}')^\intercal]\mathbf{u}_\ell = \mathbf{s}(\mathbf{x})^\intercal \frac{1}{\sqrt{\lambda_\ell}} \mathsf{C}\mathbf{u}_\ell = \sqrt{\lambda_\ell}\mathbf{s}(\mathbf{x})^\intercal \mathbf{u}_\ell = \lambda_\ell \phi_\ell(\mathbf{x})$. To verify that $\{\phi_\ell(\mathbf{x})\}_{\ell=1}^{D}$ is orthonormal, note that $\mathbb{E}_{p(\mathbf{x})}[\phi_\ell(\mathbf{x})\phi_{\ell'}(\mathbf{x})] = \frac{1}{\lambda_\ell} \mathbf{u}_{\ell'}^\intercal \mathsf{C}\mathbf{u}_\ell = \mathbf{u}_{\ell'}^\intercal \mathbf{u}_\ell = \delta_{\ell\ell'}$.

Now, the singular value expansion of $k(\mathbf{x}, \mathbf{z})$ must be of the form

$$k(\mathbf{x}, \mathbf{z}) = \sum_{\ell=1}^{D} \sqrt{\lambda_\ell} \phi_\ell(\mathbf{x}) \psi_\ell(\mathbf{z})$$

for some orthonormal functions $\{\psi_\ell(\mathbf{z})\}_{\ell=1}^{L}$. We claim that $\psi_\ell(\mathbf{z}) = \mathbf{u}_\ell^\intercal \mathbf{z}$. To see this, note that by the singular value relation,

$$\begin{aligned}\psi_\ell(\mathbf{z}) &= \frac{1}{\sqrt{\lambda_\ell}} \mathbb{E}_{p(\mathbf{x})}[k(\mathbf{x}, \mathbf{z})\phi_\ell(\mathbf{x})] \\ &= \frac{1}{\lambda_\ell} \mathbf{z}^\intercal \mathbb{E}_{p(\mathbf{x})}[\mathbf{s}(\mathbf{x})\mathbf{s}(\mathbf{x})^\intercal]\mathbf{u}_\ell = \mathbf{z}^\intercal \mathbf{u}_\ell.\end{aligned}$$

This concludes the proof. □

Using this lemma, we can prove a more general statement than Theorem 3.2.

Theorem 3.5. *For $L \geq 1$, if*

$$(\mathsf{B}^*, \mathbf{f}^*) \in \arg\min_{\substack{\mathsf{B} \in \mathbb{R}^{D \times L} \\ \mathbf{f}: \mathcal{X} \to \mathbb{R}^L}} \mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}(\mathbf{x}) - \mathsf{B}\mathbf{f}(\mathbf{x})\|^2],$$

we have $\mathsf{B}^*\mathbf{f}^*(\mathbf{x}) \equiv \mathsf{U}_{1:L \wedge r}\mathsf{U}_{1:L \wedge r}^\intercal \mathbf{s}(\mathbf{x})$.

Proof. Consider the kernel $k(\mathbf{x}, \mathbf{z}) := \mathbf{s}(\mathbf{x})^\intercal \mathbf{z}$ with a distribution $p(\mathbf{z})$ over $\mathbb{R}^D$ with unit covariance matrix, i.e., $\mathbb{E}_{p(\mathbf{z})}[\mathbf{z}\mathbf{z}^\intercal] = \mathsf{I}$. First of all, note that the objective function is

$$\mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}(\mathbf{x}) - \mathsf{B}\mathbf{f}(\mathbf{x})\|^2] = \mathbb{E}_{p(\mathbf{x})p(\mathbf{z})}[(k(\mathbf{x}, \mathbf{z}) - \mathbf{z}^\intercal \mathsf{B}\mathbf{f}(\mathbf{x}))^2],$$

since $\mathbb{E}_{p(\mathbf{z})}[\mathbf{z}\mathbf{z}^\intercal] = \mathsf{I}$ by assumption. Now, by Schmidt's low-rank approximation theorem, the minimizer of the right-hand side objective function $\mathbf{z}^\intercal \mathsf{B}^*\mathbf{f}^*(\mathbf{x})$ must be equivalent to the rank-L approximation of $k(\mathbf{x}, \mathbf{z})$, which is from Lemma 3.4,

$$\mathbf{z}^\intercal \mathsf{B}^*\mathbf{f}^*(\mathbf{x}) \equiv \mathbf{z}^\intercal \mathsf{U}_{1:L}\mathsf{U}_{1:L}^\intercal \mathbf{s}(\mathbf{x}),$$

which implies that

$$\mathsf{B}^*\mathbf{f}^*(\mathbf{x}) \equiv \mathsf{U}_{1:L}\mathsf{U}_{1:L}^\intercal \mathbf{s}(\mathbf{x}),$$

where $\mathsf{U}_{1:L}^\intercal \mathsf{U}_{1:L} = \mathsf{I}_L$. □

Theorem 3.3 then finally follows as a corollary.

Proof of Theorem 3.3. For each $1 \leq \ell \leq L \wedge r$, using Theorem 3.5 we have $\mathsf{V}_{1:\ell}^*(\mathsf{V}_{1:\ell}^*)^\intercal \mathbf{g}^*(\mathbf{x}) = \mathsf{U}_{1:\ell}\mathsf{U}_{1:\ell}^\intercal \mathbf{s}(\mathbf{x})$. In particular this means for $\ell = 1$ that $\mathbf{v}_1^*(\mathbf{v}_1^*)^\intercal \mathbf{g}^*(\mathbf{x}) \equiv \mathbf{u}_1\mathbf{u}_1^\intercal \mathbf{s}(\mathbf{x})$. Thus by inducting we can easily show that $\mathbf{v}_\ell^\star(\mathbf{v}_\ell^\star)^\intercal \mathbf{g}^\star(\mathbf{x}) \equiv \mathbf{u}_\ell\mathbf{u}_\ell^\intercal \mathbf{s}(\mathbf{x})$. □

4 Lifted Residual Score Estimation

Existing score estimation frameworks typically learn the score function $\mathbf{s}(\mathbf{x})$ by minimizing the expected squared ℓ_2 error between the true score and the model estimate $\mathbf{s}_\theta(\mathbf{x})$. This objective is common across both parametric and non-parametric methods, including the Spectral Stein Gradient Estimator (SSGE) and its parametric extension introduced in the previous chapter. As discussed in Section 2.2.3.3, non-parametric methods based on matrix-valued kernel regression, particularly those minimizing the expected Frobenius norm error, have demonstrated superior performance compared to alternative approaches.

Motivated by these insights, this chapter introduces a novel approach that estimates the score in a lifted space, defined by the outer product of a vector-valued function with itself. Specifically, the score estimator, $\mathbf{s}_\theta(\mathbf{x}) \approx \mathbf{s}(\mathbf{x})$, is learned by minimizing the expected squared Frobenius norm between a matrix-valued estimator $\mathbf{s}_\theta(\mathbf{x}_1)\mathbf{s}_\theta(\mathbf{x}_2)^\intercal$ and matrix-valued target $\mathbf{s}(\mathbf{x}_1)\mathbf{s}(\mathbf{x}_2)^\intercal$. As we operate in the lifted space, the resulting estimation framework is called lifted score estimation (LSE).

In addition, inspired by the decomposition of the score into an optimal basis, as discussed in the previous chapter, we propose a complementary technique called Iterative Residual Estimation. This method addresses the practical limitations of parametric models, which may suffer from architectural biases or limited expressivity. To mitigate such limitations, we decompose the score function as a sum of successive residual estimators $\mathbf{s}(\mathbf{x}) \approx \mathbf{s}_{\theta_1^\star}(\mathbf{x}) + \mathbf{s}_{\theta_2^\star}(\mathbf{x}) + \ldots + \mathbf{s}_{\theta_L^\star}(\mathbf{x})$, where each subsequent component $\mathbf{s}_{\theta_i^\star}(\mathbf{x})$, for $i > 1$, is trained to model the residual error left by the preceding estimators.

By combining both of these ideas—the lifted representation and the residual decomposition—we arrive at the Lifted Residual Score Estimator, a flexible and powerful framework for accurate score estimation.

4.1 Lifted Score Estimation

The score estimation frameworks in Section 2.2 estimate the score by minimizing the expected ℓ_2^2-error $\mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2]$. We instead propose a new criterion

$$\mathbb{E}_{p(\mathbf{x}_1)p(\mathbf{x}_2)}[\|\mathbf{s}(\mathbf{x}_1)\mathbf{s}(\mathbf{x}_2)^\intercal - \mathbf{s}_\theta(\mathbf{x}_1)\mathbf{s}_\theta(\mathbf{x}_2)^\intercal\|_F^2]. \tag{4.1}$$

Rather than the standard squared distance in the Euclidean space, we lift the scores to the space of their outer products and consider the squared Frobenius distance in the lifted space; hence, we call it the lifted score estimation (LSE) objective. While the score matching framework was originally proposed for training unnormalized parametric models, our goal is to construct a good score estimator, and we thus use the term estimation instead of matching.

4.1.1 Motivation: Matrix-Kernel Regression

This new criterion can be motivated from the matrix-kernel regression view, which was adopted in the nonparametric score estimator (Zhou et al., 2020) discussed in Section 2.2.3.3. For a matrix-valued kernel $\Gamma(\mathbf{x}, \mathbf{x}') \in \mathbb{R}^{D\times D}$, let $\{\phi_\ell\}_{\ell\geq 1}$ be an orthonormal basis for the set of $\mathbb{R}^D$-valued square-integrable functions over $\mathcal{X}$, i.e., its Mercer expansion (De Vito et al., 2013, Theorem 3.4) is

$$\Gamma(\mathbf{x}, \mathbf{x}') = \sum_{\ell=1}^{\infty} \lambda_\ell \phi_\ell(\mathbf{x})\phi_\ell(\mathbf{x}')^\intercal.$$

Then, the order-L approximation of the target score function $\mathbf{s}(\mathbf{x})$ with respect to the orthonormal basis is

$$\mathbf{s}^{(L)}(\mathbf{x}) := \sum_{\ell=1}^{L} \mathbb{E}_{p(\mathbf{x}')}[\mathbf{s}(\mathbf{x}')^\intercal \phi_\ell(\mathbf{x}')]\phi_\ell(\mathbf{x}),$$

and the approximation error can be written as

$$\begin{aligned}\mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}^{(L)}(\mathbf{x}) - \mathbf{s}(\mathbf{x})\|_2^2] &= \sum_{\ell\geq L+1} \mathbb{E}_{p(\mathbf{x})}[\mathbf{s}(\mathbf{x})^\intercal \phi_\ell(\mathbf{x})]\mathbb{E}_{p(\mathbf{x}')}[\mathbf{s}(\mathbf{x}')^\intercal \phi_\ell(\mathbf{x}')] \\ &= \sum_{\ell\geq L+1} \mathbb{E}_{p(\mathbf{x})p(\mathbf{x}')}[\phi_\ell(\mathbf{x})^\intercal \Gamma^\star(\mathbf{x}, \mathbf{x}')\phi_\ell(\mathbf{x}')],\end{aligned}$$

where we define the matrix-valued kernel

$$\Gamma^\star(\mathbf{x}, \mathbf{x}') := \mathbf{s}(\mathbf{x})\mathbf{s}(\mathbf{x}')^\intercal \in \mathbb{R}^{D\times D}.$$

Hence, to minimize the approximation error one needs to choose $\Gamma^\star(\mathbf{x}, \mathbf{x}')$ as a choice for $\Gamma(\mathbf{x}, \mathbf{x}')$, so that the eigenbasis is aligned with $\Gamma^\star(\mathbf{x}, \mathbf{x}')$. Since the kernel $\Gamma^\star(\mathbf{x}, \mathbf{x}')$ has rank 1 by definition, we can learn a score model $\mathbf{s}_\theta(\mathbf{x})$ by considering the rank-1 approximation error, i.e.,

$$\mathbb{E}_{p(\mathbf{x})p(\mathbf{x}')}\|\Gamma^\star(\mathbf{x}, \mathbf{x}') - \mathbf{s}_\theta(\mathbf{x})\mathbf{s}_\theta(\mathbf{x}')\|_F^2,$$

which is the lifted score estimation objective.

4.1.2 The Lifted Objective Function

The criterion in Eq. (4.1) cannot be implemented as is due to the presence of the unkown ground truth score. Below we explain how to derive a practical objective that only depends on the estimator $\mathbf{s}_\theta$ and samples from the data distribution. We consider estimating the score with lifting, by minimizing the LSE objective

$$\begin{aligned}\mathcal{L}_{\mathsf{LSE}}(\mathbf{s}; \mathbf{s}_\theta) &:= \frac{1}{2}\mathbb{E}_{p(\mathbf{x}_1)p(\mathbf{x}_2)}\left[\|\mathbf{s}(\mathbf{x}_1)\mathbf{s}(\mathbf{x}_2)^\intercal - \mathbf{s}_\theta(\mathbf{x}_1)\mathbf{s}_\theta(\mathbf{x}_2)^\intercal\|_F^2 - \|\mathbf{s}(\mathbf{x}_1)\mathbf{s}(\mathbf{x}_2)^\intercal\|_F^2\right] \\ &= -(\mathbb{E}_{p(\mathbf{x})}[\mathbf{s}^\intercal(\mathbf{x})\mathbf{s}_\theta(\mathbf{x})])^2 + \frac{1}{2}(\mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}_\theta(\mathbf{x})\|_2^2])^2. \qquad (4.2)\end{aligned}$$

Compare this to the original SM loss in Eq. (2.2). Here, the term $\mathbb{E}_{p(\mathbf{x})}[\mathbf{s}^\intercal(\mathbf{x})\mathbf{s}_\theta(\mathbf{x})]$ can be computed using integration by parts $-\mathbb{E}_{p(\mathbf{x})}[\mathrm{tr}(\mathsf{J}_\mathbf{x}\mathbf{s}_\theta(\mathbf{x}))]$ or with Hutchinson's trick (i.e., slicing) as $-\mathbb{E}_{p(\mathbf{x})p(\mathbf{v})}[\mathbf{v}^\intercal\mathsf{J}_\mathbf{x}\mathbf{s}_\theta(\mathbf{x})\mathbf{v}]$. If we deal with a noisy distribution convolved with a Gaussian as in diffusion models, then this cross term can be computed even more efficiently using the DSM trick, as we will show later in this chapter.

4.1.3 Resolving Sign Ambiguity

Note that unlike the SM objective, an optimal $\mathbf{s}_\theta(\mathbf{x})$ for Eq. (4.1) would be proportional to the score $\mathbf{s}(\mathbf{x})$, but with a potential sign flip. That is, the LSE objective is invariant to scaling with a $\{\pm 1\}$-valued function $\kappa\colon \mathcal{X} \to \{\pm 1\}$, since for $\tilde{\mathbf{s}}_\theta(\mathbf{x}) := \kappa(\mathbf{x})\mathbf{s}_\theta(\mathbf{x})$, we have

$$\mathbb{E}_{p(\mathbf{x}_1)p(\mathbf{x}_2)}[\|\mathbf{s}(\mathbf{x}_1)\mathbf{s}(\mathbf{x}_2)^\intercal - \tilde{\mathbf{s}}_\theta(\mathbf{x}_1)\tilde{\mathbf{s}}_\theta(\mathbf{x}_2)^\intercal\|_F^2] = \mathbb{E}_{p(\mathbf{x}_1)p(\mathbf{x}_2)}[\|\mathbf{s}(\mathbf{x}_1)\mathbf{s}(\mathbf{x}_2)^\intercal - \mathbf{s}_\theta(\mathbf{x}_1)\mathbf{s}_\theta(\mathbf{x}_2)^\intercal\|_F^2].$$

This implies that the sign information is absent for each point $\mathbf{x}$ in the LSE framework. To resolve this ambiguity we postulate two solutions:

1. **Track signs:** In practice, we find that it suffices to keep track of a single sign $\kappa \in \{\pm 1\}$ to form a score model, i.e., $\kappa\mathbf{s}_\theta(\mathbf{x}) \approx \mathbf{s}(\mathbf{x})$. An intuition behind the success of this simple heuristic can be given as follows. Suppose that both the underlying score function and a parametric model $\mathbf{s}_\theta(\mathbf{x})$ we assume are sufficiently smooth over $\mathbf{x}$. Then, a sign correction function $\kappa(\mathbf{x})$ must not change the sign too abruptly as $\mathbf{x}$ varies, as otherwise it will violate the smoothness. We empirically find that using a single-sign estimator $\hat{\kappa}_\theta := \mathrm{sgn}(\mathbb{E}_{p(\mathbf{x})}[\mathbf{s}(\mathbf{x})^\intercal\mathbf{s}_\theta(\mathbf{x})])$ works surprisingly well.

2. **Unlifted regularization:** Since SM (and its practical variants such as SSM or DSM) does not have such sign ambiguity, we also propose another solution that minimizes the a combination of the lifted loss and the base SM loss.

$$\mathcal{L}^{\mathrm{reg}}_{\mathsf{LSE}}(\mathbf{s}; \mathbf{s}_\theta) := \mathcal{L}_{\mathsf{LSE}}(\mathbf{s}; \mathbf{s}_\theta) + \lambda\mathcal{L}_{\mathsf{SM}}(\mathbf{s}; \mathbf{s}_\theta), \tag{4.3}$$

   where $\lambda > 0$ is some constant weight factor. In the absence of a sign estimator, the SM regularizer helps mitigate frequent transitions between the two global minima of the lifted loss at the expense of enforcing adherence to the unlifted loss landscape as well.

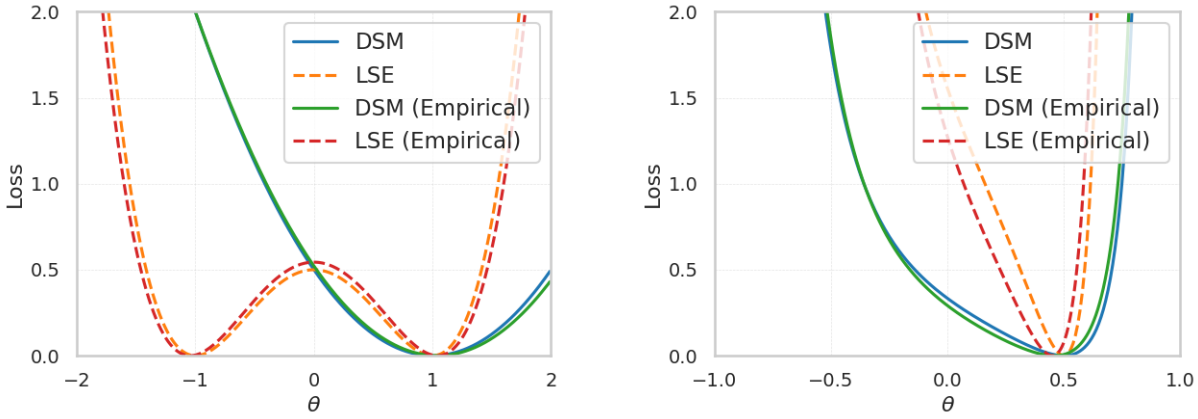
Through experiments on image datasets, we found that tracking the signs worked better in practice for lifted score estimation; see Section 4.5.1. We thus describe our proposed method with this strategy by default.

4.1.4 Analysis of Optimization Landscape

To understand the practical effect of lifting we studied the underlying optimization landscape both theoretically via a closed form population objective and empirically via simulation.

First, consider a Gaussian distribution $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mathbf{0}, \Sigma)$ so that the underlying score is $\mathbf{s}(\mathbf{x}) = -\Sigma^{-1}\mathbf{x}$. To reduce approximation error we assume a well specified score model $\mathbf{s}_\theta(\mathbf{x}) = -\Theta\mathbf{x}$. Then the lifted score estimation objective is,

$$\mathcal{L}(\Theta) := \frac{1}{2}(\mathrm{tr}(\Theta\mathbb{E}_{p(\mathbf{x})}[\mathbf{x}\mathbf{x}^\intercal]\Theta^\intercal))^2 - (\mathrm{tr}(\Theta))^2.$$



(a) 1D Gaussian Landscape

(b) 2D Gaussian Landscape

Figure 4.1.: Population and empirical optimization landscape induced by DSM and LSE when modeling the score of a 1D and 2D Gaussian distribution with a well-specific model.

In the one dimensional setting this becomes

$$\mathcal{L}_{1D}(\theta) := \frac{1}{2}(\sigma^2\theta^2)^2 - \theta^2,$$

where $\Sigma = \sigma$. Both the population and empirical versions of this objective are plotted in Figure 4.3a. Notice that the curvature of the DSM landscape is larger, while the LSE landscape appears sharper. As a result, after sign correction and with a good initialization, LSE might converge more quickly to the global minimum. However, due to the sharpness of the landscape, selecting an appropriate learning rate is crucial to prevent overshooting the minima. In this context, lifting helps to stabilize the solution, effectively resolving the minima in the underlying landscape.

The example can be made slightly more challenging by considering the two dimensional setting. We consider

$$\Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$$

for some $\rho \in (-1, 1)$. Since $\Sigma^{-1} = \frac{1}{1-\rho^2}\begin{bmatrix} 1 & -\rho \\ -\rho & 1 \end{bmatrix}$, we set $\Theta = \frac{1}{1-\theta^2}\begin{bmatrix} 1 & -\theta \\ -\theta & 1 \end{bmatrix}$. Then, the LSE loss function becomes

$$\mathcal{L}_{2D}(\theta) := \frac{1}{2}\Big(\frac{2}{(1-\theta^2)^2}(1 - 2\theta\rho + \theta^2)\Big)^2 - \Big(\frac{2}{1-\theta^2}\Big)^2.$$

Note that the global optimizer is $\theta^\star = \rho$. Again, we plot both the population and empirical version of this objective in Figure 4.3b. The empirical and population curves don't coincide as cleanly in this setting, but lifting still induces a landscape with sharp curvature that could aid in accelerating convergence and resolving the solution.

4.2 Iterative Residual Estimation

In practice, when estimating a score function using a parametric model such as a neural network, there will very likely exist a residual error even with the best possible fit under a criterion. This is especially true in modern applications, where the underlying distribution is extremely high-dimensional and multimodal, such as distributions over images or text.

In such a realistic scenario (also known as model-misspecified case), it is natural to consider learning the residual of the true score after the estimation procedure. For example, suppose that we found the best score model $\mathbf{s}_{\theta_1^\star}(\mathbf{x})$ under a criterion. Defining $\mathbf{r}^{(1)}(\mathbf{x}) := \mathbf{s}(\mathbf{x}) - \mathbf{s}_{\theta_1^\star}(\mathbf{x})$, we can attempt to find the best model $\mathbf{s}_{\theta_2^\star}(\mathbf{x})$[1] that models this residual. Repeating this procedure, we essentially learn the score by a decomposition

$$\mathbf{s}(\mathbf{x}) \approx \mathbf{s}_{\theta_1^\star}(\mathbf{x}) + \mathbf{s}_{\theta_2^\star}(\mathbf{x}) + \ldots + \mathbf{s}_{\theta_L^\star}(\mathbf{x}).$$

Such an additive decomposition is especially natural in the score function domain (i.e., gradient of log probability), as it can be understood as a successive refinement of the underlying distribution, e.g.,

$$\nabla_{\mathbf{x}} \log p(\mathbf{x}) \approx \sum_{i=1}^{L} \nabla_{\mathbf{x}} \log p_{\theta_i^\star}(\mathbf{x}).$$

Here, for the sake of motivation, we hypothetically suppose that $\mathbf{s}_{\theta_\ell^\star}(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_{\theta_\ell^\star}(\mathbf{x})$ for some density model $p_\theta(\mathbf{x})$. From a practical point of view, the residual learning idea provides a way to improve the quality of score estimation by systematically stacking a given base parametric model.

Note that this idea does not assume a specific learning criterion in the subroutine, and it can be paired with the standard SM-type objectives such as SSM and DSM. Somewhat surprisingly we empirically show that the residual learning exhibits significant performance boost with lifting, while the other pairings sometimes only exhibit marginal improvements.

4.3 Lifted Residual Score Estimation

We now combine the lifted objective with residual estimation to produce our proposed framework called lifted residual score estimation which is depicted in Fig. 4.2.

We can extend LSE with residual learning. Let $\mathcal{F} = \{\mathbf{s}_\theta \colon \mathcal{X} \to \mathbb{R}^D \,|\, \theta \in \Theta\}$ denote a class of parametric functions, e.g., a set of functions induced by a neural network architecture. Defining $\mathbf{r}^{(1)}(\mathbf{x}) := \mathbf{s}(\mathbf{x})$ to be the level-1 residual, we can find the best $\mathbf{s}^{(1)} \in \mathcal{F}$ that fits $\mathbf{r}^{(1)}(\mathbf{x})$, i.e.,

$$\mathbf{s}^{(1)} := \arg\min_{\mathbf{s} \in \mathcal{F}} \mathcal{L}_{\mathsf{LSE}}(\mathbf{r}^{(1)}; \mathbf{s}).$$

Recall that $\mathbf{s}^{(1)}$ approximates $\mathbf{r}^{(1)}$ up to a sign flip. Thus, we first estimate the sign as $\kappa^{(1)} := \operatorname{sgn}(\mathbb{E}_{p(\mathbf{x})}[\mathbf{r}^{(1)}(\mathbf{x})^\intercal \mathbf{s}^{(1)}(\mathbf{x})])$, and then define the approximation error as the level-2 residual $\mathbf{r}^{(2)} := \mathbf{r}^{(1)} - \kappa^{(1)}\mathbf{s}^{(1)}$.

[1] We note that despite the notation, the residual model does not estimate the score.

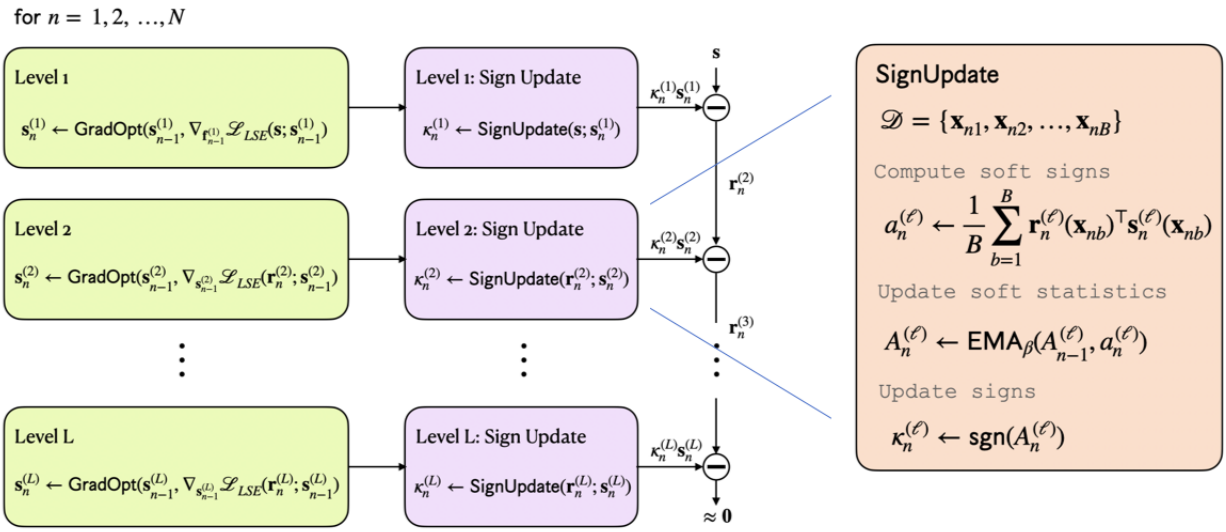


Figure 4.2.: Overview of the proposed lifted residual score estimation framework. At each level an estimator is trained to predict the residual score estimation error. The topmost level estimates the based score model using our proposed lifted score estimation objective in Eq. (4.5).

Now, we can repeatedly apply this learning procedure by considering the residual as a new object to be estimated. For example, for a given level-ℓ residual $\mathbf{r}^{(\ell)}$ for $\ell \geq 1$, we define

$$\mathbf{s}^{(\ell)} = \arg\min_{\mathbf{s} \in \mathcal{F}} \mathcal{L}_{\mathsf{LSE}}(\mathbf{r}^{(\ell)}; \mathbf{s}), \tag{4.4}$$

whereby the level-$(\ell+1)$ residual is

$$\mathbf{r}^{(\ell+1)} := \mathbf{r}^{(\ell)} - \kappa^{(\ell)} \mathbf{s}^{(\ell)} = \mathbf{s} - \sum_{i=1}^{\ell} \kappa^{(i)} \mathbf{s}^{(i)},$$

and $\kappa^{(i)} := \text{sgn}(\mathbb{E}_{p(\mathbf{x})}[\mathbf{r}^{(i)}(\mathbf{x})^\intercal \mathbf{s}^{(i)}(\mathbf{x})]$. For each ℓ, the ℓ-th LSE objective can be explicitly written as

$$\begin{aligned} \mathcal{L}_{\mathsf{LSE}}(\mathbf{r}^{(\ell)}; \mathbf{s}) &= -(\mathbb{E}_{p(\mathbf{x})}[\mathbf{r}^{(\ell)}(\mathbf{x})^\intercal \mathbf{s}(\mathbf{x})])^2 + \frac{1}{2}(\mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}(\mathbf{x})\|_2^2])^2 \\ &= -\Big(\mathbb{E}_{p(\mathbf{x})}[\mathbf{s}(\mathbf{x})^\intercal \mathbf{s}(\mathbf{x})] - \sum_{i=1}^{\ell-1} \kappa^{(i)} \mathbb{E}_{p(\mathbf{x})}[\mathbf{s}^{(i)}(\mathbf{x})^\intercal \mathbf{s}(\mathbf{x})]\Big)^2 + \frac{1}{2}\mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}(\mathbf{x})\|_2^2]^2. \end{aligned} \tag{4.5}$$

Note that the gradient with respect to $\mathbf{s}$ can be computed in an unbiased manner.

Algorithm 4.1 Lifted Score Estimation with Residual Learning

Initialize $\mathbf{s}_0^{(1)}, \dots, \mathbf{s}_0^{(L)} \in \mathcal{F}$.
for $n = 1, \dots, N$ **do**
  Get minibatch samples $\mathcal{D}_n := \{\mathbf{x}_{n1}, \dots, \mathbf{x}_{nB}\}$
  for $\ell = 1, \dots, L$ **do** ▷ Define the level-ℓ residual
    $\mathbf{s}_n^{(\ell)} \leftarrow \mathsf{GradOpt}(\mathbf{s}_{n-1}^{(\ell)}, \hat{\nabla}_{\mathbf{s}_{n-1}^{(\ell)}} \mathcal{L}_{\mathsf{LSE}}(\mathbf{r}_n^{(\ell)}; \mathbf{s}_{n-1}^{(\ell)}))$ ▷ Update the function at the level-ℓ
    $\mathbf{r}_n^{(\ell)} \leftarrow \mathbf{s} - \sum_{i=1}^{\ell-1} \kappa_{n-1}^{(i)} \mathbf{s}_n^{(i)}$ ▷ Define the level-ℓ residual
    $a_n^{(\ell)} \leftarrow \frac{1}{B} \sum_{b=1}^{B} \mathbf{r}_n^{(\ell)}(\mathbf{x}_{nb})^\intercal \mathbf{s}_n^{(\ell)}(\mathbf{x}_{nb})$ ▷ Estimate $\mathbb{E}_{p(\mathbf{x})}[\mathbf{r}_n^{(\ell)}(\mathbf{x})^\intercal \mathbf{s}_n^{(\ell)}(\mathbf{x})]$ w/ minibatch
    $A_n^{(\ell)} \leftarrow \mathsf{EMA}_\beta(A_{n-1}^{(\ell)}, a_n^{(\ell)})$ ▷ Update the soft statistics
    $\kappa_n^{(\ell)} \leftarrow \mathrm{sgn}(A_n^{(\ell)})$ ▷ Update the sign

4.3.1 Practical Optimization Scheme

In practice, solving each level of the optimization problem in a sequential manner as above will incur a $O(L)$-multiplicative computational overhead. A more efficient approach is to emulate solving the series of the optimization problems simultaneously with minibatch samples; see Algorithm 4.1. Here, the gradient of each objective is estimated in an unbiased manner with the given minibatch of size B, and $\mathsf{GradOpt}(\mathbf{s}, \hat{\nabla}_{\mathbf{s}})$ denotes any gradient-based minimization algorithm. The idea is to simultaneously update each level-ℓ model $\mathbf{s}^{(\ell)}$ based on the level-ℓ objective, as if the models from earlier levels in the previous iterates were perfect similar to joint nesting used in Section 3.3.1. For the sign estimates, we maintain the soft statistics $\mathbb{E}_{p(\mathbf{x})}[\mathbf{r}^{(i)}(\mathbf{x})^\intercal \mathbf{s}^{(i)}(\mathbf{x})]$ using an exponential moving average, which we denote as $\mathsf{EMA}_\beta(a_{\mathtt{past}}, a_{\mathtt{new}}) := \beta a_{\mathtt{past}} + (1-\beta) a_{\mathtt{new}}$ below. Ultimately, we construct the final score model as $\hat{\mathbf{s}}_t^{(\ell)}(\mathbf{x}) := \sum_{i=1}^{\ell} \kappa_t^{(i)} \mathbf{s}_t^{(i)}(\mathbf{x})$.

Notation. Residual estimation with L modes (or models) is characterized by a base score estimator $\mathbf{s}_\theta$ along with $L-1$ estimators for the residues. We use the shorthand LSE ($L = k$) to denote the lifted residual score estimation method with k modes.

4.4 Extension to Noisy Score Estimation

We now extend the LSE objective function Eq. (4.2) for the marginal score estimation of samples corrupted with additive noise. Let $p(\mathbf{x}_\sigma|\mathbf{x})$ denote a channel with noise standard deviation σ where $\mathbf{x}_\sigma := \mathbf{x} + \sigma\boldsymbol{\epsilon}$, $\boldsymbol{\epsilon}\ \mathcal{N}(\mathbf{0}, \mathbf{I}_D)$. We are interested in learning the marginal score $\mathbf{s}(\mathbf{x}_\sigma) = \nabla_{\mathbf{x}_\sigma} \log p(\mathbf{x}_\sigma)$ where $p(\mathbf{x}_\sigma) = \int p(\mathbf{x}_\sigma|\mathbf{x}) p(\mathbf{x}) \mathrm{d}\mathbf{x}$.

4.4.1 Lifted Denoising Score Estimation

Consider two such channels with parameters σ_1 and σ_2 respectively. For the ease of exposition, assume $\sigma_1 = \sigma_2 = \sigma$ such that $p(\mathbf{x}_{\sigma_1}) = p(\mathbf{x}_{\sigma_2}) = p(\mathbf{x}_\sigma)$. In the lifted space, the LSE objective

can be used for score estimation,

$$\mathcal{L}_{\mathsf{LSE}}(\mathbf{s};\mathbf{s}_\theta) \overset{(4.2)}{=} -(\mathbb{E}_{p(\mathbf{x}_\sigma)}[\mathbf{s}(\mathbf{x}_\sigma)^\intercal \mathbf{s}_\theta(\mathbf{x}_\sigma)])^2 + \frac{1}{2}(\mathbb{E}_{p(\mathbf{x}_\sigma)}[\|\mathbf{s}_\theta(\mathbf{x}_\sigma)\|^2])^2 \tag{4.6}$$

$$= -(\mathbb{E}_{p(\mathbf{x})p(\mathbf{x}_\sigma|\mathbf{x})}[\mathbf{s}(\mathbf{x}_\sigma|\mathbf{x})^\intercal \mathbf{s}_\theta(\mathbf{x}_\sigma)])^2 + \frac{1}{2}(\mathbb{E}_{p(\mathbf{x}_\sigma)}[\|\mathbf{s}_\theta(\mathbf{x}_\sigma)\|^2])^2. \tag{4.7}$$

Note that Eq. (4.7) follows by Tweedie's formula $\mathbf{s}(\mathbf{x}_\sigma) = \mathbb{E}_{p(\mathbf{x}|\mathbf{x}_\sigma)}[\mathbf{s}(\mathbf{x}_\sigma|\mathbf{x})]$ (see Theorem 2.1), which is the same computational trick exploited in DSM.

4.4.2 Lifted Residual Denoising Score Estimation

Again, let $\mathcal{F}$ denote a collection of parametric vector-valued functions. We apply residual learning to improve the score estimate. For each $\ell \geq 1$,

$$\mathbf{r}^{(\ell)}(\mathbf{x}_\sigma) := \mathbf{s}(\mathbf{x}_\sigma) - \sum_{i=1}^{\ell-1} \kappa_\sigma^{(i)} \mathbf{s}^{(i)}(\mathbf{x}_\sigma).$$

Here, we explicitly assume a dependence of the sign on the noise level σ, as we will apply the framework for learning noisy scores across multiple noise levels in our denoising diffusion model experiments. Using the residual score estimation framework from Sec. 4.3, we can directly apply the ℓ-th residual LSE objective in Eq. (4.5),

$$\mathcal{L}_{\mathsf{LSE}}(\mathbf{r}^{(\ell)};\mathbf{s}) := -\Big(\mathbb{E}_{p(\mathbf{x})p(\mathbf{x}_\sigma|\mathbf{x})}[\mathbf{s}_\sigma(\mathbf{x}_\sigma|\mathbf{x})^\intercal \mathbf{s}(\mathbf{x}_\sigma)] - \sum_{i=1}^{\ell-1} \kappa_\sigma^{(i)} \mathbb{E}_{p(\mathbf{x}_\sigma)}[\mathbf{s}^{(i)}(\mathbf{x}_\sigma)^\intercal \mathbf{s}(\mathbf{x}_\sigma)]\Big)^2$$
$$+ \frac{1}{2}\mathbb{E}_{p(\mathbf{x}_\sigma)}[\|\mathbf{s}(\mathbf{x}_\sigma)\|^2], \tag{4.8}$$

where Eq. (4.8) again follows from Tweedie's formula

4.5 Experiments

We now demonstrate how our proposed methods can be applied to training image generative models. We first experimented with LSE for training diffusion generative models and then experimented with implicit autoencoders. For more background on diffusion models please refer to Section 2.3 and for additional background on implicit autoencoders please refer to Section 3.4.1. We evaluated all methods in terms of sample quality and compared against various score estimation baselines.

4.5.1 Training Diffusion Models

We evaluated our denoising score estimation methods by training the iDDPM (Nichol and Dhariwal, 2021) and the EDM (Karras et al., 2022) diffusion model architectures on the CIFAR-10 dataset (Krizhevsky et al., 2009).

Architecture Details. We implemented our methods on top of the authors' codebase such that their models and each level-ℓ model of the residual LSE model utilized the same architecture. For example, this means that the baseline model defined by the authors and LSE ($L = 1$) have the same network backbone and number of parameters. The only change is the objective function itself. For the iDDPM architecture, we used a simple version of the model that does not leverage any KL regularizers or learned noise schedule. Given this base model, we seek to understand how much improvement our proposed ideas can bring about. We keep the architecture of the EDM model untouched to study how lifting affects the results of state-of-the-art diffusion models.

Training Details. We trained all iDDPM models for 500k iterations on the CIFAR-10 dataset with a batch size of 128. As we are interested in comparing LSE against DSM, we used the "simple" version of the loss defined in (Nichol and Dhariwal, 2021), and didn't include the KL divergence regularization term or the learned noise variance schedule. We also implemented a residual version (see Appendix 4.A). We trained all models with default hyperparameters for unconditional CIFAR-10 used by the iDDPM authors on a single NVIDIA 3090 GPU. All LSE models use a EMA decay of 0.5 for sign estimation.

We compared the same set of models for our EDM experiments on CIFAR-10. All models were trained for 400k iterations with a batch size of 512 distributed across 8 $\times$ NVIDIA V100 GPUs. The baseline EDM model uses the default hyperparameters provided by the authors whereas the residual DSM and LSE models use a smaller learning rate of 0.0001. All LSE models use a EMA decay of 0.5 for sign estimation.

Evaluation. We generated 50,000 samples with each model and measured the FID, sFID (Nash et al., 2021) and Inception Score (IS) (Salimans et al., 2016). This was repeated thrice to account for stochasticity in the results and the best numbers are reported. For the iDDPM models, we used the DDIM sampler (Song et al., 2021a) with $T = 250$ sampling steps as no significant gains in FID were attained for $T > 250$ steps per our experiments and the results in (Nichol and Dhariwal, 2021). We generated 50,000 samples with each model and computed the FID. We did this three times due to the stochasticity of the sampling scheme and reported the best FID for each model. For the EDM models, we used the default sampling noise schedule and the deterministic Heun 2^{nd} order sampler (Ascher and Petzold, 1998) with 18 sampling steps or 35 function evaluations per sample.

4.5.1.1 Results

The results of our iDDPM experiments are presented in Table 4.1. The baseline is the iDDPM model trained with DSM without any lifting or residual score estimation.

Effect of Lifting. The first observation is that LSE ($L = 1$) outperforms the baseline in terms of both FID and sFID, demonstrating that optimization in the lifted space is a viable alternative to standard DSM at no extra computational cost.

Effect of Residual Estimation. The results further validate that iterative residual modeling improves score estimation in both unlifted and lifted spaces. Notably, the residual variant

Table 4.1.: FID, sFID, and Inception Score (IS) of different score estimation methods. DSM ($L = 2$) corresponds to a residual version of DSM with two modes. The first row DSM ($L = 1$) is the baseline based on the standard DSM technique. Best numbers are highlighted in each category.

Method	FID↓	sFID↓	IS↑
DSM ($L = 1$)	8.65	10.92	**9.08**
LSE ($L = 1$)	**6.35**	**5.27**	9.05
DSM ($L = 2$)	**4.53**	**4.64**	9.06
LSE ($L = 2$)	4.88	5.88	**9.16**
Ablation: Lifting w/ DSM regularization			
LSE ($L = 1$)	**5.23**	5.37	**9.16**
LSE ($L = 2$)	**5.23**	**4.62**	9.08

of DSM achieves the best FID, while LSE also outperforms the baseline. While FID reflects the global coherence of generated samples, sFID is more sensitive to local variations between images. Overall, we observe that residual estimation does not significantly degrade sFID and, in the case of DSM, can even lead to substantial improvements.

Choosing Between Lifting and Residual Estimation. The results indicate that both lifting and residual estimation serve as effective strategies for improving a weak baseline estimator. Notably, the non-residual (i.e., with $L = 1$) LSE objective incurs no additional computational cost compared to DSM, whereas residual variants (i.e., with $L = 2$) double the memory overhead. Thus, the choice between these approaches depends on the practitioner's resource constraints, with either method offering a viable path to improved performance.

Sample Diversity. We found that LSE enhances the diversity of generated images, as evidenced by a higher Inception score. However, enabling residual estimation leads to a decline in this score. While modeling the error reduces training loss, it may slightly impair generalization, which we hypothesize as the underlying cause of this trend.

Ablation study. We conducted an ablation study on methods for mitigating the sign flip issue. Specifically, we compared LSE with sign tracking to LSE with a regularizer, as described in Section 4.1. Our findings suggest that explicitly tracking signs is more effective in practice, likely because the regularizer constrains the optimization process, limiting exploration of the lifted objective's landscape.

Compared to our modified iDDPM experiments, the EDM architecture is an instance of a state-of-the-art score estimator. Hence, it is not suprising that our results varied quite marginally with residual estimation. We observed that the learned residuals were very nearly zero everywhere suggesting that the base score estimator is excellently modeled in practice. As

Table 4.2.: FID, sFID and Inception Score (IS) of EDM trained with DSM and LSE for $L \in \{1, 2\}$.

Method	FID↓	sFID↓	IS↑
DSM ($L = 1$)	**2.21**	3.82	**9.59**
DSM ($L = 2$)	2.25	3.83	9.56
LSE ($L = 1$)	2.25	**3.80**	9.48
LSE ($L = 2$)	2.25	3.84	9.52

shown in Table 4.2 we once again notice some trends with lower sFID with lifting suggesting that the lifting helps in enforcing greater spatial diversity amongst samples. In general the EDM score estimator is very powerful and we show that we can achieve comparable performance with lifting as well.

4.5.2 Training Implicit Generative Models

We trained implicit VAEs and WAEs with our proposed score estimation method. We are particularly interested in understanding the empirical performance of the lifted objective as is and hence do not use the SSM regularized objective mentioned in Section 4.1.3 in this setting. We compared LSE against various baselines in terms of sample quality on the CelebA dataset (Liu et al., 2023) as evaluated by the FID.

Architecture details. We used the same experimental setup in Section 3.4. Detailed network architecture and implementation details can be found in Appendix 4.B. We experimented with $L = 2$ and $L = 3$ for all residual score models. All experiments used a latent dimension size of $D_{\mathbf{z}} = 32$ and the number of channel maps was set to $m = 64$. As alluded to in Section 4.1, for training WAEs, we found it sufficient to keep track of a single sign to form the marginal score model for the latent variable $\mathbf{z}$. Surprisingly, in the VAE setting, despite learning the conditional score we observed that continuing to keep track of single sign that is independent of $\mathbf{x}$ worked well in practice too.

Training details. We trained all models with a batch size of 128 for 400k iterations on 1× NVIDIA 3090 GPU. All methods compared under a generative model class (i.e., VAE or WAE) utilized the same encoder and decoder architectures. The only difference lies in the score estimation algorithm.

Results. We quantified the sample quality by computing the FID on synthesized samples, as reported in Table 4.3. Also reported is the average value of the ELBO or WAE objective. The best results were obtained with LSE and its residual versions, where $L = 3$ and $L = 2$ outperform all other VAE and WAE models respectively. Surprisingly, we noticed that residual learning paired with lifted score estimation offers significant gains in FID

Table 4.3.: FID and ELBO/WAE loss on the test set obtained using different score estimation methods. $L = 1$ denotes the non-residual variant, and the residual variants of SSM and LSE are denoted by the descriptors $L \in \{2, 3\}$, where $L - 1$ denotes the number of residual errors that are modeled.

Method	VAE		WAE	
	FID↓	**ELBO↓**	**FID↓**	**WAE↓**
Guassian Posterior	52.61	4758	-	-
Stein	91.06	4553	49.82	635
SSGE	95.06	4555	49.72	639
SSM ($L = 1$)	50.06	4678	47.44	482
SSM ($L = 2$)	50.21	4604	47.02	434
SSM ($L = 3$)	49.74	4541	46.27	**352**
LSE ($L = 1$)	50.72	4721	46.46	433
LSE ($L = 2$)	47.68	**4531**	**44.36**	497
LSE ($L = 3$)	**46.81**	4711	45.56	583

in comparison to residual SSM. In general, even the $L = 1$ version of LSE was on par or outperformed SSM, demonstrating that optimization in the lifted space is empirically beneficial for score estimation. We observed that the residual versions could sometimes lead to a slight degradation in performance and training instability as we continue to increase L beyond a limit, which was $L = 3$ in these experiments. We postulate that this is attributed to the inability of the neural network to learn useful representations if the residuals are too small and noisy. Synthesized samples from different models are included in Appendix 4.B.

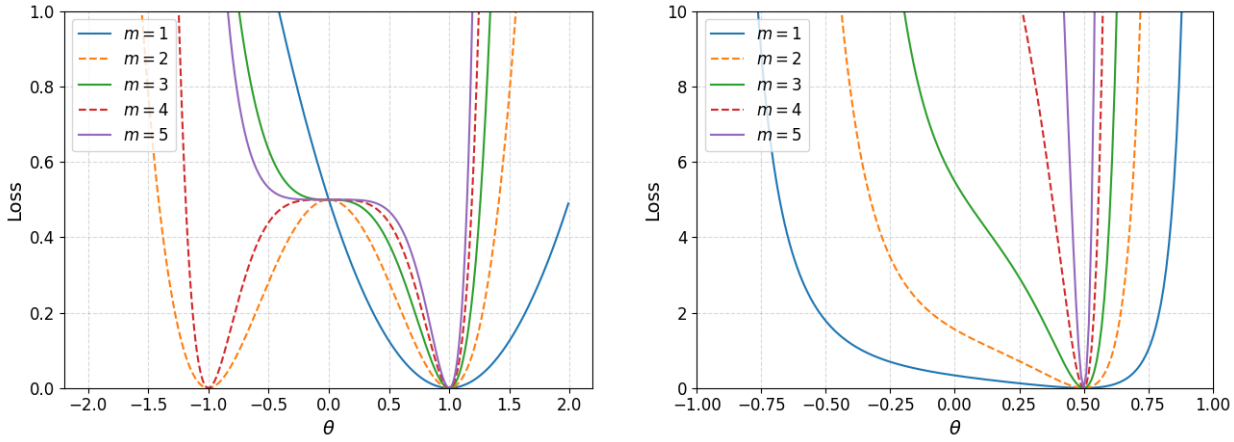
4.6 Summary

We introduced a new score estimation framework built on two core ideas: lifting and residual learning. The resulting method achieves competitive performance in generative modeling, effectively improving over SM when used as a drop-in replacement. Our theoretical analysis shows that LSE can enhance optimization by promoting faster convergence to the global minimum, thanks to the sharper curvature induced by lifting. Empirically, LSE demonstrates improved score approximation, suggesting it can reliably substitute conventional methods such as SSM or DSM in practice.

4.7 Higher-Order Lifting*

We can generalize lifting to the space of tensor products by defining the lifted objective of order m as,

$$\mathbb{E}_{p(\mathbf{x}_1)p(\mathbf{x}_2)\dots p(\mathbf{x}_m)}[\|\mathbf{s}(\mathbf{x}_1) \otimes \mathbf{s}(\mathbf{x}_2) \otimes \cdots \otimes \mathbf{s}(\mathbf{x}_m) - \mathbf{s}_\theta(\mathbf{x}_1) \otimes \mathbf{s}_\theta(\mathbf{x}_2) \otimes \cdots \otimes \mathbf{s}_\theta(\mathbf{x}_m)\|_{F*}^2],$$



(a) 1D Gaussian Landscape

(b) 2D Gaussian Landscape

Figure 4.3.: Population landscape induced by LSE of order m where $m = 1, 2, 3, 4, 5$. LSE ($m = 1$) corresponds to SM and LSE ($m = 2$) corresponds to lifting in the space of outer products. As the order increases the curvature at the global minima becomes sharper.

where $\|\cdot\|_{F*}$ denotes an extension of the Frobenius norm to higher-order tensors. When $m = 2$, this reduces to the LSE objective introduced earlier in this chapter.

While conceptually straightforward, this higher-order objective is computationally expensive to evaluate directly. However, as with the original LSE formulation, we can leverage integration by parts to derive a more tractable surrogate objective that depends only on the estimator $\mathbf{s}_\theta$ and samples from the data distribution.

$$\mathcal{L}^{(m)}_{\mathsf{LSE}}(\mathbf{s}; \mathbf{s}_\theta) := -(\mathbb{E}_{p(\mathbf{x})}[\mathbf{s}^\intercal(\mathbf{x})\mathbf{s}_\theta(\mathbf{x})])^m + \frac{1}{2}(\mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}_\theta(\mathbf{x})\|_2^2])^m.$$

A noisy extension can be similarly derived by leveraging the DSM trick.

Returning to the analysis in Section 4.1.4, we define the order-m m LSE objectives for both 1D and 2D Gaussian examples. The corresponding population loss landscapes are shown in Figure 4.3. Here, the $m = 1$ curve recovers SM, while $m = 2$ reflects lifting in the space of outer products. As the order increases, the curvature of the landscape near the global minima becomes progressively sharper.

This behavior suggests a potential practical advantage: higher-order lifting can lead to faster convergence during optimization by inducing a sharper curvature around the global minima. However, sharper minima also bring increased sensitivity to optimization hyperparameters, potentially leading to instability or divergence if not tuned carefully.

Looking ahead, several promising directions remain. In particular, exploring the practical trade-offs involved in higher-order lifting and how the effect of residual learning impacts estimation quality with higher-orders of lifting are both valuable avenues for future research.

Appendix

4.A Residual Sliced and Denoising Score Matching

In Sections 4.3 and 4.4 we introduced the residual version of LSE with slicing and the DSM trick respectively. We can similarly define the residual score estimation procedure in the original (unlifted) space for both SSM and DSM.

4.A.1 Residual Sliced Score Matching

Recall the exact SM objective which we redefine as follows,

$$\mathcal{L}_{\mathsf{SM}}(\mathbf{s}; \mathbf{s}_\theta) = -\mathbb{E}_{p(\mathbf{x})}[\mathbf{s}(\mathbf{x})^\intercal \mathbf{s}_\theta(\mathbf{x})] + \frac{1}{2}\mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}_\theta(\mathbf{x})\|^2].$$

We can now extend this objective with residual learning and the slicing trick from SSM (see Sec. 2.2.1). Let $\mathcal{F} = \{\mathbf{s}_\theta \colon \mathcal{X} \to \mathbb{R}^D | \theta \in \theta\}$ denote a class of parametric functions and let $\mathbf{r}^{(1)}(\mathbf{x}) := \mathbf{s}(\mathbf{x})$ be the level-1 residual. We seek to find the best $\mathbf{s}^{(1)} \in \mathcal{F}$ that fits $\mathbf{r}^{(1)}(\mathbf{x})$, i.e.,

$$\mathbf{s}^{(1)} := \arg\min_{\mathbf{s} \in \mathcal{F}} \mathcal{L}_{\mathsf{SM}}(\mathbf{r}^{(1)}; \mathbf{s}).$$

Up to this point we have just minimized the standard SM objective. Now, after obtaining $\mathbf{s}^{(1)}$, we can define the approximation error as the level-2 residual $\mathbf{r}^{(2)} := \mathbf{r}^{(1)} - \mathbf{s}^{(1)}$. Then, we can repeatedly apply this learning procedure by considering the residual as a new object to be estimated. In general, for a given level-ℓ residual $\mathbf{r}^{(\ell)}$ for $\ell \geq 1$, we define

$$\mathbf{s}^{(\ell)} = \arg\min_{\mathbf{s} \in \mathcal{F}} \mathcal{L}_{\mathsf{SM}}(\mathbf{r}^{(\ell)}; \mathbf{s}).$$

Expanding out the objective,

$$\begin{aligned}
\mathcal{L}_{\mathsf{SM}}(\mathbf{r}^{(\ell)}; \mathbf{s}) &= -\mathbb{E}_{p(\mathbf{x})}[\mathbf{r}^{(\ell)}(\mathbf{x})^\intercal \mathbf{s}(\mathbf{x})] + \frac{1}{2}\mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}(\mathbf{x})\|_2^2] \\
&= -\Big(\mathbb{E}_{p(\mathbf{x})}[\mathbf{s}(\mathbf{x})^\intercal \mathbf{s}(\mathbf{x})] - \sum_{i=1}^{\ell-1} \mathbb{E}_{p(\mathbf{x})}[\mathbf{s}^{(i)}(\mathbf{x})^\intercal \mathbf{s}(\mathbf{x})]\Big) + \frac{1}{2}\mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}(\mathbf{x})\|_2^2], \qquad (4.9)
\end{aligned}$$

where in Eq. (4.9) we use the slicing trick to compute the first term. We call this objective with the slicing trick, the residual sliced score matching estimator and formally define it as,

$$\mathcal{L}_{\mathsf{SSM}}(\mathbf{r}^{(\ell)}; \mathbf{s}) := -\Big(\mathbb{E}_{p(\mathbf{w})p(\mathbf{x})}[\mathbf{w}^\intercal \mathbf{J}_{\mathbf{x}} \mathbf{s}(\mathbf{x}) \mathbf{w}] - \sum_{i=1}^{\ell-1} \mathbb{E}_{p(\mathbf{x})}[\mathbf{s}^{(i)}(\mathbf{x})^\intercal \mathbf{s}(\mathbf{x})]\Big) + \frac{1}{2}\mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}(\mathbf{x})\|_2^2].$$

Henceforth, we will refer to our SSM extension with $\ell - 1$ residuals as SSM ($L = \ell$). Algorithm 4.2 describes the overall procedure for residual SSM.

Algorithm 4.2 Iterative Residual Sliced Score Matching

Initialize $\mathbf{s}_0^{(1)}, \dots, \mathbf{s}_0^{(L)} \in \mathcal{F}$.
for $n = 1, \dots, N$ **do**
  Get each minibatch samples $\mathcal{D}_n := \{\mathbf{x}_{n1}, \dots, \mathbf{x}_{nB}\}$
  for $\ell = 1, \dots, L$ **do**
    $\mathbf{s}_t^{(\ell)} \leftarrow \mathsf{GradOpt}(\mathbf{s}_{n-1}^{(\ell)}, \hat{\nabla}_{\mathbf{s}_{n-1}^{(\ell)}} \mathcal{L}_{\mathsf{SSM}}(\mathbf{r}_n^{(\ell)}; \mathbf{s}_{n-1}^{(\ell)}))$ ▷ Update the function at the level-ℓ
    $\mathbf{r}_n^{(\ell)} \leftarrow \mathbf{s} - \sum_{i=1}^{\ell-1} \mathbf{s}_n^{(i)}$ ▷ Define the level-ℓ residual

4.A.2 Residual Denoising Score Matching

Let $p(\mathbf{x}_\sigma|\mathbf{x})$ be a noisy channel. DSM learns the marginal score of noisy samples, $\mathbf{s}(\mathbf{x}_\sigma) = \nabla_{\mathbf{x}_\sigma} \log p(\mathbf{x}_\sigma)$ where $p(\mathbf{x}_\sigma) = \int p(\mathbf{x}_\sigma|\mathbf{x})p(\mathbf{x})\mathrm{d}\mathbf{x}$. Again, let $\mathcal{F}$ denote a collection of parametric vector-valued functions. We can apply residual learning to improve the DSM score estimate. For each $\ell \geq 1$,

$$\mathbf{r}^{(\ell)}(\mathbf{x}_\sigma) := \mathbf{s}(\mathbf{x}_\sigma) - \sum_{i=1}^{\ell-1} \mathbf{s}^{(i)}(\mathbf{x}_\sigma).$$

Using the residual score estimation framework introduced in the previous section, the ℓ-th residual DSM objective is,

$$\begin{aligned}&\mathcal{L}_{\mathsf{DSM}}(\mathbf{r}^{(\ell)}; \mathbf{s}) \\ &:= -\Big(\mathbb{E}_{p(\mathbf{x})p(\mathbf{x}_\sigma|\mathbf{x})}[\mathbf{s}_\sigma(\mathbf{x}_\sigma|\mathbf{x})^\intercal \mathbf{s}(\mathbf{x}_\sigma)] - \sum_{i=1}^{\ell-1} \mathbb{E}_{p(\mathbf{x}_\sigma)}[\mathbf{s}^{(i)}(\mathbf{x}_\sigma)^\intercal \mathbf{s}(\mathbf{x}_\sigma)]\Big)^2 + \frac{1}{2}\mathbb{E}_{p(\mathbf{x}_\sigma)}[\|\mathbf{s}(\mathbf{x}_\sigma)\|^2], \quad (4.10)\end{aligned}$$

where in Eq. (4.10) we use the DSM trick arising from Tweedie's formula.

4.B Network Architecture and Additional Results

4.B.1 Network Architecture for Implicit Autoencoder Experiments

Generative Model	Name	Configuration
VAE	Implicit Encoder	5×5 conv; m maps; Swish 5×5 conv; $2m$ maps; Swish 5×5 conv; $4m$ maps; Swish 5×5 conv; $8m$ maps; Swish 512 Dense, Swish $D_{\mathbf{z}}$ Dense
WAE	Encoder	concat [$\mathbf{x}$, Swish(Dense($\boldsymbol{\epsilon}$))] along channels 5×5 conv; $2m$ maps; Swish 5×5 conv; $4m$ maps; Swish 5×5 conv; $8m$ maps; Swish 512 Dense, Swish $D_{\mathbf{z}}$ Dense
VAE and WAE	Decoder	Dense, Swish 5×5 conv$^\intercal$; $4m$ maps; Swish 5×5 conv$^\intercal$; $2m$ maps; Swish 5×5 conv$^\intercal$; $1m$ maps; Swish 5×5 conv$^\intercal$; c maps; Tanh
VAE	SSM Score ($\mathbf{s}_\theta(\mathbf{z}\|\mathbf{x})$) Res. SSM and LSE Models ($\mathbf{s}_\theta^{(\ell)}(\mathbf{z}\|\mathbf{x})$)	concat [$\mathbf{x}$, Swish(Dense($\mathbf{z}$))] along channels 5×5 conv; m maps; Swish 5×5 conv; $4m$ maps; Swish 5×5 conv; $8m$ maps; Swish 512 Dense, Swish $D_{\mathbf{z}}$ Dense
WAE	SSM Score ($\mathbf{s}_\theta(\mathbf{z})$) Res. SSM and LSE Models ($\mathbf{s}_\theta^{(\ell)}(\mathbf{z})$)	Reshape(Swish(Dense($\mathbf{z}$))] to 1 channel 5×5 conv; m maps; Swish 5×5 conv; $4m$ maps; Swish 5×5 conv; $8m$ maps; Swish 512 Dense, Swish $D_{\mathbf{z}}$ Dense

Figure 4.4.: Implicit VAE and WAE architectures for CelebA. All convolutions and transposed convolutions use a stride of 2 with appropriate padding dimensions to preserve feature map spatial resolution.

4.B.2 EDM Diffusion Architecture

The EDM preconditioning diffusion model utilizes a base DDPM++ architecture from (Song et al., 2021b) for CIFAR-10 and the ADM architecture (Nichol and Dhariwal, 2021) for higher resolution images such as ImageNet 64×64. The EDM model uses a noise schedule that is defined as

$$\log \sigma_t \sim \mathcal{N}(-1.2, 1.2^2). \tag{4.11}$$

Rather than regressing against the unscaled additive noise as in DSM, EDM regresses against the original sample expressed in the following form,

$$\mathbf{x} = \frac{\sigma_{\text{data}}^2}{\sigma_t^2 + \sigma_{\text{data}}^2}\mathbf{x}_t + \frac{\sigma_t \cdot \sigma_{\text{data}}}{\sqrt{\sigma_t^2 + \sigma_{\text{data}}^2}}\mathbf{g}, \tag{4.12}$$

where $\sigma_{\text{data}} = 0.5$. To this end, EDM is parametrized with a denoising neural network,

$$\mathbf{f}_\theta(\mathbf{x}_t; t) = \frac{\sigma_{\text{data}}^2}{\sigma_t^2 + \sigma_{\text{data}}^2}\mathbf{x}_t + \frac{\sigma_t \cdot \sigma_{\text{data}}}{\sqrt{\sigma_t^2 + \sigma_{\text{data}}^2}}\mathbf{g}_\theta(\mathbf{x}_t; t), \tag{4.13}$$

which is trained by minimizing

$$\min_\theta \mathbb{E}_{p(\mathbf{x})q(\boldsymbol{\epsilon})p(t)}[w_{\text{EDM}}(t)\|\mathbf{x} - \mathbf{f}_\theta(\mathbf{x}_t; t)\|^2],$$

where

$$w_{\text{EDM}}(t) := \frac{\sigma_t \sigma_{\text{data}}}{\sqrt{\sigma_t^2 + \sigma_{\text{data}}^2}}. \tag{4.14}$$

This is equivalent to estimating $\mathbf{g}$ by minimizing the objective,

$$\mathcal{L}_{\text{EDM}}(\mathbf{g}_\theta) := \mathbb{E}_{p(\mathbf{x})q(\boldsymbol{\epsilon})p(t)}\left[\|\mathbf{g} - \mathbf{g}_\theta\left(\mathbf{x}_t; t\right)\|^2\right]. \tag{4.15}$$

Using Eq. (4.11) and Eq. (4.12) we can show that,

$$\mathbf{g} = \frac{\sqrt{\sigma_t^2 + \sigma_{\text{data}}^2}}{\sigma_t \sigma_{\text{data}}}\mathbf{x} - \frac{\sigma_{\text{data}}}{\sigma_t\sqrt{\sigma_t^2 + \sigma_{\text{data}}^2}}\mathbf{x}_t \tag{4.16}$$

$$= -\frac{\sqrt{\sigma_t^2 + \sigma_{\text{data}}^2}}{\sigma_{\text{data}}}\boldsymbol{\epsilon} + \frac{\sigma_t}{\sqrt{\sigma_t^2 + \sigma_{\text{data}}^2}\sigma_{\text{data}}}\mathbf{x}_t. \tag{4.17}$$

Therefore, in terms of Eq. (2.12) the EDM objective boils down to the standard denoising diffusion objective with weighting function,

$$w(t) = \frac{\sigma_t^2 + \sigma_{\text{data}}^2}{\sigma_{\text{data}}^2}. \tag{4.18}$$

4.B.3 CelebA - VAE

Figure 4.5.: VAE samples on CelebA.

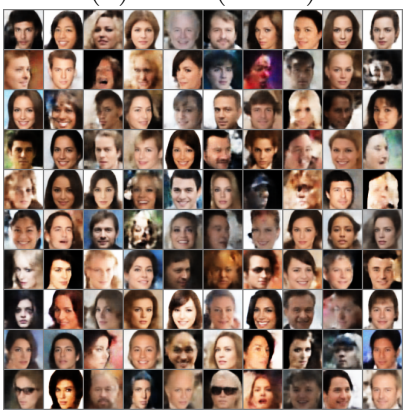
4.B.4 CelebA - WAE

(a) Gaussian Posterior (N/A)

(b) Stein

(c) Spectral

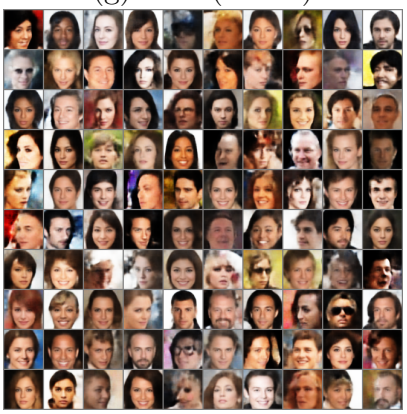
(d) SSM ($L = 1$)



(e) SSM ($L = 2$)

(f) SSM ($L = 3$)

(g) LSE ($L = 1$)



(h) LSE ($L = 2$)

(i) LSE ($L = 3$)

Figure 4.6.: WAE samples on CelebA.

Part II.

Score-based Signal Priors

5 Score-based Source Separation

Thus far, we have explored score estimation primarily through the lens of minimizing approximation error and enhancing the downstream sample quality of generative models. However, the applications of score estimation extend far beyond merely facilitating sampling. In this chapter, we revisit the signal recovery problem from Chapter 1 and introduce a novel algorithm designed to solve the pervasive challenge of single-channel source separation, a fundamental problem in engineering. At the end of this chapter we describe how the proposed framework can be extended to solve general inverse problems.

5.1 Single-Channel Source Separation

The problem of single-channel source separation (SCSS) arises in many different applications, ranging from the cocktail party problem in the audio domain to interference mitigation in the digital communications domain. Broadly speaking, the goal in SCSS is to decompose a mixture

$$\mathbf{y} = \kappa_1 \mathbf{x}_1 + \cdots + \kappa_K \mathbf{x}_K + \mathbf{w} \in \mathcal{Y}^D$$

into its K constituent components, $\{\mathbf{x}_i\}_{i=1}^K, \mathbf{x}_i \in \mathcal{X}_i^D$. The scalars $\kappa_1, \kappa_2, \ldots \kappa_K$ are mixing coefficients that affect the relative levels at which the different constituent components mix and $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_D)$ is some background noise. In the first part of this chapter, we first consider a mixture composed of two superimposed sources,

$$\mathbf{y} = \mathbf{x} + \kappa \mathbf{n} + \mathbf{w}, \tag{5.1}$$

where $\kappa \in \mathbb{R}_+$ is the relative scaling coefficient between the two signals. We will term $\mathbf{x}$ as the *signal of interest* (SOI) and $\mathbf{n}$ as the *interference signal*. For the time being, we also assume no background noise or that the affect of the noise is captured by the interference signal. This gives rise to the forward model,

$$\mathbf{y} = \mathbf{x} + \kappa \mathbf{n}. \tag{5.2}$$

To reiterate, in this context, the goal is to recover the SOI (and hence the interference) from $\mathbf{y}$. We will assume knowledge of κ beforehand. This is a reasonable assumption to make as an estimate of the relative noise level can be made by a simple detector or classifier.

Relationship to Signal Recovery Problem. Recall the mixture model in Eq. (1.1). In this case, the forward operator is simply the identity matrix, and the noise is no longer

necessarily modeled as Gaussian. Instead, it may represent a more complex, structured interference signal for which a closed-form likelihood may not be available. In this chapter, we continue to adopt a Bayesian perspective to tackle such inverse problems. We focus in particular on the SCSS problem, which introduces additional structure that further refines and specializes the underlying optimization framework. We will end the chapter with a discussion on how the proposed framework can be leveraged for solving more general inverse problems.

5.1.1 Prior Work

Prior art that uses deep learning for source separation problems has been well-studied in the audio and image domain by leveraging domain-specific structures.

A popular framework for source separation is based on supervised learning. Given a dataset of paired mixture and SOI pairs $\mathcal{D} = \{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \ldots, (\mathbf{x}^{(N)}, \mathbf{y}^{(N)})\}$ supervised learning methods or end-to-end methods learn a one-step separator $\mathbf{f}_\theta(\mathbf{y})$ by solving,

$$\min_\theta \mathbb{E}_{p_{\mathbf{x},\mathbf{y}}(\mathbf{x},\mathbf{y})}[\|\mathbf{x} - \mathbf{f}_\theta(\mathbf{y})\|_2^2]. \tag{5.3}$$

In fact, recent end-to-end methods in the audio domain leverage separability in the frequency domain and use spectrogram masking techniques to separate sparse time-frequency representations of the audio instead (Chandna et al., 2017; Tzinis et al., 2020). Similarly, specialized architectures have been developed for end-to-end source separation of digital communication signals (Lancho et al., 2024). These methods are mixture specific and do not translate well to other sources with different joint statistics.

Another class of methods leverage structural priors to introduce inductive biases in the solution as discussed in Section 1.1.1. For example, in the visual domain, natural images may be separable by exploiting local features and "smoothness" assumptions in the color space (Gandelsman et al., 2019; Zou et al., 2020). Rather than optimizing over the ambient space, these methods restrict the search space to range of a neural network $\mathbf{f}_\theta$ that captures inductive biases via the architecture, e.g., convolutional architectures. Additionally, hand-crafted priors such as the Tikhonov regularization or spectral regularization are used to enforce further constraints.

More recent efforts have tried to solve this problem by leveraging independently trained statistical priors using annealed Langevin dynamics sampling (Jayaram and Thickstun, 2020; Lutati et al., 2023; Mariani et al., 2023). However, these methods have demonstrated shortcomings on discrete sources that exhibit intricate temporal structures with equiprobable multimodal distributions (Frank and Ilse, 2020).

5.1.2 Motivation

To address the shortcomings of prior works we are motivated to develop data-driven methods for single-channel source separation that leverage statistical priors for two main reasons—i) **Unknown system parameters**, e.g., the underlying signal generation model may not be available to create hand-crafted priors; and ii) **Automation**, facilitated by learning methods to create plug-and-play priors for versatile use.

We additionally impose restrictions on access to paired training data samples $(\mathbf{x}, \mathbf{n}, \mathbf{y})$ during training. Our motivation for this restriction arises from the following consideration: given n sources, the number of two-component source separation models leveraging joint-statistics grows as $\mathcal{O}(n^2)$. Any changes to the training data—even for a single source—would require $\mathcal{O}(n)$ model updates. In contrast, solutions that leverage independently trained priors over the sources need to update only a single model (i.e., $\mathcal{O}(1)$). Additionally, these priors can also be used to solve more generalized source-separation problems, e.g., with $K > 2$ components or with a different mixture model, without requiring any additional training.

Finally, recognizing the limitations when dealing with sources that exhibit underlying discreteness, we are motivated to develop novel source separation algorithms that can also address this challenge.

5.1.3 Finite Alphabet Signal Processing

Our focus on sources with underlying discrete structures is driven by practical considerations. In many engineering systems, signals commonly exhibit continuous magnitudes. However, in particular cases, these signals may possess discrete properties, leading to a finite (but possibly large) number of possible realizations. Such signals are often expressed as,

$$x(t) = \sum_{p=-\infty}^{\infty} c_p \, g_p(t - t_p), \tag{5.4}$$

where $c_p \in \mathcal{S}$ are discrete symbols drawn from a finite set $\mathcal{S} \subset \mathbb{C}$ (or $\mathbb{R}$) and $g_p(\cdot)$ is a continuous filter that "carries" contributions from the symbols. For example, in optical or RF communications, the symbols could correspond to complex-valued mappings of the underlying bits (Lapidoth, 2017), while in the discrete tomography domain, the symbols might correspond to measurements obtained from different materials with a finite number of phases or absorption values (Gouillart et al., 2013). In this work, we will address the challenges associated with finding the global optimum within the optimization landscape formulated for separating a superposition of such discrete sources.

5.2 Maximum a Posteriori Estimation for Source Separation

Let $\mathbf{x} \in \mathcal{X} \subset \mathbb{C}^D$ and $\mathbf{n} \in \mathbb{C}^D$ be two statistically independent complex-valued vector sources, where $\mathcal{X}$ is a countable set of all realizations $\mathbf{x}$. We assume that $\mathbf{x}$ has PMF P_{x}, and we let $\mathbf{n}$ be an arbitrary source (potentially discrete with some noise) with PDF p_{n}. We assume that the latter distributions are multimodal, where the probability is generally (close to) zero except at the modes. Additionally, we seek to be robust to the challenging setting where the distributions have multiple equiprobable modes, so as to develop novel methods that can tackle the finite alphabet source separation problem. We emphasize that these assumptions do not completely characterize the often complicated fine-grained statistical source structure, and we therefore rely on generative models to learn such unknown structures from data.

5.2.1 The Combinatorial Curse

Given $\mathbf{y}$ and assuming κ is known, in order to separate the sources, it is sufficient to estimate $\mathbf{x}$ since $\mathbf{n} = (\mathbf{y} - \mathbf{x})/\kappa$. The MAP estimate of $\mathbf{x}$ is then given by,

$$\widehat{\mathbf{x}} = \underset{\mathbf{x}\in\mathcal{X} \text{ s.t } \mathbf{y}=\mathbf{x}+\kappa\mathbf{n}}{\arg\max}\; p_{\mathsf{x}|\mathsf{y}}(\mathbf{x}|\mathbf{y}). \tag{5.5}$$

Using Bayes' theorem, Eq. (5.5) can be equivalently expressed as,

$$\begin{aligned}\underset{\mathbf{x}\in\mathcal{X} \text{ s.t } \mathbf{y}=\mathbf{x}+\kappa\mathbf{n}}{\arg\max}\; p_{\mathsf{x}|\mathsf{y}}(\mathbf{x}|\mathbf{y}) &= \underset{\mathbf{x}\in\mathcal{X} \text{ s.t } \mathbf{y}=\mathbf{x}+\kappa\mathbf{n}}{\arg\max}\; p_{\mathsf{y}|\mathsf{x}}(\mathbf{y}|\mathbf{x})P_{\mathsf{x}}(\mathbf{x}) &\text{(5.6a)}\\ &= \underset{\mathbf{x}\in\mathcal{X}}{\arg\min} -\log P_{\mathsf{x}}(\mathbf{x}) - \log p_{\mathsf{n}}\left((\mathbf{y}-\mathbf{x})/\kappa\right), &\text{(5.6b)}\end{aligned}$$

where the likelihood $p_{\mathsf{y}|\mathsf{x}}(\mathbf{y}|\mathbf{x}) = p_{\mathsf{n}}\left((\mathbf{y}-\mathbf{x})/\kappa\right)$ under the constraint Eq. (5.2), and we convert to negative log probabilities in Eq. (5.6b). Due to the underlying discreteness of $\mathbf{x}$, and the potential underlying discreteness of $\mathbf{n}$ as well, the objective function in Eq. (5.6b) is not differentiable. Hence, in its current form, gradient-based techniques cannot be used to solve this problem, and we must instead resort to combinatorial methods, which are computationally infeasible even for moderate dimension size D.

5.3 Proposed Method: α-RGS

To overcome the computational complexity of combinatorial-based methods, our goal is to develop new gradient-based SCSS solutions that leverage diffusion models trained on discrete sources. To this end, we propose to use multiple levels of Gaussian smoothing with an extended MAP framework with an α-posterior, such that the optimization landscape of the resulting objective function is smoothened.

5.3.1 The Smoothing Model and α-posterior Generalized Bayes'

5.3.1.1 Surrogate Distribution

One can almost perfectly approximate P_{x} (in some well-defined sense) with the *surrogate distribution* $p_{\bar{\mathsf{x}}}$, where $\bar{\mathbf{x}} = \mathbf{x} + \boldsymbol{\varepsilon}_x$ for $\boldsymbol{\varepsilon}_x \sim \mathcal{N}(0, \sigma_x^2\mathbf{I})$, $\sigma_x \to 0$. While now theoretically amenable to optimization via gradient descent, the sharp modes, which constitute numerical pitfalls, often cause gradient-based methods to get stuck in local extrema.

5.3.1.2 Gaussian Smoothing Model

To avoid getting trapped in local minima we choose to regularize the underlying landscape further. Inspired by diffusion models we can adopt a variance-exploding smoothing model, with adjustable noise levels (see Section 2.3). Let there be a monotonic noise variance schedule such that $\sigma_r > \sigma_s$ for $0 \leq s < r \leq 1$. Define the "smoothened sources" as

$$\mathbf{x}_t(\bar{\mathbf{x}}) := \bar{\mathbf{x}} + \sigma_t\boldsymbol{\epsilon}_x, \tag{5.7a}$$

$$\mathbf{n}_u(\bar{\mathbf{x}}, \mathbf{y}) := (\mathbf{y} - \bar{\mathbf{x}})/\kappa + \sigma_u\boldsymbol{\epsilon}_n, \tag{5.7b}$$

where $t, u \sim \text{Uni}([0,1])$ and $\boldsymbol{\epsilon}_x, \boldsymbol{\epsilon}_n \sim \mathcal{N}(0, \mathbf{I}_D)$. Observe that $\mathbf{x}_t$ and $\mathbf{n}_u$—the continuous-valued proxies for $\mathbf{x}$ and $\mathbf{n}$, respectively—have PDFs rather than PMFs, and in particular, their PDFs have infinite support, and they are differentiable. More importantly, a direct consequence is that it smoothens the optimization landscape and helps in preventing gradient-based algorithms from getting stuck at spurious local minima.

5.3.1.3 Generalized Bayes' with an α-posterior

We found it useful to replace the likelihood in Eq. (5.6a) with a distribution proportional to $p_{\mathsf{y}|\mathsf{x}}(\mathbf{y}|\mathbf{x})^\alpha$, $\alpha > 1$. Intuitively, under the constraint Eq. (5.2), this sharpens the distribution of $\mathbf{n}$ and gives a higher weight to the modes of p_{n} relative to the natural weighting that arises from the MAP criterion. This is beneficial, for example, when p_{n} is more complicated and has many more modes than P_{x}.

This aforementioned reweighting has been used as an implementation trick in diffusion sampling with classifier conditioning (Dhariwal and Nichol, 2021), but we recognize this as MAP estimation with an α-posterior which is expressed through the generalized Bayes' rule (Grünwald, 2012; Holmes and Walker, 2017; Perrotta, 2020; Zhang, 2003) as,

$$\underbrace{p_{\mathsf{x}|\mathsf{y}}(\mathbf{x}|\mathbf{y};\alpha)}_{\alpha\text{-posterior}} \propto \underbrace{p_{\mathsf{y}|\mathsf{x}}(\mathbf{y}|\mathbf{x})^\alpha}_{\text{tempered likelihood}} \underbrace{P_{\mathsf{x}}(\mathbf{x})}_{\text{prior}}. \tag{5.8}$$

In practice, using an α-posterior has demonstrated increased learning speeds (Holmes and Walker, 2017; Perrotta, 2020) and could potentially help in resolving model mismatch arising from the use of approximate densities or scores (rather than exact ones) during optimization via gradient descent.

5.3.2 Single Noise Level Estimation Loss

Let $\widehat{\mathbf{x}}(\boldsymbol{\theta}) = \boldsymbol{\theta}$ be our estimate of $\mathbf{x}$.[1] Motivated by the form in Eq. (5.6b), we generalize using the smoothing Eq. (5.7a)-Eq. (5.7b) in conjunction with the α-posterior to define a new *single noise level estimation loss*,

$$\mathcal{L}_{t,u}(\boldsymbol{\theta}) := -\log p_{\mathsf{x}_t}\left(\mathbf{x}_t\left(\boldsymbol{\theta}\right)\right) - \alpha \log p_{\mathsf{n}_u}\left(\mathbf{n}_u\left(\boldsymbol{\theta}, \mathbf{y}\right)\right). \tag{5.9}$$

While departing from the original MAP Eq. (5.6b), the newly-defined approximated loss Eq. (5.9) thereof facilitates gradient-based methods, which is key to our solution approach. Particularly, by varying the level of smoothing, Eq. (5.9) is more easily explored in regions between the modes via gradient descent.

5.3.3 Estimation Rule Across Multiple Noise Levels

Intuitively, larger noise variances allow us to move between modes during gradient descent. In contrast, at lower noise levels the modes are sharper, which is beneficial in resolving the

[1] Owing to the continuous nature of the optimization landscape, $\boldsymbol{\theta}$ is in fact an estimate of $\bar{\mathbf{x}}$ (see Section 5.3.1.1), and does not affect BER and MSE estimates significantly.

Algorithm 5.1 Proposed Method: α-posterior with Randomized Gaussian Smoothing (α-RGS)

function SEPARATION($\mathbf{y}$, κ, N, $\{\eta_i\}_{i=0}^{N-1}$, $\boldsymbol{\theta}^{(0)}$) ▷ N total steps, learning rate η_i at step i
  for $i \leftarrow 0, N-1$ **do**
    $t, u \sim \text{Uni}\{1/T, 2/T, \ldots, 1\}$, $\boldsymbol{\epsilon}_x, \boldsymbol{\epsilon}_n \sim \mathcal{N}(0, \mathbf{I})$ ▷ Sample random noise levels
    $\mathbf{x}_t\left(\boldsymbol{\theta}^{(i)}\right) = \boldsymbol{\theta}^{(i)} + \sigma_t \boldsymbol{\epsilon}_x$ ▷ Smooth $\mathbf{x}$ at level t, Eq. (5.7a)
    $\mathbf{n}_u\left(\boldsymbol{\theta}^{(i)}, \mathbf{y}\right) = \left(\mathbf{y} - \boldsymbol{\theta}^{(i)}\right)/\kappa + \sigma_u \boldsymbol{\epsilon}_n$ ▷ Smooth $\mathbf{n}$ at level u, Eq. (5.7b)
    $\widehat{\boldsymbol{\epsilon}}_x, \widehat{\boldsymbol{\epsilon}}_n = \boldsymbol{\epsilon}_{\phi_\mathbf{x}}\left(\mathbf{x}_t\left(\boldsymbol{\theta}^{(i)}\right), t\right), \boldsymbol{\epsilon}_{\phi_\mathbf{n}}\left(\mathbf{n}_u\left(\boldsymbol{\theta}^{(i)}, \mathbf{y}\right), u\right)$ ▷ Compute scores, Eq. (2.8)
    $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} + \eta_i\left[\frac{\alpha}{\kappa\sigma_u}\left(\widehat{\boldsymbol{\epsilon}}_n - \boldsymbol{\epsilon}_n\right) - \frac{1}{\sigma_t}\left(\widehat{\boldsymbol{\epsilon}}_x - \boldsymbol{\epsilon}_x\right)\right]$ ▷ Subtract noise, Eq. (5.10)
  return $\widehat{\mathbf{x}} = \boldsymbol{\theta}^{(N)}$, $\widehat{\mathbf{n}} = (\mathbf{y} - \widehat{\mathbf{x}})/\kappa$

solution, assuming the iterative procedure starts at the basin of attraction of the correct local extremum point, or, alternatively, another "good" local extremum point. We propose a new estimation rule that uses Eq. (5.9) across multiple noise levels. Randomizing over multiple levels of Gaussian smoothing, our proposed gradient update asymptotically (in the number of iterations) takes the form,

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) := \underbrace{-\mathbb{E}_{p(t)q(\boldsymbol{\epsilon}_x)}\left[\mathbf{s}_{\mathsf{x}_t}\left(\mathbf{x}_t\left(\boldsymbol{\theta}\right)\right)\right] + \frac{\alpha}{\kappa}\mathbb{E}_{p(u)q(\boldsymbol{\epsilon}_n)}\left[\mathbf{s}_{\mathsf{n}_u}\left(\mathbf{n}_u\left(\boldsymbol{\theta}, \mathbf{y}\right)\right)\right]}_{\mathbb{E}_{p(t)p(u)}[\nabla_{\boldsymbol{\theta}}\mathcal{L}_{t,u}(\boldsymbol{\theta})]},$$

where $t, u \sim \text{Uni}\left([0, 1]\right)$, and the true score is defined as

$$\mathbf{s}_{\mathsf{x}_t}\left(\mathbf{x}_t\left(\boldsymbol{\theta}\right)\right) := \nabla_{\mathbf{x}} \log p_{\mathsf{x}_t}\left(\mathbf{x}\right)\Big|_{\mathbf{x}=\mathbf{x}_t(\boldsymbol{\theta})},$$

and similarly for $\mathbf{s}_{\mathsf{n}_u}$.

Our proposed updates can be implemented using stochastic gradient descent as shown in Algorithm 5.1. We use pre-trained diffusion models (see Section 2.3) to approximate the score when deriving the analytical score is not possible. Using Tweedie's formula (see Section 2.2.2), we relate the learned score to the denoiser available from the diffusion model, and also use a zero mean noise corrected estimate,

$$\mathbb{E}_{\boldsymbol{\epsilon}_x}\left[\boldsymbol{\epsilon}_{\phi_\mathbf{x}}\left(\mathbf{x}_t\left(\boldsymbol{\theta}\right), t\right)\right] = \mathbb{E}_{\boldsymbol{\epsilon}_x}\left[\boldsymbol{\epsilon}_{\phi_\mathbf{x}}\left(\mathbf{x}_t\left(\boldsymbol{\theta}\right), t\right) - \boldsymbol{\epsilon}_x\right], \tag{5.10}$$

in our updates as shown in Algorithm 5.1. The above rule is motivated to introduce numerical stability and reduces the variance of the updates, since the diffusion models were trained to minimize the squared value of the same error term as in Eq. (2.12). Furthermore, rather than sampling timesteps in the continuous range we discretize the unit interval and sample $t, u \sim \text{Uni}(\{i/T\}_{i=1}^{T})$ for some integer $T > 0$.

5.4 Characterization of α-RGS

In this section, we characterize the behavior of our objective function through a simple yet intuitive example. We first present a sufficient condition for perfect signal separation in the

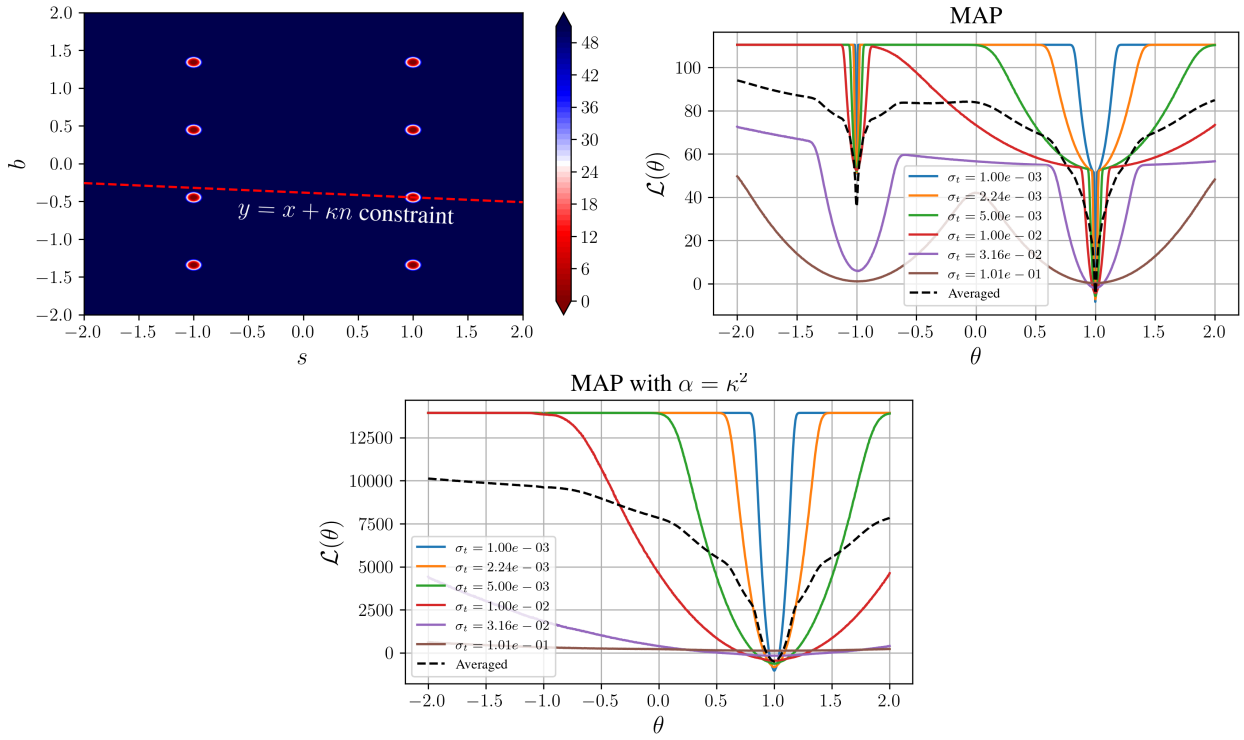


Figure 5.1.: **Top left:** Two discrete sources, with infinitesimal additive noise, superimposed to produce a joint distribution with 8 equiprobable modes. An observed mixture $\mathbf{y}$, imposes a linear constraint in this space. **Top right:** Extending vanilla MAP ($\alpha = 1$) to multiple noise levels still has a relatively large local minima. **Bottom:** By using $\alpha = \kappa^2$, we are able to accentuate the correct mode and smooth the landscape even further. Colored curves correspond to Eq. (5.12) evaluated with $T = 1$ and $t = u$.

context of discrete sources that follow the signal generation model in Eq. (5.4).

Proposition 5.1. *Let $x(t)$ and $n(t)$ be two sources following Eq. (5.4) with underlying symbols c_p^x and c_p^n respectively. Assume that the symbols are obtained as,*

$$c_p^x = f\left(\{u_i\}_{i=p}^{L+p}\right) \quad \text{and} \quad c_p^n = h\left(\{v_i\}_{i=p}^{L+p}\right), \tag{5.11}$$

where $f : \mathcal{U}^L \to \mathbb{C}$ and $h : \mathcal{V}^L \to \mathbb{C}$ are mappings from a sequence of length L over the discrete alphabets $\mathcal{U}$ and $\mathcal{V}$ respectively. If the mapping between the discrete representation and the symbol representation of the sources is unique, perfect recovery is possible.

5.4.1 Convergence Analysis

We now argue that under such conditions, our method in Algorithm 5.1 asymptotically approaches a local extremum corresponding to the following loss function,

$$\mathcal{L}(\boldsymbol{\theta}) := -\mathbb{E}_{p(t)q(\boldsymbol{\epsilon}_x)}\left[\log p_{\mathbf{x}_t}\left(\mathbf{x}_t\left(\boldsymbol{\theta}\right)\right)\right] - \alpha\mathbb{E}_{p(u)q(\boldsymbol{\epsilon}_n)}\left[\log p_{\mathbf{n}_u}\left(\mathbf{n}_u\left(\boldsymbol{\theta}, \mathbf{y}\right)\right)\right]. \tag{5.12}$$

To see this, let x and n be two discrete sources with equiprobable modes at $\{-1, +1\}$ and $\{\pm 6/\sqrt{20}, \pm 2/\sqrt{20}\}$ respectively. Figure 5.1 shows the contours of the negative log joint probability, with the sources augmented with an infinitesimally small amount of Gaussian noise (see Section 5.3.1). An observed mixture y, adds a linear constraint in the (x, n) plane. The goal of Algorithm 5.1 is to pick out the constraint-satisfying mode at $(1, -2/\sqrt{20})$. If $\alpha = 1$, as shown in the middle plot, and given a poor initialization $\boldsymbol{\theta}^{(0)}$ closer to -1, the gradients in this region may prevent the estimate from escaping the suboptimal local minimum. If instead, we use $\alpha = \kappa^2 > 1$ ($\kappa = 15.85$), as shown in rightmost plot, the optimization landscape is better conditioned in the same (absolute) neighborhood around $\boldsymbol{\theta} = -1$ with the mode at $+1$ much more accentuated. Specifically, if the algorithm is now initialized at $\boldsymbol{\theta}^{(0)} = -1$, after enough iterations, gradient steps at larger noise levels would lead the solution towards the direction of $\boldsymbol{\theta} = 1$. Thus, on average (black curve), after enough iterations, the solution will approach $+1$. In constrast, Langevin-dynamics-based approaches without the α-posterior weighting (Jayaram and Thickstun, 2020), if initialized poorly, could potentially get stuck at local extrema due to sharpening of the optimizing landscape as the level of noise decreases.

The estimate of the discrete source $\mathbf{x}$ can be obtained by mapping the solution returned by Algorithm 5.1 to the closest point (in the Euclidean sense) in the discrete alphabet $\mathcal{X}$. This relies on the extremum $\boldsymbol{\theta}^*$ of Eq. (5.12) being sufficiently close to the desired mode of $p_{\mathsf{x}|\mathsf{y}}(\mathbf{x}|\mathbf{y}; \alpha)$, so that $\boldsymbol{\theta}^*$ can be mapped to the correct point in $\mathcal{X}$ with high probability. In the context of the above example, a key observation in achieving this desired behavior is that the mode at $(1, -2/\sqrt{20})$ is still prominent at large noise levels. Thus, randomizing across different noise levels helps balance the exploration between modes and the resolution of the estimate. This mainly requires a suitable noise level range (i.e., a lower bound on σ_0), to ensure that the modes are sufficiently resolved. In our experiments, we show that no additional tuning is required and that the training noise levels from the pre-trained diffusion models can be re-used, provided that the models have learned the source's structure sufficiently well.

5.4.2 Mode Seeking Behavior

We now focus on a single term,

$$\mathcal{L}_{\mathbf{x}}(\boldsymbol{\theta}) := -\mathbb{E}_{p(t)q(\boldsymbol{\epsilon}_x)} \left[\log p_{\mathsf{x}_t} \left(\mathbf{x}_t \left(\boldsymbol{\theta}\right)\right)\right]. \tag{5.13}$$

where we recall the definition of the Gaussian smoothing model from Section 5.3.1,

$$\mathbf{x}_t \left(\boldsymbol{\theta}\right) = \bar{\boldsymbol{\theta}} + \sigma_t \boldsymbol{\epsilon}_x, \quad \boldsymbol{\epsilon}_x \sim \mathcal{N}(0, \mathbf{I}_D). \tag{5.14}$$

As detailed in Section 5.3.1, when estimating a discrete source $\mathbf{x}$, $\bar{\boldsymbol{\theta}} = \boldsymbol{\theta} + \boldsymbol{\varepsilon}_x$ for $\boldsymbol{\varepsilon}_x \sim \mathcal{N}(0, \sigma_x^2 \mathbf{I}_D)$, $\sigma_x \to 0$, is a continuous surrogate of the discrete estimate $\boldsymbol{\theta}$, useful for optimization via gradient descent. The above loss only depends on the prior of the source, and is in particular independent of the data. Therefore, this term serves (and can be viewed) as a regularizer in our inference optimization problem. Although analysis of Eq. (5.13) is (strictly speaking) not useful in order to make statements about Eq. (5.12), it is nevertheless informative and insightful to show that the local extrema of Eq. (5.13) approach the modes of the underlying source distribution P_{x}, by solving for the stationary points,

$$\mathbb{E}_{p(t)q(\boldsymbol{\epsilon}_x)} \left[\nabla_{\boldsymbol{\theta}} \log p_{\mathsf{x}_t}(\boldsymbol{\theta} + \sigma_t \boldsymbol{\epsilon}_x)\right] = 0 \tag{5.15}$$

where we have used Leibniz rule for differentiation under the integral. Through our examples we will demonstrate the mode-approaching nature of the solutions to Eq. (5.15) thus establishing another interpretation of the asymptotic behavior of our method—two loss terms whose stationary points are modes of the corresponding source distributions which work together in unison to satisfy the constraints imposed by the observed mixture. We thereby end up, with high probability, in the correct mass-points of the underlying distributions.

5.4.3 Multivariate Normal Sources

We first start with the analysis of a multivariate normal source and show that the the *exact* mode is obtained as the local extremum of Eq. (5.13).

Proposition 5.2. *Let $\mathbf{x} \in \mathbb{R}^D$ be a multivariate normal source with mean $\boldsymbol{\mu}_{\mathbf{x}}$ and covariance matrix $\boldsymbol{\Sigma}_{\mathbf{x}}$. For the Gaussian smoothing model in Eq. (5.14), the score at timestep t is*

$$\nabla_{\mathbf{x}_t(\mathbf{x})} \log p_{\mathbf{x}_t}\left(\mathbf{x}_t\left(\mathbf{x}\right)\right) = -\boldsymbol{\Sigma}_{\mathbf{x}_t}^{-1}\left(\mathbf{x} + \sigma_t \boldsymbol{\epsilon}_x - \boldsymbol{\mu}_{\mathbf{x}_t}\right), \tag{5.16}$$

$$\boldsymbol{\mu}_{\mathbf{x}_t} := \boldsymbol{\mu}_{\mathbf{x}} \quad \textit{and} \quad \boldsymbol{\Sigma}_{\mathbf{x}_t} := \boldsymbol{\Sigma}_{\mathbf{x}} + \sigma_t^2 \mathbf{I}_D.$$

Then, the minimizer $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{\mathbf{x}}(\boldsymbol{\theta})$ is equal to $\boldsymbol{\mu}_{\mathbf{x}}$, i.e., the mode of the source distribution.

Proof. Since $\mathbf{x}$ is a continuous source, we perform optimization over a continuous space with respect to $\boldsymbol{\theta} \in \mathbb{R}^d$.[2] Notice that,

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \log p_{\mathbf{x}_t}(\boldsymbol{\theta} + \sigma_t \boldsymbol{\epsilon}_x) &= \nabla_{\mathbf{x}_t(\boldsymbol{\theta})} \log p_{\mathbf{x}_t}\left(\mathbf{x}_t\left(\boldsymbol{\theta}\right)\right) && \text{(5.17a)} \\ &= \boldsymbol{\Sigma}_{\mathbf{x}_t}^{-1}\left(\boldsymbol{\theta} + \sigma_t \boldsymbol{\epsilon}_x - \boldsymbol{\mu}_{\mathbf{x}_t}\right) && \text{(5.17b)} \end{aligned}$$

Substituting Eq. (5.17b) in Eq. (5.15), we see that the minimizer must satisfy

$$\mathbb{E}_{p(t)}\left[\boldsymbol{\Sigma}_{\mathbf{x}_t}^{-1}\left(\boldsymbol{\theta} - \boldsymbol{\mu}_{\mathbf{x}_t}\right)\right] = 0,$$

where we have used the fact that $\boldsymbol{\epsilon}_x$ is a zero mean Gaussian realization. Since $\boldsymbol{\Sigma}$ is invertible the minimizer is

$$\boldsymbol{\theta}^* = \boldsymbol{\mu}_{\mathbf{x}_t} = \boldsymbol{\mu}_{\mathbf{x}}.$$

□

Thus, we have established that the local extremum of the Eq. (5.13) corresponds to the mode in the multivariate Gaussian case.

5.4.4 Finite Alphabet Sources

We next analyze Eq. (5.13) in the context of finite alphabet sources with symbols drawn from $\mathcal{A}$ as described in Section 5.1.3. We will once again leverage Tweedie's formula (see Section 2.2.2) and for completeness of the exposition, we prove it for the case of scalar random variables below.

[2] In this continuous setting, a surrogate distribution is not needed and $\bar{\boldsymbol{\theta}} = \boldsymbol{\theta}$ in Eq. (5.14)

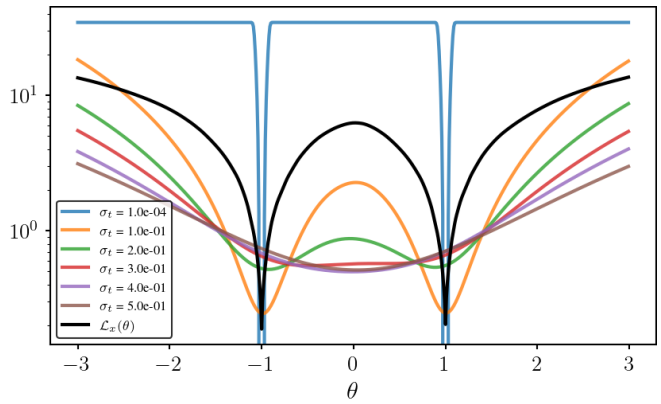


Figure 5.2.: Simulating Eq. (5.13) on a two alphabet source. The loss at individual timestamps is visualized in addition to the total loss. The minima are at the modes of the source distribution, -1 and $+1$. Larger noise levels allow for exploration between modes and smaller noise levels sharpen the mode-seeking behavior.

Proposition 5.3. *Let x be a scalar random variable representing a symbol drawn uniformly from a finite alphabet set $\mathcal{A} \subset \mathbb{C}$. For the Gaussian smoothing model in Eq. (5.14) where, at timestep t,*

$$x_t(x) = x + \sigma_t \epsilon_x,$$

the score is

$$\nabla_{x_t(x)} \log p_{\mathsf{x}_t}(x_t(x)) = \frac{1}{\sigma_t^2}\left(-x_t(x) + \sum_{a \in \mathcal{A}} a \cdot \phi_t(a, s_t(x))\right), \tag{5.18}$$

where,

$$\phi_t(a, x_t(x)) := \frac{\exp\{-\frac{|x_t(x)-a|^2}{\sigma_t^2}\}}{\sum_{\bar{a}\in\mathcal{A}} \exp\{-\frac{|x_t(x)-\bar{a}|^2}{\sigma_t^2}\}}. \tag{5.19}$$

As an illustrative example, consider a double alphabet source with modes at -1 and $+1$[3]. To solve Eq. (5.15) for $\boldsymbol{\theta}$, we require the score of the source at different non-zero levels of Gaussian smoothing. Note that since the source is discrete, the score is undefined if no smoothing is applied. Though this smoothened score can be computed using Eq. (5.18), the stationary points of Eq. (5.15) cannot be computed in closed-form. Hence, in order to study the behavior of the solutions, we simulate Eq. (5.13), as shown in Figure 5.2. The colored curves plot the loss for a single noise level, i.e., when $T = 1$, while the black curve computes the loss across multiple levels of smoothing, by computing an average over the single noise level curves. It is evident from these curves, that while the minima might not lie at the modes -1 and $+1$ for the single noise level curves, by averaging across multiple noise levels, the solution to Eq. (5.15) approach the mode of the source distribution. As pointed out in Section 5.4, an important caveat is that this behavior is only observed for a good choice of the noise level range. Larger noise levels aid in moving between the modes and smaller noise levels help resolving the solution. Only using larger noise levels, for example, can smoothen

[3]In practical engineering settings this is representative of a BPSK source, for example

out the landscape to the point that the modes are no longer discernible, thus resulting in erroneous estimates. We leverage the noise levels from pre-trained diffusion models that have learned the statistical structure sufficiently well, thus avoiding cumbersome tuning of the noise level range. Additional analysis for Gaussian mixture models is in Appendix 5.B.

5.5 Related Work

The proposed optimization framework is closely related to recent methods that use score estimators to enhance sampling quality and guide iterative samplers, as outlined below.

5.5.1 BASIS Separation

Jayaram and Thickstun (2020) introduced the BASIS separation algorithm that leverages the score from a generative model to perform source separation using annealed Langevin dynamics. Unlike our method, the BASIS algorithm relies on a specially tuned noise schedule for separation that is distinct from the diffusion model training noise schedule. We circumvent the challenges associated with tuning such a schedule and instead re-use the pre-determined training noise schedule in a randomized fashion. As such, the only parameter that we tune is the learning rate. A comparison between our method and BASIS can be found in Appendix 5.C.

5.5.2 Score Distillation Sampling

The proposed method can be viewed as an extension of the recently proposed Dreamfusion architecture (Poole et al., 2022), which uses pre-trained image diffusion models as critics to guide the generation of novel 3D visual content. Given a generated sample, $g(\boldsymbol{\theta})$, Score Distillation Sampling (SDS) updates the 3D object realization using gradient descent with a gradient given by,

$$\nabla_\theta \mathcal{L}_{\text{SDS}}\left(\phi, \mathbf{x} = g(\boldsymbol{\theta})\right) = \mathbb{E}_{p(t)q(\boldsymbol{\epsilon}_x)}\left[w(t)(\boldsymbol{\epsilon}_\phi(g(\boldsymbol{\theta}) + \sigma_t \boldsymbol{\epsilon}_x, t) - \boldsymbol{\epsilon}_x)\right], \tag{5.20}$$

where $w(t)$ is a scaling related to the noise variance. Our updates contain the same gradient terms in Eq. (5.20) and hence it can be viewed as a multi-source extension of SDS, shedding light on its use in applications to problems beyond sampling with interactions between numerous individual priors.

5.6 Summary

In this chapter we introduced α-RGS, a method that extends MAP estimation with an α-posterior across randomized levels of Gaussian smoothing, which stems from a new objective function, whose extrema points correspond to the modes of the underlying discrete distribution of interest. Our method relies only on pre-trained diffusion models as priors via a simple randomized algorithm that does not require cumbersome tuning of a special annealing schedule, as is done in existing Langevin-dynamics-based works. Through simple analytical illustrations, we demonstrate the favorable mode-preserving nature of our objective.

5.7 Extension to General Inverse Problems*

While this chapter focused on a specific instance of an inverse problem, the proposed framework, based on MAP estimation, can be extended to address a broader class of inverse problems. In the following sections, we first describe how to generalize our algorithm for recovering a signal of interest under a more flexible likelihood formulation, and then present a modified version of the algorithm suited for practical implementation.

5.7.1 Posterior Sampling

Recently, diffusion priors have emerged as a powerful tool for solving a wide range of inverse problems, including MRI reconstruction (Chung and Ye, 2022; Song et al., 2022), image restoration and colorization (Chung et al., 2022a,b; Kadkhodaie and Simoncelli, 2021; Kawar et al., 2022; Song et al., 2021c), and more broadly, for developing general-purpose inversion algorithms (Chung et al., 2023, 2022a; Rout et al., 2023; Song et al., 2023a). Most of these works focus on linear inverse problems, similar to the one introduced in Section 1.1.1, where the forward model is given by:

$$\mathbf{y} = \mathsf{A}\mathbf{x} + \mathbf{n},$$

and the goal is to recover the signal-of-interest (SOI) by sampling from the posterior distribution $p(\mathbf{x}|\mathbf{y})$.

To achieve this, these methods typically employ annealed Langevin dynamics or reverse diffusion processes that progressively reduce noise to approximate posterior samples. The noise $\mathbf{n}$ is often modeled as white Gaussian, which makes the likelihood model $p(\mathbf{y}|\mathbf{x})$ tractable and known in closed form. For example, diffusion posterior sampling (DPS) (Chung et al., 2022a) uses an annealed Langevin update rule, where the score of the posterior is decomposed via Bayes' rule. At each noise level t the update rule is,

$$\mathbf{x}_t^{(i+1)} \leftarrow \mathbf{x}_t^{(i)} + \nabla_{\mathbf{x}_t^{(i)}} \log p_{\mathsf{x}_t}(\mathbf{x}_t^{(i)}) + \nabla_{\mathbf{x}_t^{(i)}} \log p_{\mathsf{y}|\mathsf{x}_t}(\mathbf{y}|\mathbf{x}_t^{(i)}) + 2\eta\sqrt{\boldsymbol{\epsilon}_t}. \tag{5.21}$$

The score can be obtained using a pre-trained diffusion model similar to α-RGS, but the score of the likelihood conditioned on the noisy SOI is not available in closed form. A common approximation made is,

$$\nabla_{\mathbf{x}_t^{(i)}} \log p_{\mathsf{y}|\mathsf{x}_t}(\mathbf{y}|\mathbf{x}_t^{(i)}) \approx \nabla_{\mathbf{x}_t^{(i)}} \log p_{\mathsf{y}|\mathsf{x}}(\mathbf{y}|\mathbb{E}[\mathbf{x}|\mathbf{x}_t^{(i)}]), \tag{5.22}$$

where the conditional expectation $\mathbb{E}[\mathbf{x}|\mathbf{x}_t^{(i)}]$ is estimated using a pre-trained diffusion model by leveraging Tweedie's formula. When multiple plausible solutions to an inverse problem are desired, posterior sampling methods provide a compelling framework by enabling the generation of diverse samples from the posterior distribution. However, a key limitation of these approaches is that they do not naturally yield a way to compute the MAP estimate of the SOI. In the following section, we generalize the α-RGS framework for MAP estimation within this setting.

Algorithm 5.2 Proposed Method:α-RGS for Linear Inverse Problems

function SEPARATION($\mathbf{y}$, A, N, $\{\eta_i\}_{i=0}^{N-1}$, $\boldsymbol{\theta}^{(0)}$) ▷ N total steps, learning rate η_i at step i
  for $i \leftarrow 0, N-1$ **do**
    $t, u \sim \mathrm{Uni}\{1/T, 2/T, \ldots, 1\}$, $\boldsymbol{\epsilon}_x, \boldsymbol{\epsilon}_n \sim \mathcal{N}(0, \mathbf{I})$ ▷ Sample random noise levels
    $\mathbf{x}_t\left(\boldsymbol{\theta}^{(i)}\right) = \boldsymbol{\theta}^{(i)} + \sigma_t \boldsymbol{\epsilon}_x$ ▷ Smooth $\mathbf{x}$ at level t and u
    $\mathbf{x}_u\left(\boldsymbol{\theta}^{(i)}\right) = \boldsymbol{\theta}^{(i)} + \sigma_u \boldsymbol{\epsilon}_x$
    $\widehat{\boldsymbol{\epsilon}}_{x_t}, \widehat{\boldsymbol{\epsilon}}_{x_u} = \boldsymbol{\epsilon}_{\phi_{\mathbf{x}}}\left(\mathbf{x}_t\left(\boldsymbol{\theta}^{(i)}\right), t\right), \boldsymbol{\epsilon}_{\phi_{\mathbf{x}}}\left(\mathbf{x}_u\left(\boldsymbol{\theta}^{(i)}\right), u\right)$ ▷ Compute scores, Eq. (2.8)
    $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \eta_i \left[\alpha \|\mathbf{y} - \mathsf{A}(\mathbf{x}_u(\boldsymbol{\theta}) - \sigma_u \widehat{\boldsymbol{\epsilon}}_{x_u})\|_2^2 + \frac{1}{\sigma_t}\left(\widehat{\boldsymbol{\epsilon}}_x - \boldsymbol{\epsilon}_x\right)\right]$
  return $\widehat{\mathbf{x}} = \boldsymbol{\theta}^{(N)}$

5.7.2 Generalizing α-RGS

Using Bayes' rule, the MAP estimation problem with an α-posterior takes the form,

$$\arg\min_{\mathbf{x}} -\log p_{\mathsf{x}}(\mathbf{x}) - \alpha \log p_{\mathsf{y}|\mathsf{x}}(\mathbf{y}|\mathbf{x}).$$

Unlike the SCSS setting, this formulation does not impose a hard mixture constraint. Instead, we assume access to a valid likelihood model, and the underlying signals are not restricted to possess discrete or combinatorial structure.

Again, let $\boldsymbol{\theta}$ be the estimate of the SOI. By leveraging randomized levels of Gaussian smoothing for the SOI across multiple different noise levels, we can define a MAP-like objective to solve this problem,

$$\mathcal{L}_{\text{inv}}(\boldsymbol{\theta}) := -\mathbb{E}_{p(t)q(\boldsymbol{\epsilon}_x)}[\log p_{\mathsf{x}_t}(\mathbf{x}_t(\boldsymbol{\theta}))] - \alpha \mathbb{E}_{p(u)q(\boldsymbol{\epsilon}_x)}[\log p_{\mathsf{y}|\mathsf{x}_u}(\mathbf{x}_u(\boldsymbol{\theta}))]. \tag{5.23}$$

The gradient of this objective can be approximated as,

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{inv}}(\boldsymbol{\theta}) \approx -\mathbb{E}_{p(t)q(\boldsymbol{\epsilon}_x)}[\mathbf{s}_{\mathsf{x}_t}(\mathbf{x}_t(\boldsymbol{\theta}))] - \alpha \mathbb{E}_{p(u)q(\boldsymbol{\epsilon}_x)}[\nabla_{\boldsymbol{\theta}} \log p_{\mathsf{y}|\mathsf{x}}(\mathbf{y}|\mathbb{E}[\mathbf{x}|\mathbf{x}_u(\boldsymbol{\theta})])], \tag{5.24}$$

where the first term involves the marginal score of the noisy signal, and the second term uses the DPS approximation in Eq. (5.22). Contrary to the SCSS setting the gradient of the likelihood requires backpropagating through the score model in practice, which could potentially be expensive in practice. Specializing to linear inverse problems with AWGN noise, the gradient update is,

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{inv}}(\boldsymbol{\theta}) \approx -\mathbb{E}_{p(t)q(\boldsymbol{\epsilon}_x)}[\mathbf{s}_{\mathsf{x}_t}(\mathbf{x}_t(\boldsymbol{\theta}))] + \alpha \mathbb{E}_{p(u)q(\boldsymbol{\epsilon}_x)}[\|\mathbf{y} - \mathsf{A}\mathbb{E}[\mathbf{x}|\mathbf{x}_u(\boldsymbol{\theta})]\|_2^2].$$

The overall algorithm is shown in Algorithm 5.2.

Appendix

5.A Deferred Proofs

Proof of Proposition 5.3. We will use Eq. (2.8) to compute the score, for which the only quantity we need to compute is the conditional expectation. Let $\mathsf{x}' := \mathsf{x}$. Since x is discrete, $P_{\mathsf{x}'}$ is also a uniform distribution over the symbol set $\mathcal{A}_t := \{a \,|\, a \in \mathcal{A}\}$. By Bayes' rule,

$$\begin{aligned}
\mathbb{P}\left[\mathsf{x}' = a | \mathsf{x}' + \sigma_t \epsilon_x = x\right] &= \frac{\mathbb{P}\left[\mathsf{x}' + \sigma_t \epsilon_x = x | \mathsf{x}' = a\right] \mathbb{P}\left[\mathsf{x}' = a\right]}{\sum_{\bar{a} \in \mathcal{A}} \mathbb{P}\left[\mathsf{x}' + \sigma_t \epsilon_x = x | \mathsf{x}' = \bar{a}\right] \mathbb{P}\left[\mathsf{x}' = \bar{a}\right]} \\
&= \frac{\mathbb{P}\left[\epsilon_x = \frac{x-a}{\sigma_t}\right] \cdot \frac{1}{|\mathcal{A}|}}{\sum_{\bar{a} \in \mathcal{A}} \mathbb{P}\left[\epsilon_x = \frac{x-\bar{a}}{\sigma_t}\right] \cdot \frac{1}{|\mathcal{A}|}} \\
&= \frac{\exp\{-\frac{|x-a|^2}{\sigma_t^2}\}}{\sum_{\bar{a} \in \mathcal{A}} \exp\{-\frac{|x-\bar{a}|^2}{\sigma_t^2}\}} \\
&= \phi_t(a, x).
\end{aligned}$$

Therefore, from Eq. (2.8),

$$\nabla_{x_t(x)} \log p_{\mathsf{x}_t}\left(x_t\left(x\right)\right) = \frac{1}{\sigma_t^2}\left(-x_t\left(x\right) + \sum_{a \in \mathcal{A}} a \cdot \phi_t\left(a, x_t\left(x\right)\right)\right)$$

□

5.B Gaussian Mixture Source Model

We also consider a scalar Gaussian mixture model (GMM) that can often model arbitrary complicated scalar distributions. We derive the score for a 2-component GMM and then generalize the result to an arbitrary K-component GMM.

Proposition 5.4. *Consider a two-component scalar Gaussian mixture source,*

$$p_{\mathsf{x}}(x) = \lambda \mathcal{N}(x; \mu_1, \sigma_1^2) + (1-\lambda)\mathcal{N}(x; \mu_2, \sigma_2^2), \quad \lambda \in (0,1).$$

Under the Gaussian smoothing model in Eq. (5.14),

$$\begin{aligned}
\nabla_{x_t(x)} \log p_{\mathsf{x}_t}(x_t(x)) = \frac{1}{Z(x_t(x))}\Bigg[&\lambda\, c_1(x_t(x))\left(\frac{\mu_1 - x_t(x)}{\sigma_1^2 + \sigma_t^2}\right) \\
&+ (1-\lambda)\, c_2(x_t(x))\left(\frac{\mu_2 - x_t(x)}{\sigma_2^2 + \sigma_t^2}\right)\Bigg],
\end{aligned}$$

where

$$
\begin{aligned}
c_1(x_t(x)) &:= \mathcal{N}(x_t(x); \mu_1, \sigma_1^2 + \sigma_t^2), \\
c_2(x_t(x)) &:= \mathcal{N}(x_t(x); \mu_2, \sigma_2^2 + \sigma_t^2), \\
Z(x_t(x)) &:= \lambda\, c_1(x_t(x)) + (1-\lambda)\, c_2(x_t(x)).
\end{aligned}
$$

Proof. We need to compute the score of the following distribution,

$$
p_{\mathsf{x}_t}(x_t(x)) = p_{\mathsf{x}+\sigma_t\epsilon}(\mathsf{x} + \sigma_t\epsilon_x = x_t(x)), \tag{5.25}
$$

Our goal, to this end, is to leverage Tweedie's formula to compute the score, thereby requiring an expression for the conditional expectation . Since x and ϵ_x are independent, Eq. (5.25) can be written as a convolution between two distributions,

$$
p_{\mathsf{x}_t}(x_t(x)) = (p_{\mathsf{x}'} * p_{\epsilon_{\sigma_t}})(\mathsf{x}' + \epsilon_{\sigma_t} = x_t(x)),
$$

where

$$
\mathsf{x}' := \mathsf{x} \quad \text{and} \quad \epsilon_{\sigma_t} := \sigma_t\epsilon_x. \tag{5.26}
$$

In order to compute the conditional expectation, we first show that,

$$
\begin{aligned}
&p_{\mathsf{x}'+\epsilon_{\sigma_t}|\mathsf{x}'}(\mathsf{x}' + \epsilon_{\sigma_t} = x_t(x) | \mathsf{x}' = a') p_{\mathsf{x}'}(\mathsf{x}' = a') \\
&= \frac{1}{Z(x_t(x))} \left[\lambda\, c_1(x_t(x))\, \mathcal{N}\left(a; \frac{\sigma_1^2 x_t(x) + \sigma_t^2\mu_1}{\sigma_1^2 + \sigma_t^2}, \frac{\sigma_t^2\sigma_1^2}{\sigma_1^2 + \sigma_t^2}\right) \right. \\
&\qquad \left. + (1-\lambda)\, c_2(x_t(x))\, \mathcal{N}\left(a; \frac{\sigma_2^2 x_t(x) + \sigma_t^2\mu_2}{\sigma_2^2 + \sigma_t^2}, \frac{\sigma_t^2\sigma_2^2}{\sigma_2^2 + \sigma_t^2}\right) \right],
\end{aligned} \tag{5.27}
$$

where

$$
\begin{aligned}
c_1(x_t(x)) &:= \mathcal{N}(x_t(x); \mu_1, \sigma_1^2 + \sigma_t^2), \\
c_2(x_t(x)) &:= \mathcal{N}(x_t(x); \mu_2, \sigma_2^2 + \sigma_t^2), \\
Z(x_t(x)) &:= \lambda\, c_1(x_t(x)) + (1-\lambda)\, c_2(x_t(x)).
\end{aligned}
$$

We start by deriving the density functions for the random variables in Eq. (5.26). Since the convolution between two Gaussians is a Gaussian,

$$
\begin{aligned}
p_{\mathsf{x}'}(a) &= \lambda\mathcal{N}(a; \mu_1, \sigma_1^2) + (1-\lambda)\mathcal{N}(a; \mu_2, \sigma_2^2), \\
p_{\epsilon_{\sigma_t}}(w) &= \mathcal{N}(w; 0, \sigma_t^2),
\end{aligned} \tag{5.28}
$$

from which it follows, for example through the identities derived in (Bromiley, 2003, Sec 1)), that

$$
p_{\mathsf{x}_t}(x_t(x)) = \lambda\mathcal{N}(x_t(x); \mu_1, \sigma_1^2 + \sigma_t^2) + (1-\lambda)\mathcal{N}(x_t(x); \mu_2, \sigma_2^2 + \sigma_t^2).
$$

We conclude the derivation of the conditional distribution by using Bayes' rule,

$$
p_{\mathsf{x}'|\mathsf{x}'+\epsilon_{\sigma_t}}(\mathsf{x}' = a | \mathsf{x}' + \epsilon_{\sigma_t} = x_t(x)) = \frac{p_{\mathsf{x}'+\epsilon_{\sigma_t}|\mathsf{x}'}(\mathsf{x}' + \epsilon_{\sigma_t} = x_t(x) | \mathsf{x}' = a) p_{\mathsf{x}'}(\mathsf{x}' = a)}{\int p_{\mathsf{x}'+\epsilon_{\sigma_t}|\mathsf{x}'}(\mathsf{x}' + \epsilon_{\sigma_t} = x_t(x) | \mathsf{x}' = a') p_{\mathsf{x}'}(\mathsf{x}' = a')\, \mathrm{d}a'}.
$$

whereby simplifying the terms in the numerator further, we have

$$p_{\mathsf{x}'+\epsilon_{\sigma_t}|\mathsf{x}'}(\mathsf{x}'+\epsilon_{\sigma_t}=x_t(x)|\mathsf{x}'=a)=p_{\epsilon_{\sigma_t}}(\epsilon_{\sigma_t}=x_t(x)-a)=\mathcal{N}(a;x_t(x),\sigma_t^2). \tag{5.29}$$

Finally, by following the Gaussian product identities in (Bromiley, 2003, Sec 1), between the distribution in Eq. (5.29) and Eq. (5.28) we reach the result in Eq. (5.27).

Now we can compute the expectation as,

$$\mathbb{E}\left[\mathsf{x}'|\mathsf{x}'+\epsilon_{\sigma_t}=x_t(x)\right]=\frac{1}{Z(x_t(x))}\left[\lambda\, c_1(x_t(x))\,\left(\frac{\sigma_1^2 x_t(x)+\sigma_t^2\mu_1}{\sigma_1^2+\sigma_t^2}\right)\right.$$
$$\left.+\,(1-\lambda)\, c_2(x_t(x))\,\left(\frac{\sigma_2^2 x_t(x)+\sigma_t^2\mu_2}{\sigma_2^2+\sigma_t^2}\right)\right].$$

Using Eq. 2.8 we can express the score as,

$$\nabla_{x_t(x)}\log p_{\mathsf{x}_t}(x_t(x))=\frac{1}{Z(x_t(x))}\left[\lambda\, c_1(x_t(x))\,\left(\frac{\mu_1-x_t(x)}{\sigma_1^2+\sigma_t^2}\right)\right.$$
$$\left.+\,(1-\lambda)\, c_2(x_t(x))\,\left(\frac{\mu_2-x_t(x)}{\sigma_2^2+\sigma_t^2}\right)\right].$$

□

We can now generalize the score to an arbitrary K-component GMM ..

Proposition 5.5. *Consider a K-component Gaussian mixture source,*

$$p_{\mathsf{x}}(x)=\sum_{i=1}^{K}\lambda_i\,\mathcal{N}(x;\mu_i,\sigma_i^2),\quad \sum_{i=1}^{K}\lambda_i=1.$$

For the Gaussian smoothing model in Eq. (5.14), the score at timestep t is,

$$\nabla_{x_t(x)}\log p_{\mathsf{x}_t}(x_t(x))=\frac{1}{Z(x_t(x))}\sum_{i=1}^{K}\lambda_i\, c_i(x_t(x))\,\left(\frac{\mu_i-x_t(x)}{\sigma_i^2+\sigma_t^2}\right),$$

where

$$c_i(x_t(x)):=\mathcal{N}(x_t(x);\mu_i,\sigma_i^2+\sigma_t^2),$$
$$Z(x_t(x)):=\sum_{i=1}^{K}\lambda_i\, c_i(x_t(x)).$$

Similar to the finite alphabet case, since a closed-form expression for the solution to Eq. (5.15) cannot be obtained, we simulate Eq. (5.13) for a four-component GMM with two large equiprobable modes and two smaller modes. As shown in the colored curves on the right in Figure 5.3, we again notice similar behavior where the modes are sharper at lower noise levels, with the sharpness decreasing at large noise levels. Hence, gradient-based methods can leverage the randomly chosen larger noise levels to move between modes and use the smaller noise levels to resolve the estimate. On average, the minima of the loss approach the modes of the original GMM source, as shown by the black curve on the right in Figure 5.3.

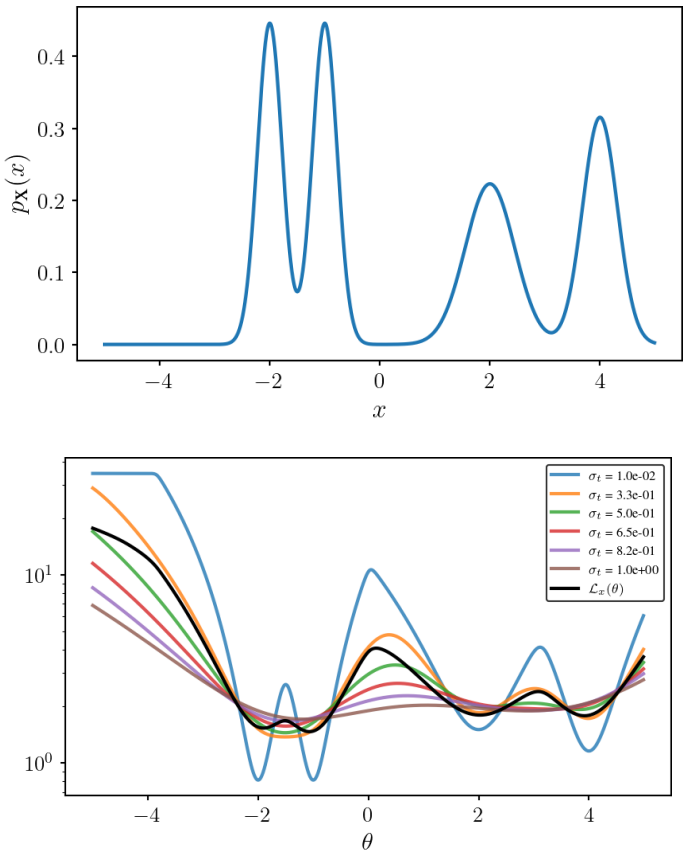


Figure 5.3.: **Top:** A GMM source with two equiprobable modes at -1 and -2. Two smaller modes are present at $+2$ and $+4$. **Bottom:** The minima of Eq. (5.13) lie (approximately) at the modes of the source distribution (black curve). Colored curves correspond to Eq. (5.13) evaluated with $T = 1$.

Algorithm 5.2 BASIS Separation (original, as proposed in (Jayaram and Thickstun, 2020))

function SEPARATION($\mathbf{y}$, κ, N, $\{\sigma_t^2\}_{t=1}^T$, $\boldsymbol{\psi}_x^{(0)}$, $\boldsymbol{\psi}_n^{(0)}$) ▷ Specially tuned noise schedule $\{\sigma_t\}_{t=1}^T$
  for $t \leftarrow 1, T$ **do**
    $\boldsymbol{\psi}_x^{(t)} \leftarrow \boldsymbol{\psi}_x^{(t-1)}$, $\boldsymbol{\psi}_n^{(t)} \leftarrow \boldsymbol{\psi}_n^{(t-1)}$, $\eta_t \leftarrow f(\sigma_t^2)$ ▷ Learning rate η_t
    for $i \leftarrow 0, N-1$ **do**
      $\boldsymbol{\epsilon}_x, \boldsymbol{\epsilon}_n \sim \mathcal{N}(0, \mathbf{I})$
      $\widehat{\boldsymbol{\epsilon}}_x, \widehat{\boldsymbol{\epsilon}}_n = \boldsymbol{\epsilon}_{\phi_{\mathbf{x}}}\left(\boldsymbol{\psi}_x^{(t)}, t\right), \boldsymbol{\epsilon}_{\phi_{\mathbf{n}}}\left(\boldsymbol{\psi}_n^{(t)}, t\right)$ ▷ Compute scores at noise level σ_t
      $\boldsymbol{\psi}_x^{(t)} \leftarrow \boldsymbol{\psi}_x^{(t)} - \eta_t \underbrace{(\widehat{\boldsymbol{\epsilon}}_x/\sigma_t)}_{\text{approx. score}} - \eta_t \left(\mathbf{y} - \boldsymbol{\psi}_x^{(t)} - \kappa\boldsymbol{\psi}_n^{(t)}\right)/\sigma_t^2 + \sqrt{2\eta_t}\boldsymbol{\epsilon}_x$
      $\boldsymbol{\psi}_n^{(t)} \leftarrow \boldsymbol{\psi}_n^{(t)} - \eta_t (\widehat{\boldsymbol{\epsilon}}_n/\sigma_t) - \eta_t\kappa \left(\mathbf{y} - \boldsymbol{\psi}_x^{(t)} - \kappa\boldsymbol{\psi}_n^{(t)}\right)/\sigma_t^2 + \sqrt{2\eta_t}\boldsymbol{\epsilon}_n$
  return $\widehat{\mathbf{x}} = \boldsymbol{\psi}_x^{(T)}$, $\widehat{\mathbf{n}} = \boldsymbol{\psi}_n^{(T)}$

5.C BASIS Separation

Similar to our method, BASIS (Jayaram and Thickstun, 2020) leverages pre-trained generative models as statistical priors and does not rely on end-to-end (supervised) training with paired data. The manner in which these priors are used for separation, however differs:

1. **Soft constraint vs. hard constraint:** In the context of our problem formulation, we are interested in decomposing a mixture $\mathbf{y} = \mathbf{x} + \kappa\mathbf{n}$ into its constituent components. As described in Section 5.3 our proposed method recovers the components by extending a MAP estimation rule with a hard constraint given by,

$$\widehat{\mathbf{x}} = \underset{\mathbf{x}\in\mathcal{X} \text{ s.t } \mathbf{y}=\mathbf{x}+\kappa\mathbf{n}}{\arg\max} p_{\mathsf{x}|\mathsf{y}}(\mathbf{x}|\mathbf{y}),$$

   where the resulting estimate of $\mathbf{n}$ is $\widehat{\mathbf{n}} = (\mathbf{y} - \widehat{\mathbf{x}})/\kappa$. The result of imposing this constraint is a MAP objective that leverages two priors but only requires optimization with a single variable,

$$\begin{aligned} \underset{\mathbf{x}\in\mathcal{X} \text{ s.t } \mathbf{y}=\mathbf{x}+\kappa\mathbf{n}}{\arg\max} p_{\mathsf{x}|\mathsf{y}}(\mathbf{x}|\mathbf{y}) &= \underset{\mathbf{x}\in\mathcal{X} \text{ s.t } \mathbf{y}=\mathbf{x}+\kappa\mathbf{n}}{\arg\max} p_{\mathsf{y}|\mathsf{x}}(\mathbf{y}|\mathbf{x}) P_{\mathsf{x}}(\mathbf{x}) && \text{(5.30a)} \\ &= \underset{\mathbf{x}\in\mathcal{X}}{\arg\min} -\log P_{\mathsf{x}}(\mathbf{x}) - \log p_{\mathsf{b}}\left((\mathbf{y}-\mathbf{x})/\kappa\right). && \text{(5.30b)} \end{aligned}$$

   In contrast, BASIS strives to estimate the underlying components by *sampling from the posterior*. In particular, they do not enforce a hard constraint during sampling and instead assume a likelihood of the form

$$p^{\text{BASIS}}_{\mathsf{y}|\mathsf{x},\mathsf{n}}(\mathbf{y}|\mathbf{x},\mathbf{n}) = \mathcal{N}\left(\mathbf{y}; \mathbf{x}+\kappa\mathbf{n}, \gamma^2\mathbf{I}\right), \tag{5.31}$$

   which actually corresponds to an alternate mixture model with auxilliary noise $\mathbf{w} \sim \mathcal{N}(0, \gamma^2\mathbf{I})$ such that,

$$\mathbf{y} = \mathbf{x} + \kappa\mathbf{n} + \mathbf{w}. \tag{5.32}$$

   Thus, the estimates $\widehat{\mathbf{x}}^{\text{BASIS}}$ and $\widehat{\mathbf{n}}^{\text{BASIS}}$ are obtained by sampling from the posterior,

$$p^{\text{BASIS}}_{\mathsf{x},\mathsf{n}|\mathsf{y}}(\mathbf{x},\mathbf{n}|\mathbf{y}) = p^{\text{BASIS}}_{\mathsf{y}|\mathsf{x},\mathsf{n}}(\mathbf{y}|\mathbf{x},\mathbf{n})\, P_{\mathsf{x}}(\mathbf{x}) p_{\mathsf{b}}(\mathbf{n}). \tag{5.33}$$

   Notice that sampling from this posterior requires computing *two separate estimates* due to the soft constraint in Eq. (5.32).

2. **Annealed Langevin dynamics vs. randomized levels of Gaussian smoothing:** The aforementioned BASIS estimates satisfy $\mathbf{y} = \widehat{\mathbf{x}}^{\text{BASIS}} + \kappa\widehat{\mathbf{n}}^{\text{BASIS}}$ only when $\gamma \to 0$ in the soft constraint in Eq. (5.32) enforced via the likelihood term. As shown in Algorithm 5.2, the idea behind BASIS is to leverage annealed Langevin dynamics sampling to sample from the posterior by slowly decreasing the value of γ towards zero by *tuning a specially chosen noise schedule*. While generally a separate noise schedule is required per prior, the authors suggest sharing the same schedule across both priors. Apart from tuning the noise schedule, the learning rate schedule is another hyperparameter that must also be tuned. In contrast, our algorithm leverages the existing noise schedule from pre-trained diffusion models in a randomized fashion *without any additional tuning*.

3. **Optimization via sampling vs. optimization via gradient descent:** Starting with initializations $\boldsymbol{\psi}_x^{(0)}$ and $\boldsymbol{\psi}_n^{(0)}$ for the SOI and interference respectively, BASIS refines the estimate such that $\boldsymbol{\psi}_x^{(t)}$ approximates a sample drawn from the distribution of the smoothened source p_{x_t}. Langevin dynamics leverages the approximate score of p_{x_t}, to update the estimate by *sampling from higher density regions*. A similar update is performed for the interference. The estimates are constrained to satisfy Eq. (5.32) through the Gaussian likelihood.

6 RF Source Separation

In this chapter, we present practical solutions for source separation of radio frequency (RF) signals, focusing specifically on digital communication signals. With the proliferation of intelligent wireless devices competing for limited spectrum resources, ensuring reliable operation within a rapidly evolving heterogeneous wireless networking ecosystem has become increasingly important. Interference from other sources operating in the same channel, e.g., 5G waveforms or WiFi signals, can lead to deterioration in quality of service. To address this challenge, we develop data-driven score-based SCSS methods grounded in the α-RGS framework, which we show achieves substantial performance gains over both traditional and modern approaches. These results highlight new opportunities for intelligent and effective interference mitigation. We conclude the chapter with a broader discussion of alternative RF source separation strategies that may be better suited to different system constraints.

6.1 Background on Digital Communication Signals

At a high level, digital communications deals with the transmission of bits by modulating a so-called "carrier signal". Groups of bits, from which the underlying discreteness of these sources originates, are first mapped to symbols $c_p \in \mathbb{C}$ via the *digital constellation*—a mapping between groups of bits and a finite set of complex-valued symbols. The constellation is chosen (among other considerations) by the number of bits modulated simultaneously. Typically used digital constellations (the mapping between bits and symbols) include binary phase shift keying (BPSK) and quadrature phase shift keying (QPSK). Specific BPSK and QPSK examples are depicted in Figure 6.1.

The symbols are subsequently aggregated to form a complex-valued continuous waveform. The signal models introduced in Section 5.1.3 can be particularized to RF signals. One of the simplest RF signals is a single-carrier signal modulated waveform, e.g., QPSK symbols, which can be expressed as,

$$x(t) = \sum_{p=-\infty}^{\infty} c_{p,\ell}\, g(t - pT_s). \tag{6.1}$$

The second form corresponds to multi-carrier signals, such as Orthogonal Frequency Division Multiplexing (OFDM) signals, where multiple symbols are modulated in parallel,

$$n(t) = \sum_{p=-\infty}^{\infty} \sum_{\ell=0}^{L-1} c_{p,\ell}\, \exp\{j2\pi\ell t/L\}. \tag{6.2}$$

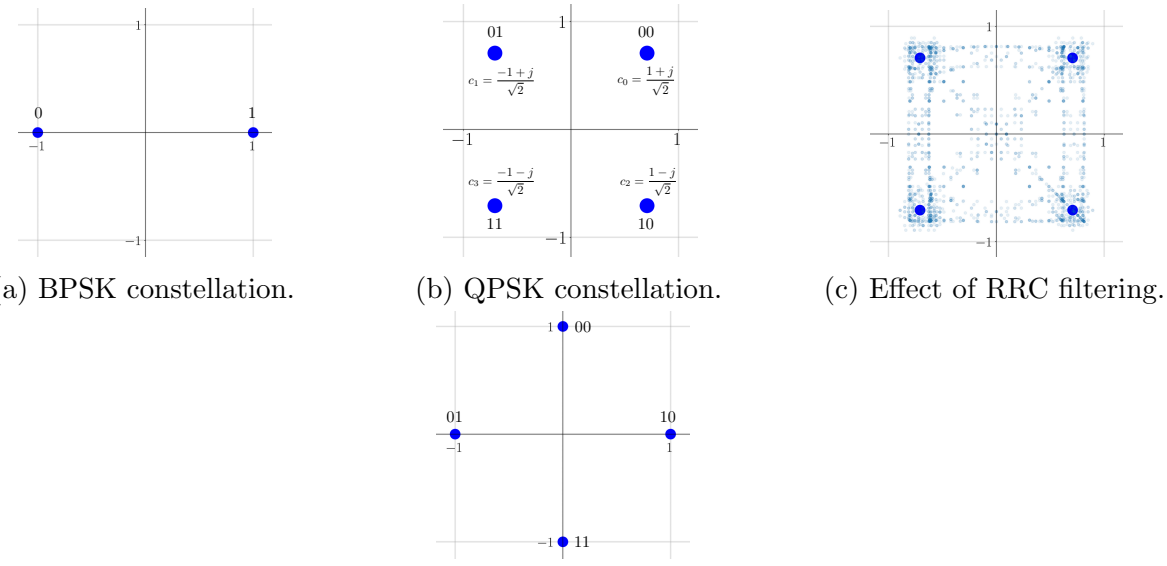
Both types of signals can be grouped in a single family of signals represented as,

$$u(t) = \sum_{p=-\infty}^{\infty} \sum_{\ell=0}^{L-1} c_{p,\ell}\, g(t - pT_s, \ell)\, \exp\{j2\pi\ell t/L\}. \tag{6.3}$$

In the RF domain, $g(\cdot)$ is known as the *pulse shaping* filter and helps limit the signal's bandwidth (Heath Jr, 2017, Sec 4.4.3). As shown above, each symbol $c_{p,l}$ is multiplied with a pulse shaping function with different time offsets and a main lobe width T_s, helping smoothen the signal and remove high frequencies. In this work, we will make use of the root-raised cosine (RRC) pulse shaping for our single-carrier QPSK SOI. On the other had, OFDM signals do not use an explicit pulse shaping function (as shown in Eq. (6.2)), i.e., $g(t, \cdot)$ corresponds to a rectangular function in Eq. (6.3).

Figure 6.2 illustrates a representative modulation pipeline. To recover the bits at the receiver, one may adopt *matched filtering* (MF) (Lapidoth, 2017, Sec 5.8) before the estimation of the underlying symbols, and thereafter decode them back to bits. For commonly used pulse shaping functions, such as the root-raised cosine (RRC) shown in Figure 6.2, the matched filter and pulse shaping filter coincide.

We use $x(t)$ to represent the signal of interest (SOI) and $\mathbf{x}$ as the vector representation for a collection of N consecutive samples thereof. Similarly, the interference is represented by $n(t)$ or $\mathbf{n}$.



(a) BPSK constellation. (b) QPSK constellation. (c) Effect of RRC filtering.

(d) Effect of 45° phase.

Figure 6.1.: Discrete constellations for a BPSK and QPSK signal. Application of the RRC filter in the time domain leads to interpolation between constellation point as shown in (c). Phase offsets in the waveform domain can be corrected by looking at the rotated constellation as in (d).

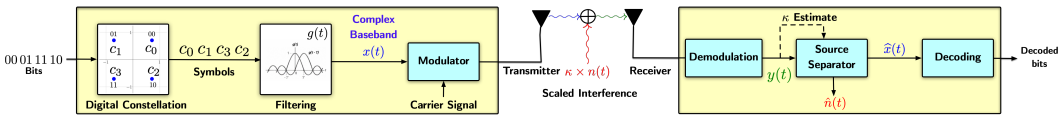


Figure 6.2.: The single-carrier digital modulation pipeline with an intelligent decoder that performs a pre-processing stage of source separation. Illustrated is an example with a QPSK constellation and a root-raised cosine (RRC) pulse shaping function.

6.2 Interference Mitigation

Mitigating co-channel interference (interference from sources within the same channel) is a challenging problem, especially in heterogeneous networks (Damnjanovic et al., 2011; Khandekar et al., 2010; Xu et al., 2021). If the system parameters and signal generation model are known ahead of time, one can leverage this knowledge to devise hand-crafted priors. However, one often deals with interference from more complicated wireless sources, for which the signal model is unknown. In such scenarios, data-driven methods that learn priors from background recordings can be useful. We envision that our SCSS solution based on α-RGS could help mitigate such interference prior to decoding the signal, as shown in the right hand (receiver) side of Figure 6.2.

6.2.1 Classical RF Interference Mitigation Techniques

Two classical approaches for interference mitigation are matched filtering and linear minimum mean squared estimation. We present a brief overview of these techniques as they will serve as baselines in our experiments.

6.2.1.1 Matched Filtering

Matched filtering (MF) is a signal processing technique commonly used in the demodulation of RF waveforms. It exploits knowledge about the signal waveform to enhance the detection and recovery of the transmitted symbols/bits, and is optimal in the maximum-SNR sense for signals contaminated with additive Gaussian noise.

The basic principle involves filtering the received sampled RF waveform with a known reference waveform called the "matched filter". The goal is to maximize the signal-to-interference-plus-noise ratio (SINR) at the filtered output, which consequently minimizes the error probability in the subsequent symbol detection when the noise is Gaussian.

Consider a baseband RRC-QPSK signal and suppose that we adopt a simple additive white Gaussian noise (AWGN) channel model, thereby representing our received signal as

$$\begin{aligned} y(t) &= \sum_p c_p \, g_{\text{tx}}(t - pT_s) + w(t) \\ &= g_{\text{tx}}(t) * \sum_p c_p \, \delta(t - pT_s) + w(t), \end{aligned}$$

where c_p are the symbols from a QPSK constellation, $\delta(\cdot)$ is the dirac delta fucnction, and $w(t) \sim \mathcal{N}(0, \sigma^2_{\text{AWGN}})$ is the additive noise in the observed signal, statistically independent

of all $\{c_p\}$. Of particular interest in this formulation is the transmit pulse shaping function $g_{\text{tx}}(t)$, where we chose to use the RRC function.

At the receiver, we seek a receiver filter, $g_{\text{rx}}(t)$, such that the filtered and sampled output

$$\begin{aligned} y_{\text{filt}}(t) &= \underbrace{g_{\text{rx}}(t) * g_{\text{tx}}(t)}_{:=g(t)} * \sum_p c_p \delta(t - pT_s) + g_{\text{rx}}(t) * w(t) \\ y[n] = y_{\text{filt}}(nT_s) &= \sum_p c_p \, g((n-p)T_s) + \underbrace{\int w(\tau) \, g_{\text{rx}}(nT_s - \tau) \mathrm{d}\tau}_{:=v[n]} \\ &= \underbrace{c_n \, g(0)}_{:=y_s[n]} + \underbrace{\sum_{p \neq n} c_n \, g((n-p)T_s) + v[n]}_{:=y_v[n]} \end{aligned} \tag{6.4}$$

would maximize the output SINR. In other words, we are looking to maximize

$$\text{SINR} = \frac{\mathbb{E}\left[|y_s[n]|^2\right]}{\mathbb{E}\left[|y_v[n]|^2\right]} = \frac{\mathbb{E}\left[|c_n|^2\right] |g(0)|^2}{\mathbb{E}\left[|c_n|^2\right] \sum_{p \neq n} |g(pT_s)|^2 + \sigma^2_{\text{AWGN}} \int |G_{\text{rx}}(f)|^2 \mathrm{d}f}$$

(where $G_{\text{rx}}(f)$ is the Fourier transform of $g_{\text{rx}}(t)$) via an appropriate choice of $g(t)$—and thereby, $g_{\text{rx}}(t)$. This can be done by finding an upper bound on the SINR that reaches equality for the appropriate filter choices. Ultimately, one such choice is $g_{\text{rx}}(t) = g^*_{\text{tx}}(-t)$—termed as the matched filter—that leads to a maximized SINR. In the case of an RRC pulse shaping function (which is real and symmetric), the matched filter is also the same RRC function. As part of the demodulation pipeline, the filtered output is sampled (as in Eq. (6.4)), and then mapped to the closest symbol in a predefined constellation (in the Euclidean distance sense). Finally, we can map these complex-valued symbols back to their corresponding bits to recover the underlying information.

Demodulation with matched filtering is optimal for waveforms in the presence of additive Gaussian noise. However, in our signal separation problem, we consider the presence of an additive interference, which is not necessarily Gaussian. Thus, exploiting the non-Gaussian characteristics of the interference would likely lead to enhanced decoding performance.

6.2.1.2 LMMSE Estimation

Recall that our observation model is

$$\mathbf{y} = \mathbf{x} + \kappa \mathbf{n},$$

where we assume $\mathbf{x}$ and $\mathbf{n}$ are zero-mean and that they are statistically independent. The linear minimum mean square error (LMMSE) estimator is the estimator $\widehat{\mathbf{x}} = \mathsf{W}_{\text{LMMSE}}\mathbf{y}$, such that

$$\mathsf{W}_{\text{LMMSE}} = \underset{\mathsf{W} \in \mathbb{C}^{T \times T}}{\arg\min} \, \mathbb{E}\left[\|\mathbf{x} - \mathsf{W}\mathbf{y}\|_2^2\right]. \tag{6.5}$$

In this case, the optimal linear transformation (in the sense of Eq. (6.5)) can be written as

$$\mathsf{W}_{\text{LMMSE}} = \mathsf{C}_{xy} \, \mathsf{C}_{yy}^{-1} = \mathsf{C}_{xx} \left(\mathsf{C}_{xx} + \kappa^2 \mathsf{C}_{nn}\right)^{-1}$$

where $\mathsf{C}_{xy} := \mathbb{E}[\mathbf{x}\mathbf{y}^{\mathrm{H}}]$ corresponds to the cross-covariance between $\mathbf{x}$ and $\mathbf{y}$, C_{yy}, C_{xx}, C_{nn} are the auto-covariance of $\mathbf{y}$, $\mathbf{x}$ and $\mathbf{n}$ respectively. The second equality is obtained by statistical independence, thereby $\mathsf{C}_{xy} = \mathsf{C}_{xx}$, $\mathsf{C}_{yy} = \mathsf{C}_{xx} + \kappa^2\mathsf{C}_{nn}$.

In relevant literature, this may also be referred to as linear MMSE receivers (Tse and Viswanath, 2005, Sec 8.3.3). Note that we aim to mitigate the effects of an additive interference channel. For the purposes of this work, we use LMMSE as one of the baselines as a signal separation (interference mitigation) method. Thereafter, we assess performance based on the squared error between $\widehat{\mathbf{x}}$ with the ground truth $\mathbf{x}$. To obtain the underlying bits, we perform a standard matched filtering operation on the estimator $\widehat{\mathbf{x}}$.

Note that the LMMSE estimator is optimal if the components were Gaussian. However, as digital communication signals contain some underlying discreteness and undergo unknown time-shifts, these signals are typically non-Gaussian (and often, even far from Gaussian). Hence, better performance can generally be obtained through nonlinear methods.

6.2.2 Deep Learning for RF Systems

Recently deep learning methods have demonstrated the potential to reap significant gains over handcrafted model-based methods in RF applications (Eldar et al., 2022; Oyedare et al., 2022). Some works have studied the problem of symbol detection (Samuel et al., 2019; Shlezinger et al., 2020a), where they assume that the channel is stationary. Other works, such as DeepSIC (Shlezinger et al., 2020b), use deep learning for interference cancellation in the multi-user setting within the same channel. In contrast, we deal with the more challenging setting of non-stationary interference, thereby requiring efficient exploitation of intricate temporal structures. While the latter works consider the superposition of independent and identically distributed sources (same technology), we assume unknown additive interference (cross technology), a hard problem to solve with naïve decoding methods in the absence of explicit prior knowledge about the interference. Our problem formulation is closer to recent work in (Lee et al., 2022). However, they learn an end-to-end estimate of the signal from paired data samples. We instead assume restricted access to joint data, with a focus on capturing properties of the components through independent priors.

6.3 Experiments

We now detail the setup for training diffusion models on RF signals. We subsequently explain how to use the learned score estimator to implement our RF source separation algorithm based on the α-RGS framework (see Chapter 5).

6.3.1 RF SCSS Formulation

We are interested in recovering $\mathbf{x}$, the signal of interest (SOI), from a mixture $\mathbf{y} = \mathbf{x} + \kappa\mathbf{n}$, where $\mathbf{n}$ is assumed to be a co-channel interference with unknown system parameters. We evaluate performance using two metrics—i) the mean squared error (MSE), that measures the distortion between the estimated SOI and the ground truth; and ii) the bit error rate (BER) of the decoded discrete representation, which is obtained from the estimated SOI by

extracting the underlying bits. The latter measure is particular to digital communication signals as it captures the fidelity of the estimated representation that is only partially reflected in the MSE criterion.

6.3.2 Datasets

We trained diffusion models on different RF datasets —i) synthetic QPSK signals with RRC pulse shaping, ii) synthetic OFDM signals (BPSK and QPSK) with structure similar to IEEE 802.11 WiFi signals; and iii) signals corresponding to "CommSignal2" from the RF Challenge (Lancho et al., 2024), which contains datasets of over-the-air recorded signals. All synthetic datasets were created using the NVIDIA Sionna toolkit (Hoydis et al., 2022). All datasets contain between 150k - 500k samples and we use a 90-10 train-validation split during training.

6.3.3 Diffusion Model Training

We adopted the Diffwave (Kong et al., 2021) architecture for our experiments, with a minor changes (see Appendix 6.A) to accommodate the complex-valued nature of our signals. Our models were trained in the waveform domain on inputs of length 2560 with the real and imaginary components concatenated in the channel dimension. We trained unconditional diffusion models and assumed no access to knowledge about the signal generation model. We used noise standard deviation in the range $(0.03, 99.97)$ discretized into 50 levels. We trained for 500k steps with early stopping on 2 x NVIDIA 3090 GPUs.

6.3.4 Source Separation Setup

We considered three different mixtures in our experiments, all using an RRC-QPSK signal as the SOI $\mathbf{x}$. The interference signal $\mathbf{n}$ was one of OFDM (BPSK), OFDM (QPSK) or a windowed recording from the CommSignal2 dataset. Our proposed algorithm used $\alpha = \kappa^2$ and was initialized with the MF solution given the mixture $\mathbf{y}$. Note that κ can be equivalently described as the signal to interference ratio ($\text{SIR} := 1/\kappa^2 := \mathbb{E}_{\mathsf{x}}\left[\|\mathbf{x}\|_2^2\right] / \mathbb{E}_{\mathsf{n}}\left[\|\kappa\mathbf{n}\|_2^2\right]$). We assumed that κ was known[1] and used $N = 20,000$ with a cosine annealing learning rate schedule (Loshchilov and Hutter, 2017). The OFDM mixtures used $(\eta_{\max}, \eta_{\min}) = (5\text{e}{-3}, 1\text{e}{-6})$ and the CommSignal2 mixture used $(\eta_{\max}, \eta_{\min}) = (2\text{e}{-3}, 1\text{e}{-6})$. Importantly, we *re-used the training noise levels* from the diffusion models and randomized over all but the smallest noise level. We tested performance across SIR levels ranging from -24 dB to -3 dB ("strong interference" regime), by changing the value of κ in the mixture. Each set of separation experiments was conducted on a single NVIDIA V100 GPU.

6.3.5 Baselines

We compared our proposed method against baselines that also leverage independent statistical or structural priors over the sources. The simplest baseline, which nevertheless is still

[1] Many communication systems have power constraints and equalization capabilities, and with the endowment of such knowledge it is possible to estimate the signal to interference ratio (SIR) within reasonable margin.

commonly used in most communication systems, is the matched filtering solution, which treats the interference as white Gaussian noise. The linear minimum mean square error (LMMSE) solution, a commonly used technique for signal estimation, is another baseline that leverages (up to) second-order statistics of the underlying source distributions.

We also compared with the BASIS separation algorithm (see Section 5.5), which is the closest learning-based method that resembles our problem formulation. Applying their method as is yielded poor results, and hence we modified the original hyperparameters by tuning the annealing schedule to the best of our abilities for a fair comparison.

We also implemented a baseline based on simulating the reverse diffusion process (Ho et al., 2020) as a denoiser. Given a mixture $\mathbf{y}$, we interpreted the SOI as (scaled) additive noise on top of the interference $\mathbf{n}$. Since the reverse diffusion process can be interpreted as an iterative denoiser , we ran a small chain of reverse diffusion for 10 timesteps starting at a noise standard deviation of 0.07 and ending with variance of 0.03, which was chosen based on different trials. Similar ideas have been used in prior works involving inverse problems for images (Kadkhodaie and Simoncelli, 2021; Kulikov et al., 2022). We note that it is generally cumbersome to find the optimal reverse diffusion noise range and it might in fact be dependent on the specific mixture's SIR.

To study the fidelity of our learned score models, we derived the analytical score function of the QPSK SOI in the symbol domain (i.e., before pulse shaping). We used this analytical score as another comparison to demonstrate the performance of our method if the score was known perfectly. This formulation is closer to a learning-based interference mitigation setting, where we assume perfect knowledge about the SOI model, and rely on a learned interference model. For more details please refer to Appendix 6.B.

6.3.6 Source Separation Results

Figures 6.3 and 6.4 show the complete source separation results for mixtures with OFDM and CommSignal2 as interference, respectively. Our model that uses an analytical SOI score for the SOI and a diffusion-based score for the interference generally performs the best and outperforms all baselines. Furthermore, we show that using a learned SOI score still outperforms all baselines in terms of BER, despite the slight degradation. The trained SOI score models consistently outperform all methods at high SIR in terms of BER since the SOI diffusion model was trained with RRC-QPSK samples as opposed to the approximations made in implementing the analytical score.

Our method also outperforms baselines in terms of MSE for OFDM mixtures as shown in Figure 6.3. The performance at low SIR in the context of CommSignal2 mixtures is not the same. As shown in Figure 6.4, the large MSE at low SIRs suggest that there is background noise present in the CommSignal2 sources that is amplified for large values of κ, i.e., the interference is actually of the form $\mathbf{n} + \mathbf{w}$ for some background noise $\mathbf{w}$. We validated that this was indeed the case, by visualizing samples in both the time-domain and frequency domain using the RF challenge demo notebook[2]. As shown in Figure 6.5, we noticed segments of lower magnitude at the start and end, which we believe to be background noise that is not part of intended communication signal. During source separation, this presumably results in

[2] https://github.com/RFChallenge/rfchallenge_singlechannel_starter/blob/main/notebook/Demo.ipynb

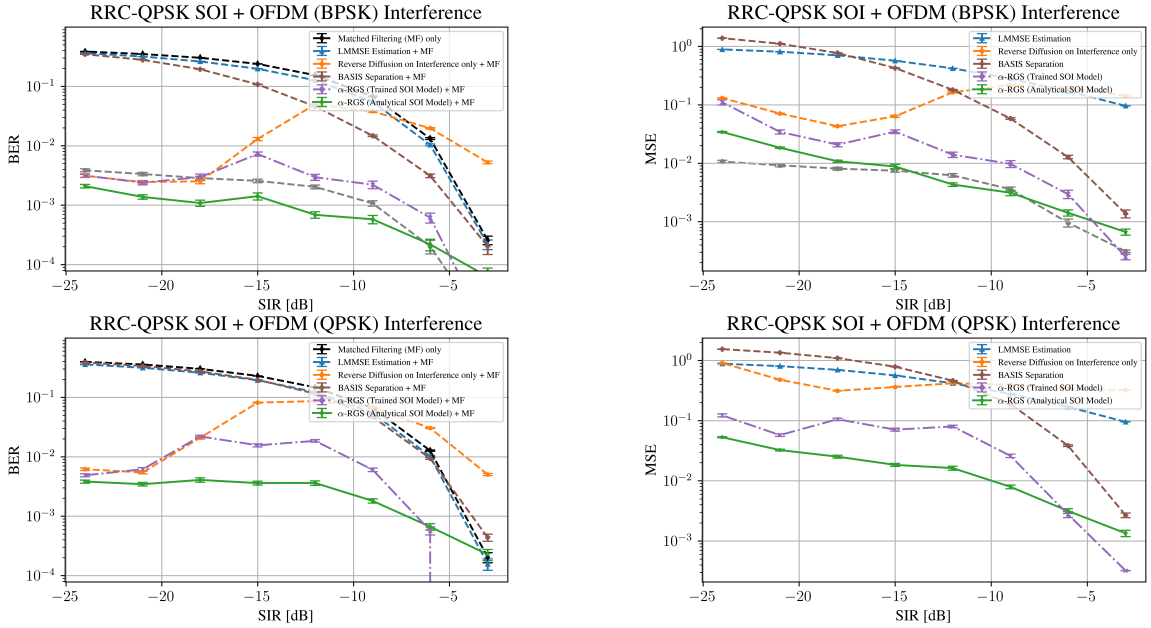


Figure 6.3.: Comparing our method against various baselines on separating (**top**) RRC-QPSK + OFDM (BPSK) mixtures and (**bottom**) RRC-QPSK + OFDM (QPSK) mixtures. Our model that uses an analytical SOI score outperforms all baselines in terms of BER. Reverse diffusion is competitive at low SIRs since the interference dominates the mixture in this regime and hence iterative denoising to separate the SOI is effective. The trained SOI diffusion models significantly outperform all baselines at high SIR. The reason the analytical SOI performs slightly worse is due to the approximations we make.

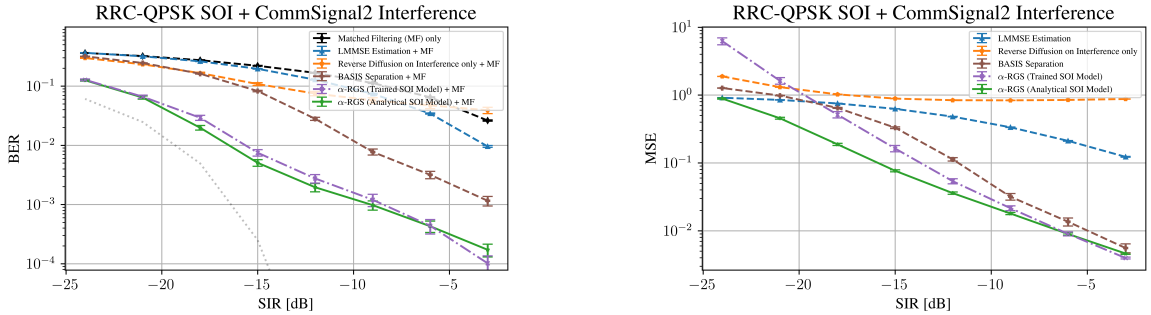


Figure 6.4.: Comparing our method against various baselines on separating RRC-QPSK + CommSignal2 mixtures. Our method significantly outperforms all baselines in terms of BER. The large MSE at low SIRs suggests that there is background noise present in the CommSignal2 source, which is amplified at lower SIR (large κ). We try to estimate the amount of noise and model it as additive Gaussian noise. Accounting for this noise could presumably lead to a lower bound on the BER, shown by dotted black line on the left.

a noisier estimate of the SOI in comparison to mixtures with no additional background noise. We estimated the SNR to be 16.9 dB by averaging across multiple samples. The dotted black

curve on the left of Figure 6.4 is a presumable lower bound on the BER by accounting for the magnitude of the background noise and modeling it as additive white Gaussian noise.

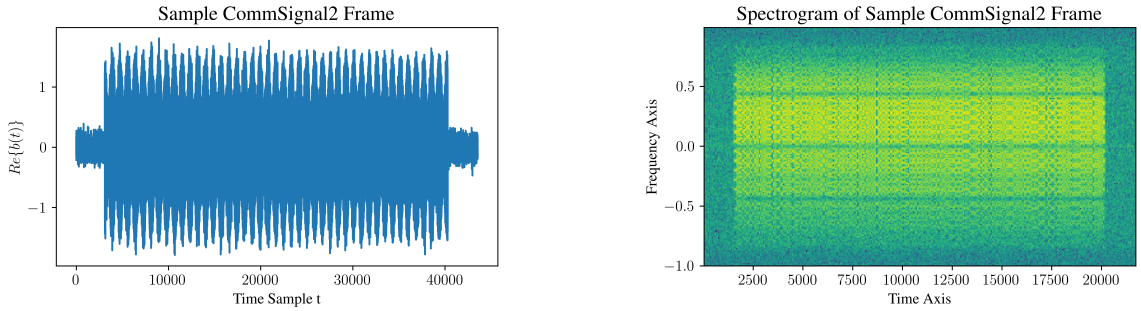


Figure 6.5.: Time-domain and frequency-domain plots of a single frame from the CommSignal2 dataset. **Left:** In the time-domain we observe segments of lower magnitude at the start and end, which we believe to be background noise. **Right:** In the frequency-domain we observe the bandlimited communication signal with spectrally flat features in the regions we identified as background noise.

Nevertheless, across all experiments, we demonstrate that our method sets a new state-of-the-art for applications such as interference mitigation where the interference can be learned from data recordings and the desired information can be decoded with knowledge of the SOI demodulation pipeline.

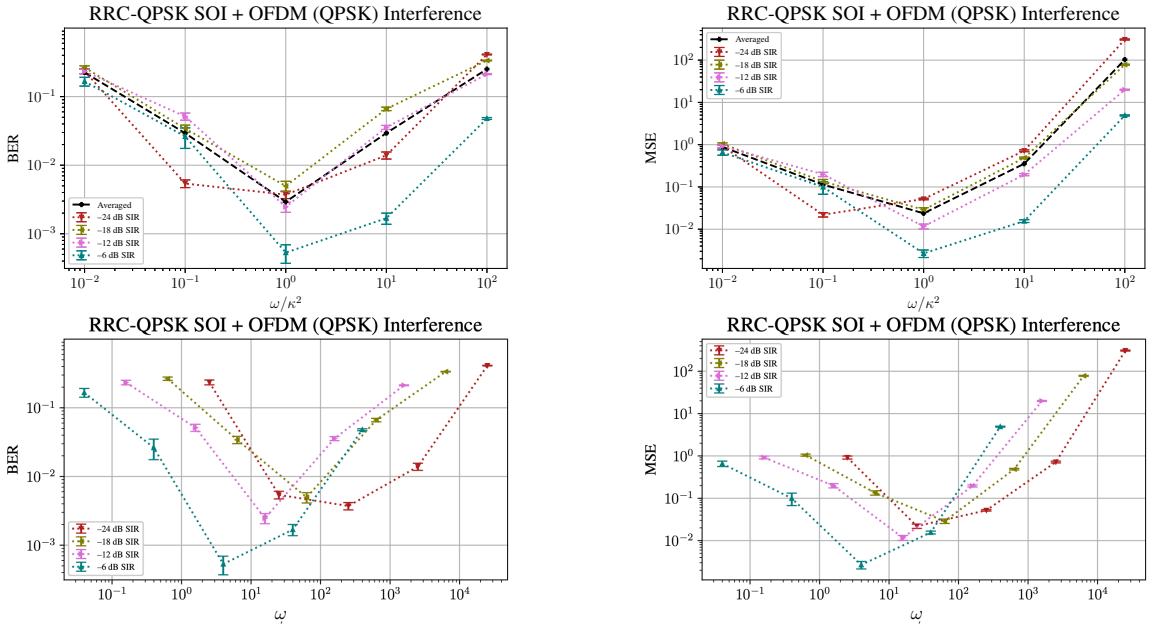


Figure 6.6.: **Top:** BER and MSE versus α/κ^2 for different SIR levels. Evidently, a good choice of α, on average, across different noise levels is $\alpha = \kappa^2$. **Bottom:** By looking at individual SIRs we see that the minimum BER and MSE is achieved when α increases with increasing κ^2 (decreasing SIR), visualized using a log scale on the x-axis.

6.3.6.1 Choice of α

We numerically verifed that $\alpha = \kappa^2$ is a good choice of the α-posterior term, To this end, we first found a suitable order of magnitude for α, by varying α/κ^2 between 10^{-2} and 10^2 across different noise levels. As shown in the top row in Figure 6.6, on average, the minimum BER and MSE is achieved when $\alpha = \kappa^2$. We additionally validated that it is beneficial to adapt α as the SIR changes by varying α and studying the BER and MSE curves at individual noise levels. As shown in the bottom row of Figure 6.6, we observe that α should increase with κ^2 to achieve good results.

6.3.6.2 Comparison with Supervised Methods

We are interested in leveraging independently trained priors in our source separation setup. Nevertheless, we compared against a supervised setup that learns to separate mixtures end-to-end by training a UNet on *paired* data. We trained three supervised models based on the recent work in (Lancho et al., 2024), using their open-sourced training code[3]. These models are trained with an ℓ_2 loss. As such, we should expect the MSE performance to be better than our unsupervised approach, which is indeed the case across all the mixtures as shown in Figure 6.7. However, we notice that our method, which uses an analytical SOI score, is able to perform similarly to the UNet and even better in terms of BER for some SIRs, especially in the OFDM interference setting. Furthermore, in the challenging low SIR (strong interference regime) our method performs well, showing that our interference diffusion models were able to learn the underlying statistical structures. Thus, if the mixture model changes in the future we can re-use our priors whereas the supervised approached could require an entire change to the architecture and additional training. We can potentially drive down the BER in the CommSignal2 setting by modeling the background noise in these signals either through a different mixture model or through additional priors.

6.3.6.3 Computation Time

The inference time is as of now slightly longer than BASIS. To separate a single mixture, our method that uses $N = 20,000$ iterations takes 328 seconds on average, whereas BASIS takes 284 seconds for the same number of iterations. Meanwhile, the UNet only requires one forward pass through the model and can separate a mixture in less than one second.

6.4 Summary and Future Directions

Experiments on RF sources demonstrate that our approach achieves significantly improved separation performance—yielding up to 95% gains in both BER and MSE compared to classical and existing score-based SCSS methods. In the current problem setup, the background noise in the CommSignal2 scenario is assumed to be entangled with the interference signal. However, if a statistical model of the background noise (in the form of a known likelihood) is available, the generalized extension of α-RGS, described in Section 5.7, can be applied to more precisely

[3] https://github.com/RFChallenge/SCSS_CSGaussian

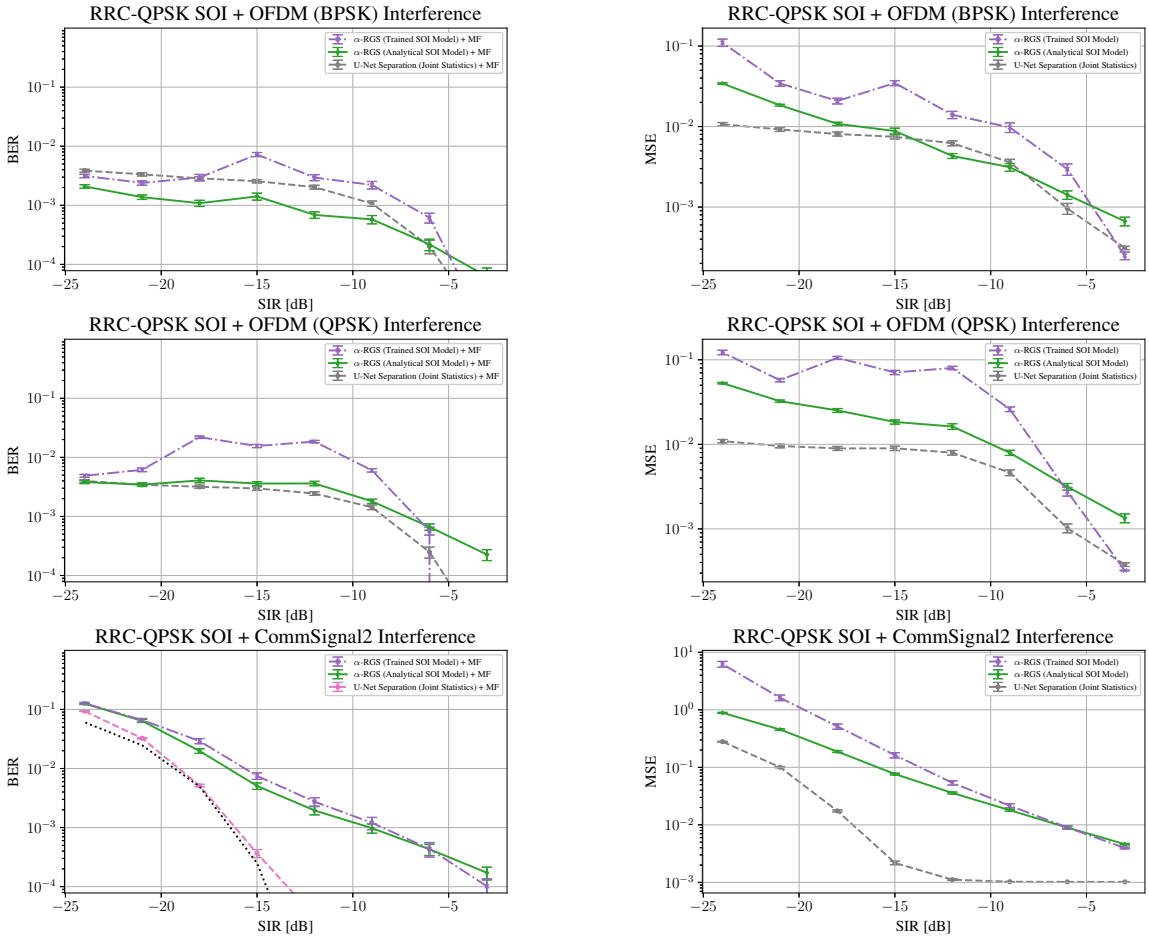


Figure 6.7.: Comparing our method against a supervised learning setup that trains a UNet on paired data samples with an ℓ_2 loss. We show that we are able to achieve competitive BERs, and even outperform the supervised method at certain SIRs. Our method is particularly competitive in the challenging strong interference regime (low SIR), demonstrating the fidelity of our trained interference diffusion models. The gap is larger for CommSignal2 mixtures as the supervised method is presumably able to leverage knowledge of the joint statistics between the SOI and interference to effectively deal with the background noise in the interference signal.

isolate the sources while still leveraging independent priors over signals. This framework can also be extended to handle mixtures with more than two component signals. Looking ahead, an exciting direction for future work involves accelerating the algorithm to enable real-time source separation.

6.5 Real-Time Solutions*

Throughout this chapter, we primarily assumed restricted access to paired samples of the SOI and their corresponding mixture signals. Our proposed solution leveraged powerful diffusion-based priors to achieve source separation without relying on joint statistics. While α-RGS demonstrated strong performance, consistently outperforming both classical and learning-based baselines, its computational cost remains a major limitation—making real-time deployment impractical in many scenarios. In contrast, supervised learning approaches, given paired SOI-mixture recordings, offer compelling advantages in terms of speed. As noted earlier, UNet-based models can separate a mixture signal in under a second. Although α-RGS typically yields better separation quality, the ability to perform real-time inference makes supervised models highly valuable in practical applications. In the following sections, we explore how to build more efficient and faster solutions for real-time RF source separation.

6.5.1 WaveNet with Dilated Convolutions

Unlike standard UNet architectures used for image processing, the architecture adapted for RF source separation in (Lancho et al., 2024) features an initial convolution layer with kernel size of 101 to capture the effective correlation length of the both the SOI and the interference. For example, this long kernel could capture long-scale temporal structures in OFDM signals such as the cyclic prefix which could lead to significant performance gains. Inspired by this we adapt the the DiffWave architecture for RF source separation in order to leverage its large receptive field to capture long-range temporal structures. The main change from our diffusion modeling experiments is that the architecture is trained to regress against the ground truth SOI by minimizing an MSE loss, similar to the original WaveNet architecture that it is based on Oord et al. (2016), and hence we call the resulting model WaveNet. More architectural details can be found in Appendix 6.A.

Unlike the downsampling and upsampling networks used in UNets, WaveNet preserves the temporal resolution at each hidden layer by leveraging dilated convolutions. The dilated convolution can be interpreted as a "virtual" kernel with spacing between elements, enlarging the effective receptive field of the convolution. For example, a dilated convolution with a kernel width of 3 and a dilation of 2 has an effective receptive field of 5. Thus, rather than employing a single large convolutional kernel, the WaveNet progressively grows the receptive field of the network, thereby accurately modeling both local and global temporal interactions. Unlike the UNet we found that the WaveNet also allowed for mixed precision training without significant drop in performance, thereby allowing for more powerful and lightweight source separation solutions.

Similar to the UNet model, we trained the WaveNet with an ℓ_2 loss. Formally, denoting the WaveNet separator as $\mathbf{f}_\theta : \mathbb{C}^D \to \mathbb{C}^D$ and given a dataset of SOI-interference pairs $(\mathbf{x}, \mathbf{n})$, the training objective is

$$\min_\theta \mathbb{E}_{p(\mathbf{x})p(\mathbf{n})p(\kappa)} \|\mathbf{x} - \mathbf{f}_\theta(\mathbf{y})\|_2^2, \tag{6.6}$$

where $\mathbf{y} = \mathbf{x} + \kappa\mathbf{n}$ and $p(\kappa)$ is a distribution over the choice of SIR, which we assume to be uniform over the range $[-30, 0]$ dB in practice. In practice we compute empirical expectations using a minibatch of samples.

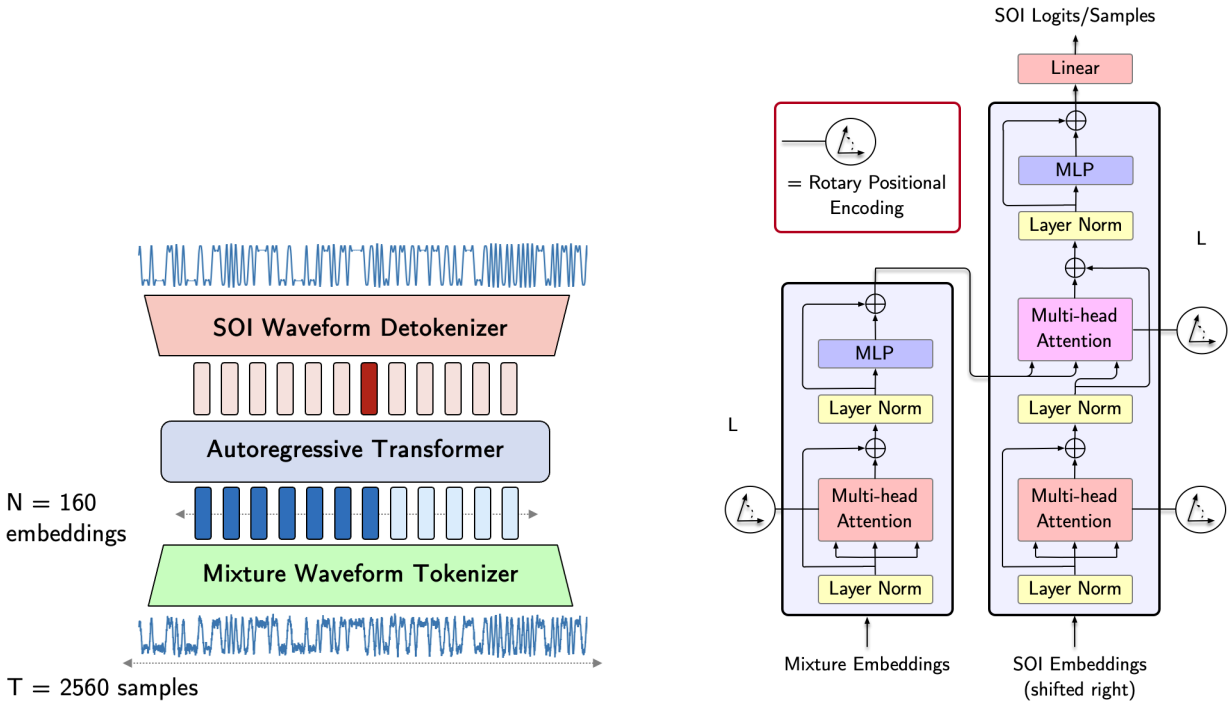


Figure 6.8.: **Left:** Real-time streaming operation of an autoregressive RF source separation transformer. The current SOI token being decoded is shown in red. The transformer leverages past mixture tokens shown in dark blue and previously decoded SOI tokens. **Right:** An encoder-decoder transformer for RF source separation.

6.5.2 Low-Latency Autoregressive Transformer

The UNet and WaveNet architectures operate by processing entire waveforms in parallel, making them ideal for offline or batch processing. However, this design introduces significant latency in streaming scenarios, where signal samples arrive sequentially. These models require buffering large segments of the waveform before producing any output, which is impractical for real-time applications. To address this limitation and enable low-latency source separation, we introduce autoregressive models built on transformer architectures (Vaswani, 2017) that can operate causally and efficiently in a streaming setup.

To facilitate causal processing, the incoming mixture waveform is divided into small, overlapping windows—each spanning only one or two SOI symbols. Instead of working with raw complex-valued samples, each window is projected into a compact vector embedding using a simple linear tokenization module:

$$\mathbf{c}_{\text{in}}^{(i)} = \mathsf{W}_{\text{in}}\mathbf{y}^{(i)},$$

where $\mathbf{y}^{(i)}$ is the i'th mixture window, W_{in} is a learned matrix and $\mathbf{c}_{\text{in}}^{(i)}$ is the embedding or tokenized representation of the mixture window. These overlapping tokens provide a smooth temporal representation of the signal and are streamed and cached as new samples arrive.

These tokenized mixture representations are streamed and cached as they arrive. Concur-

rently, a transformer module (shown on the right of Figure 6.8) decodes the corresponding SOI token (in red) for the current window, using attention mechanisms over both the cached mixture tokens (in blue) and previously decoded SOI tokens. In our experiments an SOI token is a just vectorized representation of the currently decoded SOI window.

The core operation of the transformer is multi-head attention, which computes a weighted sum of value vectors based on the similarity between query and key vectors. Given a matrix of input embeddings C_{in} and a matrix of reference embeddings C_{ref}, we define:

$$\mathsf{Q} = \mathsf{W}_{\mathsf{Q}}\mathsf{C}_{\text{in}}, \quad \mathsf{K} = \mathsf{W}_{\mathsf{K}}\mathsf{C}_{\text{ref}}, \quad \mathsf{V} = \mathsf{W}_{\mathsf{V}}\mathsf{C}_{\text{ref}}$$

where Q is the matrix of queries, K is the matrix of keys and V is the matrix of values. The output embeddings of the attention layer are obtained by taking each query embedding and computing a weighted sum of the values based on the similarity between queries and keys, i.e.,

$$\mathsf{C}_{\text{out}} = \text{softmax}\left(\frac{\mathsf{Q}\mathsf{K}^{\intercal}}{\sqrt{d_k}}\right)\mathsf{V}.$$

Here d_k is the dimensionality of the key embeddings and all matrices above are defined such that the queries, keys and values have the same dimensionality. Typically the output from the attention layer is adding back to the input embeddings C_{in} and then passed to an MLP layer.

When $\mathsf{C}_{\text{in}} = \mathsf{C}_{\text{ref}}$, the mechanism is called self-attention; otherwise, it is referred to as cross-attention. The encoder employs self-attention over the mixture tokens to model interactions within the mixture signal, while the decoder combines self-attention with previously decoded SOI tokens and cross-attention with the streamed in mixture tokens to perform source separation.

Training is done using a ℓ_2 loss, similar to WaveNet. However, the transformer offers greater flexibility: its autoregressive structure naturally supports variable-length sequences and real-time decoding. When efficiently implemented—e.g., using low-bit quantization and high-throughput accelerators like NVIDIA H100s—the transformer can achieve real-time operation at throughput exceeding 1 Mbps, all while maintaining competitive performance.

6.5.3 Results on Real-World Mixtures

We trained all models on signals of length 40,960 using 2 × A6000 GPUs. Unlike the UNet, which required manual early stopping, the new architectures were trained with a cosine annealing learning rate schedule. To address the data-hungry nature of transformers, we applied a range of synthetic augmentations—including time shifts, phase shifts, and Doppler shifts—to both the SOI and interference signals, enhancing model generalization. The WaveNet model was designed with 128 hidden channels and 30 residual layers, while the Transformer featured 14 encoder and decoder layers, each with a hidden dimension of 256. All inference experiments were carried out on a single NVIDIA 3090 GPU.

As illustrated in Figure 6.9, these new architectures significantly outperform the UNet-based solution, achieving substantial improvements through more sophisticated model design. While WaveNet sets a new state-of-the-art in separation quality, the Transformer achieves impressive results despite operating in a fully causal manner. With efficient implementation

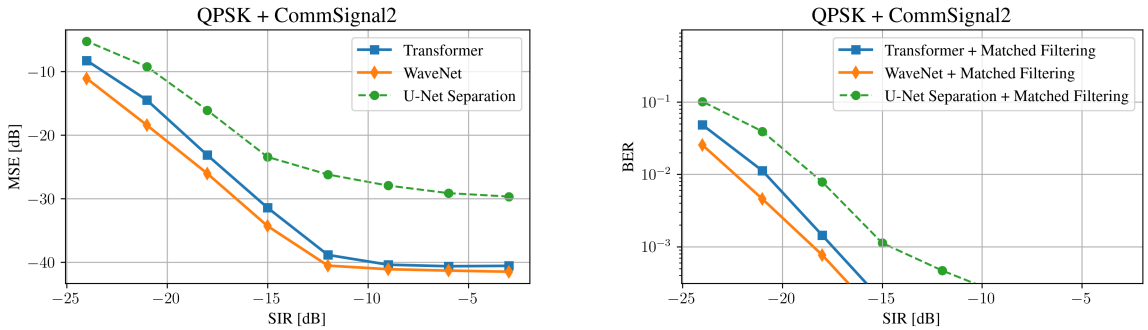


Figure 6.9.: MSE and BER plots for source separation of QPSK + CommSignal2 mixtures using different supervised learning solutions.

using low-bit precision and deployment on high-end accelerators such as the NVIDIA H100, the Transformer has the potential to reach decoding speeds exceeding 1 Mbps—well beyond WaveNet's capabilities. Notably, it achieves this while being approximately 25 times larger than WaveNet, underscoring the promise of software-hardware co-design in enabling cutting-edge real-time RF separation solutions.

Appendix

6.A Diffusion Models for RF Signals

We trained diffusion models to learn the statistical structure inherent to digital RF signals. We adopted an architecture based on an open-sourced implementation[4] of the DiffWave (Kong et al., 2021) diffusion model, initially developed for speech synthesis.

As illustrated in Fig. 6.10, DiffWave employs R residual blocks (He et al., 2016) with dilated convolutions (Oord et al., 2016), where the output of block $i-1$ serves as an input to block i, $i \in \{0, \ldots, R-1\}$. The dilated convolutions assist in the learning of long range temporal and periodic structures. The dilations first start small and successively get larger, such that the dilation at block i is given by $2^{i \bmod m}$, where m is the dilation cycle length. For example, if the dilation periodicity is $m = 10$, then in block $i = 9$ the dilation is 512 and in block 10 the dilation is reset to 1. This allows the network to efficiently tradeoff between learning local and global temporal structures. All residual blocks use the same number of channels, C.

The output of each residual block is passed not only to the next block but also aggregated into a unified feature representation. This aggregated feature is then processed by an MLP, implemented as a 1×1 convolution, to produce the final waveform output. This intermediate feature aggregation, known as a skip connection, helps retain low-level information that might

[4] https://github.com/lmnt-com/diffwave

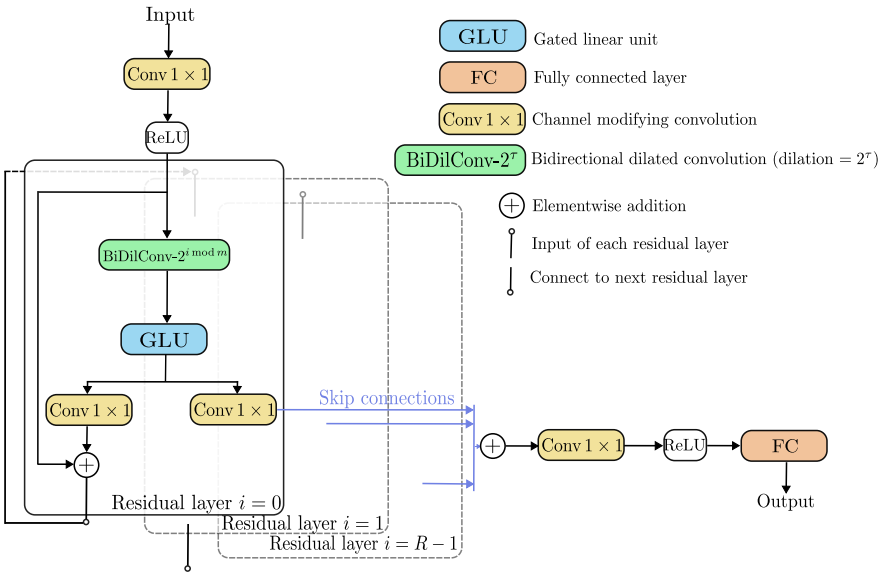


Figure 6.10.: DiffWave architecture for modeling RF signals.

Table 6.1.: Hyperparameters used for training diffusion models.

	QPSK	OFDM (BPSK)	OFDM (QPSK)	CommSignal2
Number of residual blocks (R)	30	30	30	30
Dilation cycle length (m)	10	10	10	10
Residual channels (C)	64	128	256	128
Random shift + phase rotation	✗	✓	✓	✓
Batch size	128	128	64	128
Learning rate	5e−4	5e−4	5e−4	1e−4
Early stopping iteration	360,000	220,000	340,000	90,000

otherwise be lost in deeper networks.

6.A.1 Architectural Modifications

Several changes were made in order to facilitate training with RF signals. First, since we are dealing with complex-valued continuous waveforms, we trained on two channel signals where the real and imaginary components of the RF signals are concatenated in the channel dimension. Second, while the open-source implementation uses an ℓ_1 loss for training, we trained with an MSE (squared ℓ_2) loss to match the training objective in (Ho et al., 2020). We monitored the validation MSE loss, and once the loss stops decreasing substantially, we chose to stop training early. Lastly, we had to increase the channel dimension C to learn more complicated RF signals, e.g., OFDM (QPSK). We trained all our models on 2 × NVIDIA 3090 GPUs. Additionally, during data loading, we performed random time shifts and phase rotations on the OFDM (BPSK), OFDM (QPSK) and CommSignal2 signals. Physically, these simulate transmission impairments in RF systems. The hyperparameters that we used for training each model are shown in Table 6.1.

6.A.2 Evaluation Metrics

In the image domain, metrics such as the Fréchet Inception Distance (FID) (Heusel et al., 2017) can be used to assess the quality of generative models. However, such perceptual metrics are less relevant to the digital communications domain. In the RF domain, the fidelity is measured by the ability to extract the underlying transmitted bits. Hence, in the context of our synthetic datasets, we probed generated samples and compared the estimated received symbols with the underlying constellation. However, for real world signals such as CommSignal2, for which we do not know the system parameters, we have no other means to assess the fidelity apart from looking at time-domain structure. This is another motivation for studying RF source separation, as it provides us with a framework to assess the quality of the learned statistical priors.

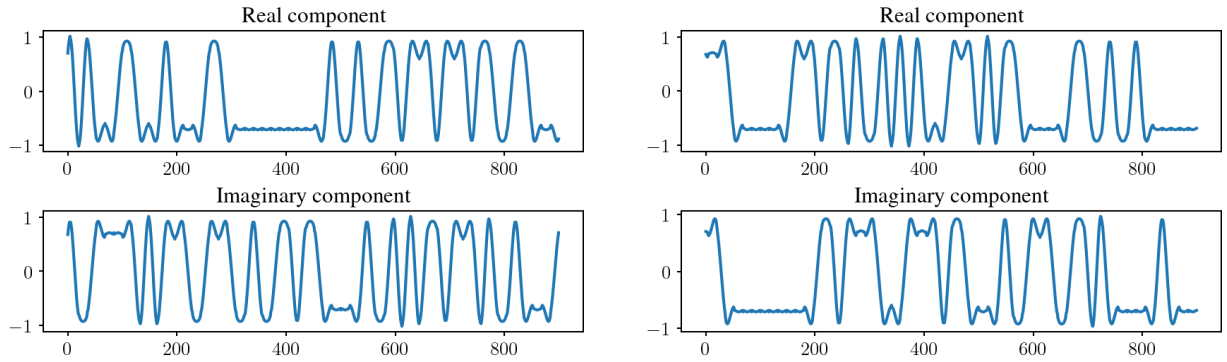


Figure 6.11.: **Left:** A ground truth RRC-QPSK time-domain waveform. **Right:** A sample generated by our trained RRC-QPSK diffusion model. Evidently, the generated waveform resembles the true one.

6.A.3 RRC-QPSK

We trained a diffusion model on the RRC-QPSK dataset. While a QPSK signal has four distinct constellation points, application of the RRC filter results in interpolation between points as shown in Figure 6.1c. Figure 6.11 shows an example of a ground truth RRC-QPSK waveform on the left. On the right we show a RRC-QPSK sample generated by our diffusion model. While it is hard to judge whether it has learned the underlying discrete structure, visually, the waveform seems to have characteristics similar to the ground truth.

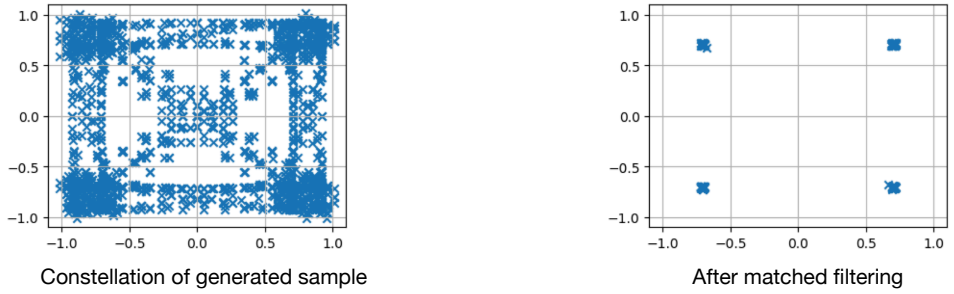


Figure 6.12.: Samples generated from the diffusion model trained on RRC-QPSK samples. The image on the left is underlying constellation of the realization generated by the diffusion model. Notice that the RRC filtering results in interpolation between the QPSK constellation points. After applying MF, the effects of pulse shaping are reversed and the original QPSK constellation is recovered.

In Figure 6.12, we show that by probing generated samples for the underlying (interpolated) symbols, we do indeed recover the constellation for an RRC-QPSK signal. Note that we can do such probing since we know the parameters and signal generation model for an RRC-QPSK signal. Furthermore, by performing MF and removing the effects of pulse shaping we are able to recover the original QPSK constellation, thus demonstrating that the diffusion model has indeed successfully learned the discrete constellation along with the pulse shaping function from data samples.

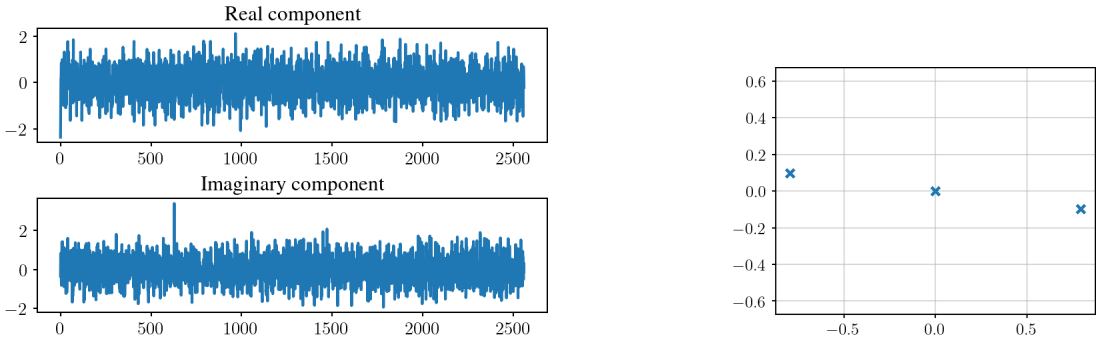


Figure 6.13.: **Left:** A complex-valued OFDM (BPSK) source augmented with a time shift of 8 samples and with a random phase rotation plotted in the time domain. In the time domain, the discrete structure is not discernible and the time-domain waveforms visually look like Gaussian noise. **Right:** Extracting the underlying (rotated) symbols from the waveform on the left by demodulating the OFDM signal using oracle knowledge about the FFT size, cyclic prefix, and time shift.

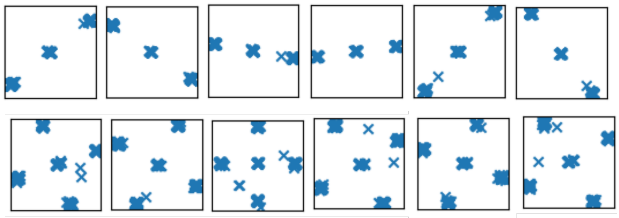


Figure 6.14.: **Top:** Probing six generated samples from the OFDM (BPSK) diffusion model recovers the underlying BPSK constellation after time synchronization. **Bottom:** Probing six generated samples from the OFDM (QPSK) diffusion model recovers the underlying BPSK constellation after time synchronization.

6.A.4 OFDM (BPSK and QPSK)

We trained two OFDM diffusion models, on OFDM (BPSK) and the OFDM (QPSK) dataset of synthetic signals respectively . Figure 6.13, shows an example of an OFDM (BPSK) signal from our dataset. On the left we plot the real and imaginary components of the waveform. These components visually look like Gaussian noise and it is not evident that there is actually inherent structure to these signals. The OFDM signals can be demodulated using oracle knowledge about the FFT size and the cyclic prefix. Additionally, since the signal undergoes a random time shift and phase rotation, the underlying (rotated) BPSK constellation will only be visible when the time shift has been compensated for, i.e., the signal has been time-synchronized, as shown on the right in Figure 6.13.

The top row of Figure 6.14 shows the recovered constellation for six generated OFDM (BPSK) symbols and the bottom row shows the same for six generated OFDM (QPSK) signals. Note that since there are some null symbols, i.e., unused subcarriers in the frequency domain, there is an additional "constellation point" at the origin. In general the recovered constellations are clean, except for a few symbols that lie slightly off the constellation. When converting back to bits the symbols are mapped to the nearest constellation point.

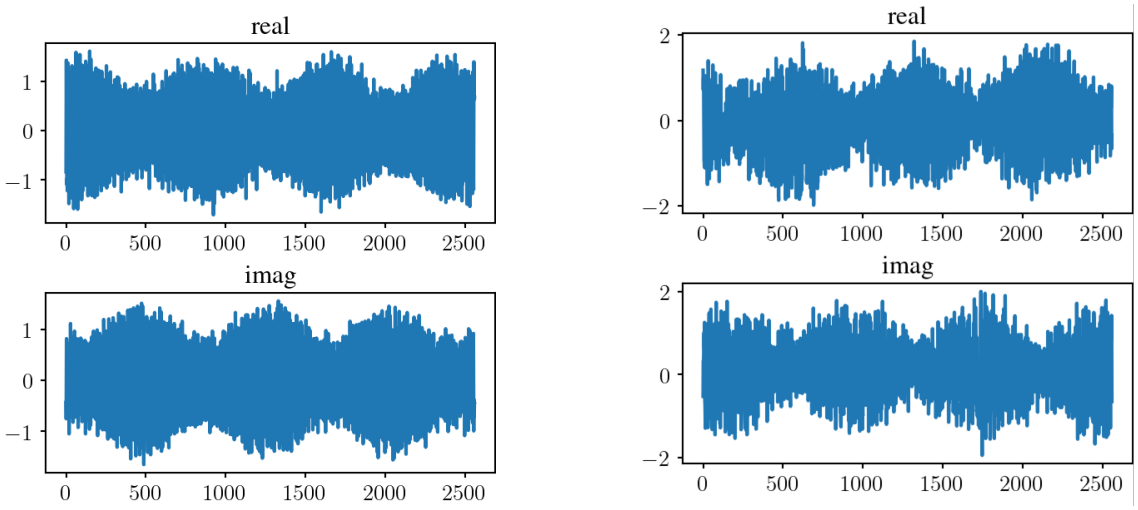


Figure 6.15.: **Left:** Ground truth CommSignal2 waveform from the dataset. **Right:** A generated CommSignal2 waveform from the learned diffusion model.

6.A.5 CommSignal2

The CommSignal2 dataset was recorded over-the-air and hence, unlike the synthetic datasets, we cannot probe it without knowledge of the true system parameters and signal generation model, which is not provided in (Lancho et al., 2024). These complications serve as one of our motivations to develop data-driven source separation models. In the real-world, CommSignal2 could interfere with an SOI such as a QPSK signal for which the signal generation model is known. A learned prior for CommSignal2 could help mitigate such interference.

As shown in Figure 6.15, we observe that the generated CommSignal2 waveforms generally display similar global temporal structure to the ground truth waveforms. To further demonstrate that our diffusion model has truly learned the statistical structure of the signal, we carried out source separation experiments and demonstrate that leveraging this diffusion model outperforms conventional and learning-based source separators.

6.B Analytical SOI Score

We used an RRC-QPSK signal as the SOI across all our source separation experiments. As detailed in Section 5.4.4, the score for a smoothened QPSK source can be analytically computed via Proposition 5.3, where it is more amenable to model and compute it in the symbol space (i.e., as i.i.d. symbols). Nevertheless, for the problem of separating the SOI from an interference source, we have to consider the components jointly in the time domain. Thus, we relate the time-domain representation to the symbols via $\mathbf{x} = \mathsf{H}\mathbf{a}$, where H represents the RRC filter matrix and $\mathbf{a}$ is a vector of symbols. To compute this analytical score, we smooth the symbols via a Gaussian smoothing model at noise level corresponding to timestep t,

$$\mathbf{x}_t = \mathsf{H}\mathbf{a}_t := \mathsf{H}(\mathbf{a} + \sigma_t \boldsymbol{\epsilon}_t).$$

With this relation, we express the score of the smoothened source as,

$$\begin{aligned}\nabla_{\mathbf{x}_t} \log p_{\mathbf{x}_t}(\mathbf{x}_t) &= \mathsf{H} \cdot \nabla_{\mathbf{a}_t} \log p_{\mathbf{a}_t}(\mathbf{a}_t) \\ &= \mathsf{H} \cdot \left(\frac{1}{\sigma_t^2} \left(-\mathbf{a}_t + \sum_{\mathbf{a} \in \mathcal{A}^K} \mathbf{a} \odot \phi_t \left(\mathbf{a}; \mathbf{a}_t \right) \right) \right) \qquad (6.7) \\ &\approx \frac{1}{\sigma_t^2} \left(-\mathbf{x}_t + \mathsf{H} \sum_{\mathbf{a} \in \mathcal{A}^K} \mathbf{a} \odot \phi_t \left(\mathbf{a}; \mathsf{H}^\dagger \mathbf{x}_t \right) \right). \qquad (6.8)\end{aligned}$$

where $\odot$ represents element-wise product, ϕ_t is the softmax-like operator defined in Eq. (5.19) that is applied element-wise here, and $\mathsf{H}^\dagger$ is the pseudo-inverse of H. Note that Eq. (6.7) is obtained by applying Eq. (5.18) to a vector of i.i.d. smoothened QPSK symbols. In our implementation, the estimate of these smoothened symbols $\mathbf{a}_t$ is obtained by reversing the RRC filter using $\mathsf{H}^\dagger$, as in Eq. (6.8).

On the other hand, our RRC-QPSK diffusion model is trained directly on the time-domain waveform, and can be used directly to separate the SOI waveform from the mixture without conversion to the symbol domain. This once again sheds light on the practicality of using data-driven methods to circumvent otherwise computationally challenging statistical modeling technical problems.

Part III.

Score-based One-Step Generative Modeling

7 Overview of One-Step Generative Modeling Techniques

Data-driven engineering and scientific systems often depend on large volumes of data to accurately model physical phenomena and calibrate systems for precise measurements in downstream tasks. In data-scarce environments, the ability to generate synthetic data becomes critical for system design and performance.

Consider, for instance, the SBI framework discussed in Section 1.1.2. In this approach, samples of ground truth signals and corresponding measurements are generated via simulation. A posterior sampler, typically implemented as a neural network, is then trained to invert the simulator. Ideally, this sampler should be both fast and capable of providing real-time signal recovery and uncertainty estimates.

However, in such settings, using diffusion models for sampling is often prohibitively expensive. This challenge becomes even more pronounced in high-dimensional settings, where each evaluation of the score function incurs a significant computational cost. For example, generating a single low-resolution image can take several seconds on a modest GPU.

If the ultimate goal is to train a high-fidelity neural sampler, simulating the entire reverse diffusion process becomes impractical for many real-world applications. As discussed in Section 1.2, implicit models—such as generative adversarial networks (GANs) or implicit autoencoders like those introduced in Chapter 4—offer a compelling alternative. These models define a one-step generator, enabling fast and efficient sampling.

In this chapter, we present an overview of existing one-step generative models, with a focus on the inherent strengths and weaknesses of different approaches.

7.1 Generative Adversarial Network

The most prominent approach in training implicit generative models is the generative adversarial network (GAN) (Goodfellow et al., 2014). In its most standard and widely used form, it alternates between the gradient steps of discriminator and generator training. As shown in Figure 7.1, the generator $\mathbf{g}_\theta : \mathbb{R}^d \to \mathbb{R}^D$ is map between a tractable noise distribution $q(\mathbf{z})$, typically chosen to be a uniform noise distribution defined over a $[0,1]^d$ or a standard multivariate normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, and a generated distribution $q_\theta(\mathbf{x})$ where $\mathbf{x} = \mathbf{g}_\theta(\mathbf{z})$.

In its original form[1], the discriminator plays an adversarial game with the generator and

[1]This is informally known as the "vanilla GAN"

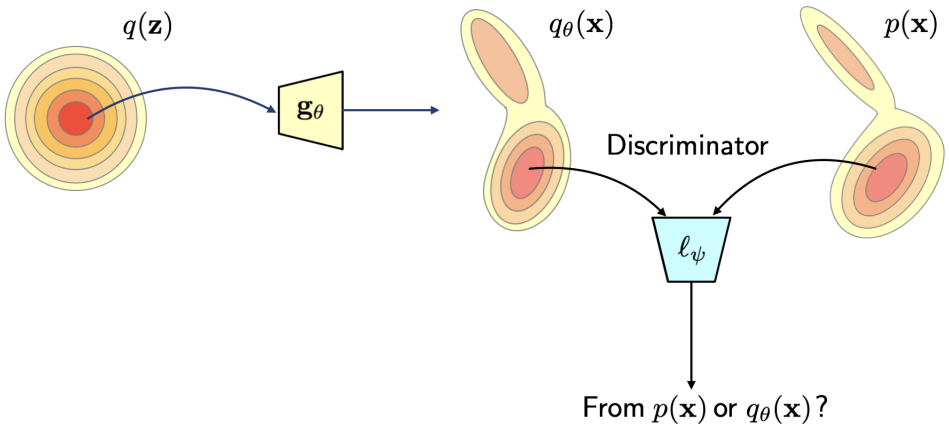


Figure 7.1.: A GAN consists of a one-step generator and a discriminator that are trained in alternating fashion.

given an input $\mathbf{x}$, it is trained to predict with high confidence whether the sample is drawn from the generated or data distribution by solving,

$$\min_\psi -\mathbb{E}_{p(\mathbf{x})}[\log D_\psi(\mathbf{x})] - \mathbb{E}_{q_\theta(\mathbf{x})}[\log(1 - D_\psi(\mathbf{x}))]\}.$$

In the non-parametric limit, the optimal discriminator for each θ is

$$D^*(\mathbf{x}) = \frac{p(\mathbf{x})}{p(\mathbf{x}) + q_\theta(\mathbf{x})} = \frac{1}{1 + r^{-1}(\mathbf{x})},$$

where the density ratio is defined as $r(\mathbf{x}) := \frac{p(\mathbf{x})}{q_\theta(\mathbf{x})}$. In practice, the discriminator training is implemented as

$$\min_\psi \ \mathbb{E}_{p(\mathbf{x})}[\mathsf{sp}(-\ell_\psi(\mathbf{x}))] + \mathbb{E}_{q_\theta(\mathbf{x})}[\mathsf{sp}(\ell_\psi(\mathbf{x}))]\}, \tag{7.1}$$

where $\mathsf{sp}(y) := \log(1+e^y)$ denotes the softplus function and $\ell_\psi(\mathbf{x})$ models the log-density-ratio. With a slight abuse of terminology we will also refer to $\ell_\psi(\mathbf{x})$ as the discriminator.

The generator objective is the opposite of the discriminator's as it is trained to generate samples that are indistinguishable from the true samples. Hence, to the train the generator the discriminator loss is maximized,

$$\max_\theta \ \mathbb{E}_{p(\mathbf{x})}[\mathsf{sp}(-\ell_\psi(\mathbf{x}))] + \mathbb{E}_{q_\theta(\mathbf{x})}[\mathsf{sp}(\ell_\psi(\mathbf{x}))]\},$$

or equivalently

$$\min_\theta \mathbb{E}_{q_\theta(\mathbf{x})}[-\mathsf{sp}(\ell_\psi(\mathbf{x}))].$$

This is the so called *saturating* version of the GAN loss as it is empirically observed to result in the gradient vanishing as training progresses. Instead, many successful GAN models adopt the *non-saturating* version that results in the same optimal solution but is slightly more stable in practice,

$$\min_\theta \mathbb{E}_{q_\theta(\mathbf{x})}[\mathsf{sp}(-\ell_\psi(\mathbf{x}))].$$

This so-called adversarial training can be understood as minimizing the Jensen–Shannon divergence (JSD) with the help of discriminator, via the variational characterization of JSD, where the JSD is defines as

$$\mathrm{D}_{\mathrm{JSD}}(p\|q_\theta) = \frac{1}{2}\mathrm{D}_{\mathrm{KL}}\left(p\middle\|\frac{p+q_\theta}{2}\right) + \frac{1}{2}\mathrm{D}_{\mathrm{KL}}\left(q_\theta\middle\|\frac{p+q_\theta}{2}\right). \tag{7.2}$$

Despite the popularity of GANs, training them is challenging. Although various techniques have been proposed to regularize the GAN objective—through alternatives to JSD (Arjovsky et al., 2017; Mao et al., 2017; Nowozin et al., 2016), novel regularizers (Miyato et al., 2018), and specialized network architectures (Brock et al., 2019; Karras et al., 2021; Sauer et al., 2022)—the discriminator training remains unstable. This has sparked increasing interest in developing new objectives for training generative models which we briefly discuss below.

7.2 Diffusion Distillation

The goal of diffusion distillation is to distill a teacher diffusion model into a student model that can generate high-fidelity samples in few steps and ideally in a single step similar to GANs.

The earliest works on distillation such as progressive distillation (Salimans and Ho, 2022) train a student diffusion model with drastically reduced sampling budget to match the performance of a teacher model that is simulated in reverse. For example, given a teacher diffusion model parametrized as a denoiser $\mathbf{f}_\phi$ and a noisy sample $\mathbf{x}_t$, a "clean" target $\mathbf{x}_\phi^{(k)}$ is constructed by running the teacher model for k steps in reverse. The student denoiser $\mathbf{f}_\theta$ is then trained by minimizing,

$$\min_\theta \mathbb{E}_{p(\mathbf{x})p(\mathbf{z})p(t)}[w(t)\|\mathbf{f}_\theta(\mathbf{x}_t;t) - \mathbf{x}_\phi^{(k)}\|^2].$$

More recently a class of new diffusion distillation techniques grounded in reverse KL divergence (KLD) minimization have gained popularity. DiffInstruct (Luo et al., 2024a), DMD (Yin et al., 2024b) and DMD2 (Yin et al., 2024a) all train a one-step generator $\mathbf{g}_\theta$ mapping noise $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}_D)$ to generated samples by minimizing $D_{\mathrm{KL}}(q_\theta\|p)$. The gradient of this objective is,

$$\nabla_\theta \mathrm{D}_{\mathrm{KL}}(q_\theta\|p) = \mathbb{E}_{q(\mathbf{z})}[\nabla_\theta \mathbf{g}_\theta(\mathbf{z})(\nabla_{\mathbf{x}} \log q_\theta(\mathbf{x}) - \log p(\mathbf{x}))|_{\mathbf{x}=\mathbf{g}_\theta(\mathbf{z})}],$$

which depends on the difference between the scores of the generated samples evaluated with score model of the fake samples (the fake score) and the data score (the true score). As we did in Chapter 5, the score of a noisy distribution can be obtained from a diffusion model. Hence, the aforementioned schemes minimize the divergence at multiple different noise levels to obtain a gradient update that leverages the difference between scores of noisy distributions,

$$\nabla_\theta \mathrm{D}_{\mathrm{KL}}^{\mathsf{avg}}(q_\theta\|p) := \mathbb{E}_{p(t)}[\nabla_\theta \mathrm{D}_{\mathrm{KL}}^{\mathsf{avg}}(q_{\theta,t}\|p_t)].$$

Here the reverse KLD between the noisy distributions is

$$\nabla_\theta \mathrm{D}_{\mathrm{KL}}^{\mathsf{avg}}(q_{\theta,t}\|p_t) = \mathbb{E}_{q(\mathbf{z})q(\boldsymbol{\epsilon})}[\nabla_\theta \mathbf{g}_\theta(\mathbf{z})(\mathbf{s}_{q_\theta}(\mathbf{x}_t) - \mathbf{s}_p(\mathbf{x}_t))\ |_{\mathbf{x}=\mathbf{g}_\theta(\mathbf{z})}], \tag{7.3}$$

where $\mathbf{s}_p(\mathbf{x}_t) = \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$ and $\mathbf{s}_{q_\theta}(\mathbf{x}_t) = \nabla_{\mathbf{x}_t} \log q_\theta(\mathbf{x}_t)$.

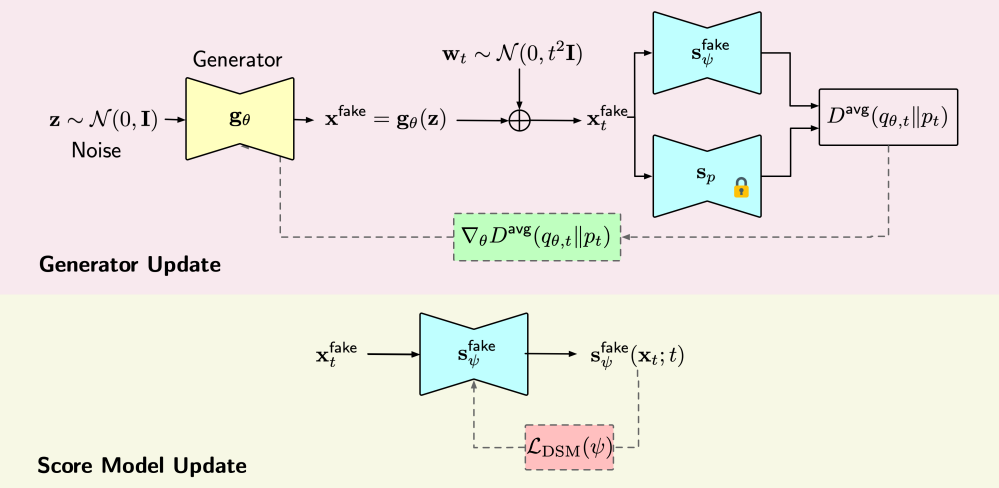


Figure 7.2.: Overview of reverse KL-based distillation techniques. **Top:** To update the generator, they compute the gradient of the reverse KLD on noisy fake samples with the fake score model using Eq. (7.3). **Bottom:** The fake score model is updated by computing the score of the fake noisy samples.

7.2.1 Training and Practical Implementation

In practice, Eq. (7.3) is implemented using a minibatch of generated samples. Each sample in the minibatch has a random noise level asociated with it. Thus, the scale of the gradients will differ across noise levels. Assuming that the diffusion model was parametrized as a denoiser (from which the score can be obtained using Tweedie's formula in Eq. (2.8)) distillation methods such as DMD scale the gradient and express it in terms of a pretrained denoiser $\mathbf{f}_\phi$ and a denoiser for the fake samples $\mathbf{f}_\psi$,

$$\nabla_\theta \mathcal{L}_{\text{DMD}}(\theta) = \mathbb{E}_{q(\mathbf{z})p(t)q(\boldsymbol{\epsilon})}[w_{\text{DMD}}(\mathbf{x}_t, \mathbf{x}, t)\nabla_\theta \mathbf{g}_\theta(\mathbf{z})(\mathbf{f}_\psi(\mathbf{x}_t; t) - \mathbf{f}_\phi(\mathbf{x}_t; t)) \mid_{\mathbf{x}=\mathbf{g}_\theta(\mathbf{z})}],$$

where an adaptive weight is used to ensure that the scale of the gradient is roughly uniform across noise levels,

$$w_{\text{DMD}}(\mathbf{x}_t, \mathbf{x}, t) := \frac{1}{\|\mathbf{x} - \mathbf{f}_\phi(\mathbf{x}_t; t)\|_1}. \tag{7.4}$$

Training is performed in alternating fashion. As shown in Figure 7.2, the generator is updated by computing the difference between scores with the true score coming from a pretrained diffusion model and the fake score estimated from samples synthesized by the generator. During generator training, the score models are kept frozen and their weights are not updated. Similarly during fake score training the generator weights are frozen.

Distillation is generally a stable training routine and thus is very appealing in practice. However, these techniques can still suffer pitfalls such as mode collapse, where the generator learns a fixed subset of modes to sample from. To mitigate mode collapse and enhance sample diversity, DMD employs an ODE-based regularizer by simulating the pretrained diffusion model in reverse. This process generates noise-image pairs, which are then used to further supervise the generator's training. However, collecting this dataset becomes prohibitively expensive for high-dimensional samples. To address this limitation, DMD2 introduces a

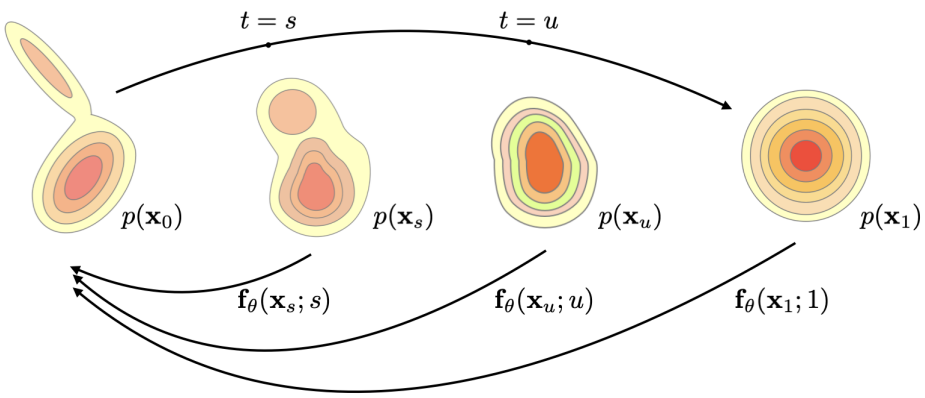


Figure 7.3.: A consistency function maps all points along the trajectory of the probability flow ODE back to the origin.

GAN-based regularizer, which effectively minimizes the Jensen-Shannon divergence alongside the reverse KLD, or a variant of the forward KLD when implemented in a non-saturating manner.

Several methods build upon the divergence minimization framework by introducing regularizers based on alternative statistical distance measures. For instance, Moment Matching Distillation (MMD) (Salimans et al., 2024), Score Identity Distillation (SiD) (Zhou et al., 2024), and Score Implicit Matching (SiM) (Luo et al., 2024b) align the fake score model with the pretrained score model using a variant of the Fisher divergence:

$$\mathcal{L}_{\text{Fisher}}(\psi) := \mathbb{E}_{q_\theta(\mathbf{x})p(t)q(\boldsymbol{\epsilon})}[w'(t)\|\mathbf{f}_\psi(\mathbf{x}_t; t) - \mathsf{sg}[\mathbf{f}_\phi(\mathbf{x}_t; t)]\|^2].$$

Here sg stands for the stop gradient operator. Additionally, both SiD and SiM extend this approach to generator training by minimizing the Fisher divergence, which requires a computationally expensive gradient calculation through the entire score model. To address this, they employ statistical approximations to make these gradient computations more practical.

7.3 Consistency Models

Consistency models are a new class of generative models introduced by Song et al. (2023b) that learn a consistency function between all points along the trajectory of the probability flow ODE such that they are mapped back to the origin as shown in Figure 7.3. Concisely, given points along one such trajectory, $\mathbf{x}_t, t \in [\epsilon, 1]$, where $\mathbf{x}_1 \sim \mathcal{N}(0, \mathbf{I})$, the consistency function satisfies,

$$\mathbf{f}(\mathbf{x}_t, t) = \begin{cases} \mathbf{x} & \text{if } t = \epsilon \\ \mathbf{f}(\mathbf{x}_s, s) & s \in [\epsilon, 1] \end{cases}$$

Given the boundary condition at the origin, the consistency function can be parametrized using a neural network similar to existing popular architectures such as EDM (see Section 4.B.2),

$$\mathbf{f}_\theta(\mathbf{x}_t, t) = \frac{\sigma_{\text{data}}^2}{(\sigma_t - \sigma_\epsilon)^2 + \sigma_{\text{data}}^2}\mathbf{x}_t + \frac{(\sigma_t - \sigma_\epsilon) \cdot \sigma_{\text{data}}}{\sqrt{\sigma_t^2 + \sigma_{\text{data}}^2}}\mathbf{g}_\theta(\mathbf{x}_t; t).$$

Given a noisy sample $\mathbf{x}_t = \mathbf{x} + \sigma_t \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$, first a single step of the probability flow ODE is simulated using the Euler sampler by running one step of sampling using Eq. (2.11),

$$\mathbf{x}_s = \mathbf{x}_t + (t - s) t \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$$

This can be computed using either a pretrained score model or via a single sample Monte-Carlo estimate. In the latter setting, it is important that the timesteps s and t are very close to each other for the approximation to hold. In consistency distillation a pretrained score model $\mathbf{s}_\phi$ is available and a single sampling step along the PF-ODE is simulated as

$$\mathbf{x}_s^\phi = \mathbf{x}_t + (t - s) t \mathbf{s}_\phi(\mathbf{x}_t; t).$$

Then the consistency function is learned by minimizing

$$\mathcal{L}_{\text{CD}}(\theta) = \mathbb{E}_{p(\mathbf{x})q(\boldsymbol{\epsilon})p(t)}[w(t) d(\mathbf{f}_\theta(\mathbf{x}_t; t), \mathsf{sg}[\mathbf{f}_\theta(\mathbf{x}_{t-\Delta t}^\phi; t - \Delta t)])],$$

where d is some distance measure, $w(t)$ is some positive weighting function and $s = t - \Delta t$, with Δt some fixed timestep difference. Song et al. (2023b) initially proposed using the LPIPS distance but subsequent works (Geng et al., 2024; Song and Dhariwal, 2024) have shown that similar performance can be achieved by using the ℓ_2 distance or a pseudo-Huber norm.

7.3.1 Consistency Training

Unlike distillation techniques, consistency models can also be trained from scratch. Assume that $s = t - \delta t, \delta t \to 0$. Then, the sampling step can be approximated using Tweedie's formula,

$$\begin{aligned} \mathbf{x}_s &\approx \mathbf{x}_t + (t - s) \frac{\mathbf{x} - \mathbf{x}_t}{t} \\ &= \mathbf{x} + s\boldsymbol{\epsilon}. \end{aligned}$$

Thus, the consistency function can now be learned by minimizing,

$$\mathcal{L}_{\text{CT}}(\theta) = \mathbb{E}_{p(\mathbf{x})q(\boldsymbol{\epsilon})p(t)}[w(t) d(\mathbf{f}_\theta(\mathbf{x} + t\boldsymbol{\epsilon}; t), \mathsf{sg}[\mathbf{f}_\theta(\mathbf{x} + (t - \delta t)\boldsymbol{\epsilon}; t - \delta t)])],$$

Consistency distillation still lags behind distillation methods based on reverse KL minimization, but consistency training often demonstrates more impressive results. However, consistency training is still inherently unstable and requires careful design of both the noise schedule due to limiting nature of δt and distance measure (Geng et al., 2024; Song and Dhariwal, 2024). Stabilizing and making this objective simpler is the focus of a lot of current research in the area.

7.4 Drawbacks and Outlook

Table 7.1 provides an overview of the various one-step and score-based generative models discussed in this chapter. Each method has its pros and cons. Both GANs and consistency

Table 7.1.: Comparison of different generative modeling techniques capable of high-quality sample generation.

Generative models	**Training idea**	**Generation**	**Training stability**	**Require pretrained model?**
GAN	minimizing JSD, with discriminator	one-step	unstable	N
Diffusion models	training multi-noise-level denoisers via DSM	multi-step	stable	N
Diffusion distillation	(mostly) minimizing reverse KLD (in DMD)	{one,few}-step	stable	Y
Consistency distillation	simulating trajectories of probability flow ODE	{one,few}-step	stable	Y
Consistency training			unstable	N

training are attractive options for training a one-step generative model from scratch, but they suffer from instabilities during training. Meanwhile schemes based on distillation are stable but require access to a pre-trained diffusion model which could require several days or weeks to train to optimality. Thus, bridging these different methods is of great interest and in the next chapter we will develop a new framework for simple, stable anf efficient one-step generative model training from scratch.

8 Stable and Efficient Generative Modeling with Score-of-Mixture Training

In the last chapter we introduced several popular one-step generative modeling frameworks. It was evident that each framework has its pros and cons but that in general fast one-step sampling comes at a cost. It either comes with training instabilities or via constraints on elaborate pre-training or access to a pre-trained model.

In this chapter, we tackle the problem of training high-quality one-step generative models more directly, i.e., *without* simulating an iterative reverse diffusion process for sampling or leveraging a pretrained diffusion model during training. Starting from first principles of statistical divergence minimization, we show that a high-quality one-step generative model can be trained from scratch in a stable manner, via the multi-noise-level DSM technique used in diffusion models.

The proposed framework achieves the best of several worlds, thus addressing drawbacks of existing one-step generative models: (1) a new, simple statistical divergence minimization framework without probability paths of ODE (like GAN), (2) stable training using denoising score matching (like diffusion models), (3) training from scratch without a pretrained diffusion model (like consistency models), and (4) near state-of-the-art one-step image generative performance (like GAN and consistency models). We also demonstrate that the proposed method can be extended to distill from a pretrained diffusion model, and can achieve performance similar to state-of-the-art methods for the same.

Finally, we show that the framework can be naturally extended to multi step generative modeling, similar to consistency models, offering practitioners greater flexibility and control over the generation process.

8.1 Score-of-Mixture Training

In this section, we introduce a new framework for generative modeling called *Score-of-Mixture Training* (SMT). We describe how to efficiently train one-step generative models from scratch, i.e.,, without a pretrained diffusion model. The key ingredient of this framework is distribution matching using a new family of statistical divergences, whose gradient can be approximated by estimating the score of mixture distributions of real and fake distributions, hence the name Score of Mixture Training. We adopt the concept of multi-noise level learning from

diffusion models and propose multi-divergence minimization for stable training.

8.1.1 Minimizing α-Skew Jensen–Shannon Divergences

The crux of the new framework lies in minimizing a class of statistical divergences between $p(\mathbf{x})$ and $q_\theta(\mathbf{x})$ defined as

$$\mathrm{D}_{\mathrm{JSD}}^{(\alpha)}(q_\theta, p) := \frac{1}{\alpha}\mathrm{D}_{\mathrm{KL}}(q_\theta \parallel \alpha p + (1-\alpha)q_\theta) + \frac{1}{1-\alpha}\mathrm{D}_{\mathrm{KL}}(p \parallel \alpha p + (1-\alpha)q_\theta)$$

for some $\alpha \in (0,1)$, which we call the α-skew Jensen-Shannon divergence (α-JSD) (Nielsen, 2010). This divergence belongs to f-divergences (Csiszár et al., 2004).

Interestingly, α-skew JSD naturally interpolates between the forward Kullback–Leibler divergence (KLD) $\mathrm{D}_{\mathrm{KL}}(p \parallel q_\theta)$ (when $\alpha \to 0$), the standard definition of JSD (when $\alpha = \frac{1}{2}$), and the reverse KLD $\mathrm{D}_{\mathrm{KL}}(q_\theta \parallel p)$ (when $\alpha \to 1$). In contrast to the forward KLD and reverse KLD, the α-skew JSD with $\alpha \in (0,1)$ is well-defined even when there is a support mismatch in p and q_θ, which may be the case especially in the beginning of training.

Feature 1: Multi-Divergence Training. Hence, we propose to minimize a weighted sum of the α-JSD's for different α's, as divergences with different α's exploit different geometries between two distributions. For example, it is known that minimizing the forward and reverse KLD leads to mode-covering and mode-seeking behaviors, respectively, and we can enforce better support matching behavior by considering the entire range of α.

To minimize this family of divergences in practice, we consider its gradient expression:

Proposition 8.1. *Suppose that* $\mathbb{E}_{q_\theta(\mathbf{x})}[\nabla_\theta \log q_\theta(\mathbf{x})] = 0$.[1] *Then, we have*

$$\nabla_\theta D_{\mathrm{JSD}}^{(\alpha)}(q_\theta, p) = \frac{1}{\alpha}\mathbb{E}_{q(\mathbf{z})}\Big[\nabla_\theta \mathbf{g}_\theta(\mathbf{z})(\mathbf{s}_{\theta;0}(\mathbf{x}) - \mathbf{s}_{\theta;\alpha}(\mathbf{x}))\Big|_{\mathbf{x}=\mathbf{g}_\theta(\mathbf{z})}\Big] \tag{8.1}$$

where we define the score of the mixture distribution

$$\mathbf{s}_{\theta;\alpha}(\mathbf{x}) := \nabla_{\mathbf{x}} \log(\alpha p(\mathbf{x}) + (1-\alpha)q_\theta(\mathbf{x})).$$

This proposition suggests that we can update the generator $\mathbf{g}_\theta(\mathbf{z})$ using this gradient expression, provided that we can estimate the score of the mixture distribution $\mathbf{s}_{\theta;\alpha}(\mathbf{x})$.

Feature 2: Amortized Score Model. To implement this idea, in this paper, we propose to use an amortized score model $(\mathbf{x}, \alpha) \mapsto \mathbf{s}_\psi(\mathbf{x}; \alpha)$, to approximate the score of mixture $\mathbf{s}_{\theta;\alpha}(\mathbf{x})$. Through our experiments we show that learning the scores of mixture over different α's using a single model is effective and helps training. In Section 8.1.3, we explain how we can train the amortized score model $(\mathbf{x}, \alpha) \mapsto \mathbf{s}_\psi(\mathbf{x}; \alpha)$ using samples from $p(\mathbf{x})$ and $q_\theta(\mathbf{x})$.

[1] It is a standard assumption in the literature (Hyvärinen, 2005), which holds under a mild regularity assumption on the parametric model $q_\theta(\mathbf{x})$ so that $\int \nabla_\theta q_\theta(\mathbf{x})\,\mathrm{d}\mathbf{x} = \nabla_\theta \int q_\theta(\mathbf{x})\,\mathrm{d}\mathbf{x}$.

8.1.2 Learning with Multiple Noise Levels

To achieve stable training, we opt to minimize the divergence at different noise levels by considering the convolved distributions, $p_t := p * \mathcal{N}(\mathbf{0}, \sigma_t^2 \mathbf{I}_D)$ and $q_{\theta,t} := q_\theta * \mathcal{N}(\mathbf{0}, \sigma_t^2 \mathbf{I}_D)$. This idea is widely used in the existing distillation methods. We borrow the variance-exploding Gaussian noising process notation from Karras et al. (2022) where $\sigma_t \in [\sigma_{\min}, \sigma_{\max}]$. As we also integrate over different α's, the final objective becomes

$$\mathcal{L}_{\text{gen}}(\theta) := \mathbb{E}_{p(\alpha)p(t)}[\mathrm{D}_{\text{JSD}}^{(\alpha)}(q_{\theta,t}, p_t)], \tag{8.2}$$

where we will prescribe the choice of $p(\alpha)$ in Section 8.1.5. Similar to Eq. (8.1), the gradient of the divergence at noise level t can be approximated via the amortized score as

$$\begin{aligned} &\nabla_\theta \mathrm{D}_{\text{JSD}}^{(\alpha)}(q_{\theta,t}, p_t) \approx \boldsymbol{\gamma}_\psi(\theta; \alpha, t) \\ &:= \mathbb{E}_{q(\mathbf{z})q(\boldsymbol{\epsilon})}\left[\nabla_\theta \mathbf{g}_\theta(\mathbf{z}) \frac{\mathbf{s}_\psi(\mathbf{x}_t; 0, t) - \mathbf{s}_\psi(\mathbf{x}_t; \alpha, t)}{\alpha}\Big|_{\mathbf{x}=\mathbf{g}_\theta(\mathbf{z})}\right], \end{aligned} \tag{8.3}$$

where the amortized score model $\mathbf{s}_\psi(\mathbf{x}_t; \alpha, t)$, which is conditioned on the noise level t, is an estimate of $\mathbf{s}_{\theta;\alpha,t}(\mathbf{x}_t) := \nabla_{\mathbf{x}_t} \log(\alpha p(\mathbf{x}_t) + (1-\alpha) q_\theta(\mathbf{x}_t))$. We provide a practical implementation of the amortized score model as a small modification of a diffusion model architecture in Section 8.1.4. We remark in passing that this expression can be understood as a generalization of the gradient update of Eq. (7.3) used in the existing reverse-KLD-based distillation schemes.

Finally, we can then approximate the generator gradient as

$$\nabla_\theta \mathcal{L}_{\text{gen}}(\theta) \approx \mathbb{E}_{p(\alpha)p(t)}[\boldsymbol{\gamma}_\psi(\theta; \alpha, t)].$$

Importantly, similar to existing distillation methods, the gradient only involves the output of the score model, but not its gradient. This is beneficial since such extra gradient information requires expensive backpropagation through the score model to the generator (Zhou et al., 2024).

8.1.3 Estimating Score of Mixture Distributions

Estimating the score of the mixture distribution turns out to be as simple as minimizing a mixture of the score matching losses, as stated in the following proposition:

Proposition 8.2. *For any $\alpha \in [0, 1]$, the minimizer of the objective function*

$$\begin{aligned} \mathcal{L}(\psi; \alpha) = \alpha\, &\mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}_\psi(\mathbf{x}; \alpha) - \mathbf{s}_p(\mathbf{x})\|^2] \\ +(1-\alpha)\, &\mathbb{E}_{q_\theta(\mathbf{x})}[\|\mathbf{s}_\psi(\mathbf{x}; \alpha) - \mathbf{s}_{q_\theta}(\mathbf{x})\|^2] \end{aligned} \tag{8.4}$$

satisfies $\mathbf{s}_{\psi^*}(\mathbf{x}; \alpha) = \mathbf{s}_{\theta;\alpha}(\mathbf{x})$.

Since we train with multiple noise levels, we are interested in the marginal score of $\mathbf{x}_t = \mathbf{x} + \sigma_t \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ at some noise level σ_t. We can use denoising score matching (see Eq. (2.6)) to define an equivalent sample-only objective to learn the score using Tweedie's formula, as stated in the following proposition:

Proposition 8.3. *Let $\alpha \in [0,1]$ be fixed and σ_t be some fixed noise level. Then, the minimizer of the objective function*

$$\begin{aligned}\mathcal{L}_{score}(\psi;\alpha,t) := \alpha\,\mathbb{E}_{p(\mathbf{x})q(\boldsymbol{\epsilon})}[\|\mathbf{s}_\psi(\mathbf{x}_t;\alpha,t)+\boldsymbol{\epsilon}/\sigma_t\|^2] \\ +(1-\alpha)\,\mathbb{E}_{q_\theta(\mathbf{x})q(\boldsymbol{\epsilon})}[\|\mathbf{s}_\psi(\mathbf{x}_t;\alpha,t)+\boldsymbol{\epsilon}/\sigma_t\|^2].\end{aligned} \tag{8.5}$$

satisfies

$$\mathbf{s}_{\psi^*}(\mathbf{x}_t;\alpha,t) = \mathbf{s}_{\theta;\alpha,t}(\mathbf{x}_t).$$

Hence, to approximate $\mathbf{s}_{\theta;\alpha,t}(\mathbf{x})$ using the amortized score model $\mathbf{s}_\psi(\mathbf{x};\alpha,t)$, we can minimize

$$\mathcal{L}_{\text{score}}(\psi) := \mathbb{E}_{p(\alpha)p(t)}[\mathcal{L}_{\text{score}}(\psi;\alpha,t)].$$

In practice, we parametrize the score model in the form of a denoiser and reconstruct the score from the denoiser output via Tweedie's formula similar to diffusion models (see Eq. 2.8).

Feature 3: Leveraging Real and Fake Samples via Amortized Score Estimation. We remark that our score learning objective seamlessly utilizes both real and fake samples throughout the training, helping the generator better generalize. This is in contrast to some existing diffusion distillation methods, which introduce expensive regularizers to integrate real samples, or backpropagate through the pretrained score model (Salimans et al., 2024; Yin et al., 2024a,b).

8.1.4 Practical Design of Amortized Score Network

With an additional conditioning scheme to embed auxiliary information about α in addition to the noise level σ_t, any existing diffusion model backbone can be used to parameterize the amortized score network $\mathbf{s}_\psi(\mathbf{x};\alpha,t)$. Here, we describe how we can modify the popular UNet-based score architectures (Karras et al., 2022; Nichol and Dhariwal, 2021; Song et al., 2020) with minimal modifications.

First, drawing from the noise embedding sensitivity analysis by Song and Dhariwal (2024), we opt for a Fourier embedding $\mathbf{c}_\alpha$ with a default scale of 16. This choice ensures that the embedding is sufficiently sensitive to fluctuations in α, particularly during the early stage of training.

Then, we concatenate the α-embedding with the embedding of other auxiliary information (e.g., t and labels) and apply a single SiLU (Elfwing et al., 2018) activated linear layer:

$$\mathbf{c}_{\text{out}} = \mathsf{silu}(\mathsf{W}_{\text{aux}}\mathbf{c}_{\text{aux}} + \mathsf{W}_\alpha\mathbf{c}_\alpha).$$

The rationale behind this choice is as follows: as training progresses, the real and fake distributions begin to overlap, making it natural for the amortized score model to become less sensitive to α. Thanks to the additional linear layer W_α after the α-embedding $\mathbf{c}_\alpha$, this behavior can be realized when $\mathsf{W}_\alpha \approx \mathbf{0}$, when necessary.

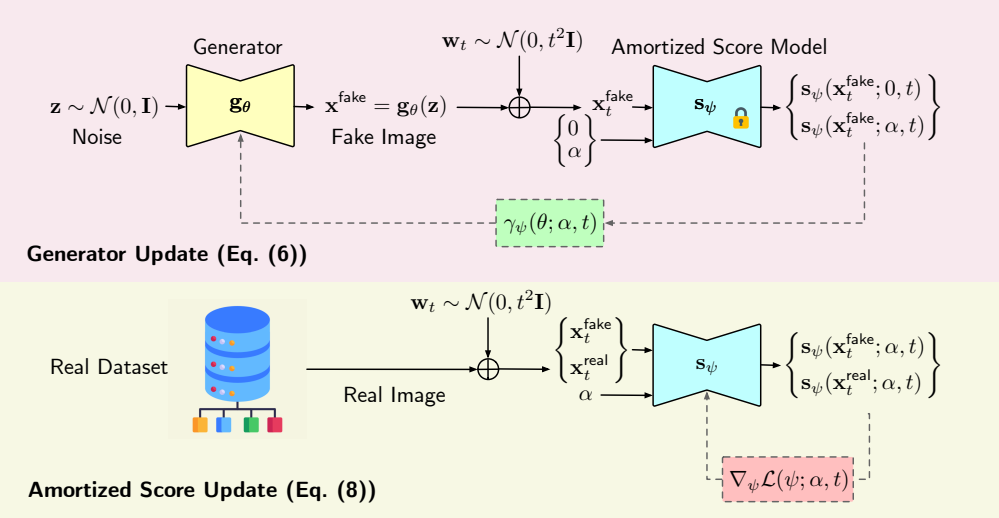


Figure 8.1.: Overview of SMT. **Top:** To update the generator, we compute the gradient of the α-JSD on noisy fake samples with the *frozen* amortized score model using Eq. (8.3). **Bottom:** The amortized score model is updated by computing the score of the mixture distribution on both fake and real noisy samples, and then updating the weights using the gradient in Eq. (8.5).

8.1.5 Training

Alternating Training. Our training scheme alternates between the score estimation with the score matching objective in Eq. (8.5), and the generator training with Eq. (8.3), where we plug-in $\mathbf{s}_\psi(\mathbf{x}_t; \alpha, t)$ in place of $\mathbf{s}_{\theta;\alpha,t}(\mathbf{x}_t)$. This is similar in spirit to GAN training, but the DSM technique in our framework in place of the discriminator training naturally stabilizes training. The overall training framework is summarized in Figure 8.1 and Algorithm 8.1 in Appendix 8.C.

Initialization. We warm up the generator with a standard denoising task as in diffusion models for several steps to better initialize the weights, as we empirically found that initializing the generator with pretrained weights from a denoiser significantly accelerated convergence. The amortized score network is randomly initialized.

Choice of $p(\alpha)$. The choice of $p(\alpha)$ is crucial in our framework. To train both the generator and score model, we sample α from a uniform distribution over 1000 equally spaced points in $[0, 1]$, ensuring a dense enough grid to generalize to any α. For score training, we further ensure that 25% of the sampled α's are zero, since this is always used in our gradient update; see Eq. (8.3).

Adaptive Weighting. In practice we compute the gradient with an adaptive weight $w(\mathbf{x}_t, \mathbf{x}, \alpha, t)$ to ensure that the scale of the gradient for each minibatch sample is roughly uniform for different values of α and t. Hence, we modify the generator gradient in Eq. (8.3)

as

$$\gamma_\psi^w(\theta;\alpha,t) := \mathbb{E}_{q(\mathbf{z})}\Big[\nabla_\theta \mathbf{g}_\theta(\mathbf{z})\times \Big\{w(\mathbf{x}_t,\mathbf{x},\alpha,t)\frac{\mathbf{s}_\psi(\mathbf{x}_t;0,t)-\mathbf{s}_\psi(\mathbf{x}_t;\alpha,t)}{\alpha}\Big\}\Big|_{\mathbf{x}=\mathbf{g}_\theta(\mathbf{z})}\Big], \tag{8.6}$$

where the weighting is defined as

$$w(\mathbf{x}_t,\mathbf{x},\alpha,t) := w_\alpha(\mathbf{x}_t,t)w_{\mathrm{DMD}}(\mathbf{x}_t,\mathbf{x},t). \tag{8.7}$$

Here w_{DMD} is the adaptive noise weighting introduced by (Yin et al., 2024b) (see Eq. (7.4) in Section 8.2) and $w_\alpha(\mathbf{x}_t,t)$ is a new weighting inspired by the pseudo-Huber norm (Geng et al., 2024; Song and Dhariwal, 2024)

$$w_\alpha(\mathbf{x}_t,t) := \alpha\sqrt{\frac{\|\mathbf{s}_\psi(\mathbf{x}_t;0,t)-\mathbf{s}_\psi(\mathbf{x}_t;1,t)\|^2}{\|\mathbf{s}_\psi(\mathbf{x}_t;0,t)-\mathbf{s}_\psi(\mathbf{x}_t;\alpha,t)\|^2}}.$$

This weighting still preserves the limiting forward KLD behavior of the objective as $\alpha\to 0$ and simplifies to DMD gradient when $\alpha=1$. We empirically show the efficacy of our adaptive weighting term $w_\alpha(\mathbf{x}_t,t)$ through ablation studies on the CIFAR-10 dataset in Section 8.3.3; see Figure 8.3b.

Regularization with GAN. We empirically found that a GAN-type regularization can accelerate convergence even further in the beginning of training. More concretely, we can train the discriminator $\ell_\psi(\mathbf{x}_t;t)\approx\log\frac{p(\mathbf{x})}{q_\theta(\mathbf{x})}$ by the GAN discriminator training in Eq. (7.1). In our implementation, we opt to train a discriminator using a variant based on the α-JSD. Given a discriminator $\ell_\psi(\mathbf{x}_t;t)$, we minimize a *non-saturating version* of the α-JSD loss,

$$\mathcal{L}_{\mathrm{GAN}}^{(\alpha,t)}(\theta) = \mathbb{E}_{q_\theta(\mathbf{x}_t)}\Big[\mathsf{sp}\Big(-\ell_\psi(\mathbf{x}_t;t)-\log\frac{\alpha}{1-\alpha}\Big)\Big]. \tag{8.8}$$

Similar to Yin et al. (2024a), we parameterized the discriminator by a stack of convolution layers, applied on top of an intermediate feature of the amortized score network at $\alpha=1/2$.

8.2 Score-of-Mixture Distillation

In our development so far, we do not assume access to a pretrained diffusion model. In this section, we show how a practitioner can train a one-step generative model leveraging a pretrained diffusion model, if available, within our framework. The proposed distillation scheme is comparable or even outperforms the state-of-the-art distillation schemes.

8.2.1 How To Leverage Pretrained Diffusion Model

In the distillation setup, we treat the pretrained diffusion model as the data score $\mathbf{s}_p(\mathbf{x}_t;t)$, and thus training the score of mixture $\mathbf{s}_{\theta;\alpha}(\mathbf{x}_t;t)$ using a single, amortized model may not be the most efficient parameterization. Hence, instead, we consider an alternative parametrization, as guided by the following statement:

Proposition 8.4. *Let $\alpha \in [0,1]$, $\mathbf{s}_p(\mathbf{x})$ be the data score, $\mathbf{s}_{q_\theta}(\mathbf{x})$ be the score of the generated samples. Then, the score of the mixture distribution can be expressed as*

$$\mathbf{s}_{\theta;\alpha}(\mathbf{x}) = D_{\theta;\alpha}(\mathbf{x})\mathbf{s}_p(\mathbf{x}) + (1 - D_{\theta;\alpha}(\mathbf{x}))\mathbf{s}_{q_\theta}(\mathbf{x}), \tag{8.9}$$

where

$$D_{\theta;\alpha}(\mathbf{x}) := \sigma\Big(\log \frac{p(\mathbf{x})}{q_\theta(\mathbf{x})} + \log \frac{\alpha}{1-\alpha}\Big), \tag{8.10}$$

In words, we can express the score of mixture $\mathbf{s}_{\theta;\alpha}(\mathbf{x})$ as a mixture of scores $\mathbf{s}_p$ and $\mathbf{s}_{q_\theta}$, where the weight is $(D_{\theta;\alpha}(\mathbf{x}), 1 - D_{\theta;\alpha}(\mathbf{x}))$. This suggests that instead of an amortized modeling of the score of mixture, we can use an alternative parameterization,

$$\mathbf{s}^{\mathsf{exp}}_\psi(\mathbf{x};\alpha) := D_\psi(\mathbf{x};\alpha)\mathbf{s}_p(\mathbf{x}) + (1 - D_\psi(\mathbf{x};\alpha))\mathbf{s}^{\mathsf{fake}}_\psi(\mathbf{x}),$$

where

$$D_\psi(\mathbf{x};\alpha) := \sigma\Big(\ell_\psi(\mathbf{x}) + \log \frac{\alpha}{1-\alpha}\Big).$$

Here, we can parameterize the discriminator $\mathbf{x} \mapsto \ell_\psi(\mathbf{x})$ in the same way as we do for the GAN discriminator (see Section 7.1).

We can extend this to multiple noise levels easily. Hence, an alternative parameterization for $\mathbf{s}_{\theta;\alpha}(\mathbf{x}_t;t)$ is

$$\begin{aligned}\mathbf{s}^{\mathsf{exp}}_\psi(\mathbf{x}_t;\alpha,t) := {} & D_\psi(\mathbf{x}_t;\alpha,t)\mathbf{s}_p(\mathbf{x}_t;t) \\ & + (1 - D_\psi(\mathbf{x}_t;\alpha,t))\mathbf{s}^{\mathsf{fake}}_\psi(\mathbf{x}_t;t),\end{aligned} \tag{8.11}$$

where

$$D_\psi(\mathbf{x}_t;\alpha,t) := \sigma\Big(\ell_\psi(\mathbf{x}_t;t) + \log \frac{\alpha}{1-\alpha}\Big). \tag{8.12}$$

Plugging this explicit score model into Eq. (8.5), we can learn both the fake score model $\mathbf{s}^{\mathsf{fake}}_\psi$ and the discriminator ℓ_ψ at different noise levels.

Corollary 8.5. *Let $\alpha \in [0,1]$ be fixed and σ_t be some fixed noise level. Then, the minimizer of the objective function*

$$\begin{aligned}\mathcal{L}^{\mathsf{exp}}(\psi;\alpha,t) := {} & \alpha\, \mathbb{E}_{p(\mathbf{x})q(\boldsymbol{\epsilon})}[\|\mathbf{s}^{\mathsf{exp}}_\psi(\mathbf{x}_t;\alpha,t) + \boldsymbol{\epsilon}/\sigma_t\|^2] \\ & + (1-\alpha)\, \mathbb{E}_{q_\theta(\mathbf{x})q(\boldsymbol{\epsilon})}[\|\mathbf{s}^{\mathsf{exp}}_\psi(\mathbf{x}_t;\alpha,t) + \boldsymbol{\epsilon}/\sigma_t\|^2]\end{aligned} \tag{8.13}$$

satisfies

$$\mathbf{s}^{\mathsf{fake}}_{\psi^*}(\mathbf{x};t) = \mathbf{s}_{q_\theta}(\mathbf{x};t) \quad \textit{and} \quad \ell_{\psi^*}(\mathbf{x}_t;t) = \log \frac{p(\mathbf{x}_t)}{q_\theta(\mathbf{x}_t)}.$$

We remark that this new regression objective in Eq. (8.13) provides a new way to compute the log density ratio, as an alternative to the GAN training. In Appendix 8.B, we establish a connection between this objective for training a discriminator to an existing GAN discriminator objective in the literature.

With this new, explicit parameterization, we can approximate the gradient expression in Eq. (8.3) as

$$\begin{aligned}\nabla_\theta \mathrm{D}^{(\alpha)}_{\mathrm{JSD}}(q_{\theta,t}, p_t) &\approx \boldsymbol{\gamma}^{\mathsf{exp}}_\psi(\theta;\alpha,t) \\ &:= \mathbb{E}_{q(\mathbf{z})}\Big[D_\psi(\mathbf{x}_t;\alpha,t)\times \\ &\qquad \nabla_\theta \mathbf{g}_\theta(\mathbf{z})\frac{\mathbf{s}^{\mathsf{fake}}_\psi(\mathbf{x}_t;t)-\mathbf{s}_p(\mathbf{x}_t,t)}{\alpha}\Big|_{\mathbf{x}=\mathbf{g}_\theta(\mathbf{z})}\Big].\end{aligned} \tag{8.14}$$

8.2.2 Implementation and Training

Model Architectures. We can leverage any existing diffusion model architectures directly for the fake score $\mathbf{s}^{\mathsf{fake}}_\psi(\mathbf{x}_t;t)$. We parametrize the discriminator $\ell_\psi(\mathbf{x}_t;t)$ similar to the noise-conditional discriminator in our training from scratch setting (see Section 8.1.5). The difference is that we can train the discriminator by minimizing the DSM loss in Eq. (8.13) naturally, without an additional GAN loss. When training the generator, we plug in this approximate log density ratio into Eq. (8.8) to regularize the generator updates.

Training. We also train in an alternating fashion. Since we have access to a pretrained score model, we use this to initialize the weights of both the generator and the fake score model. We utilize the same sampling distribution for α as in our training from scratch setup (see Section 8.1.5). The procedure is summarized in Figure 8.2 and Algorithm 8.2 in Appendix 8.C.

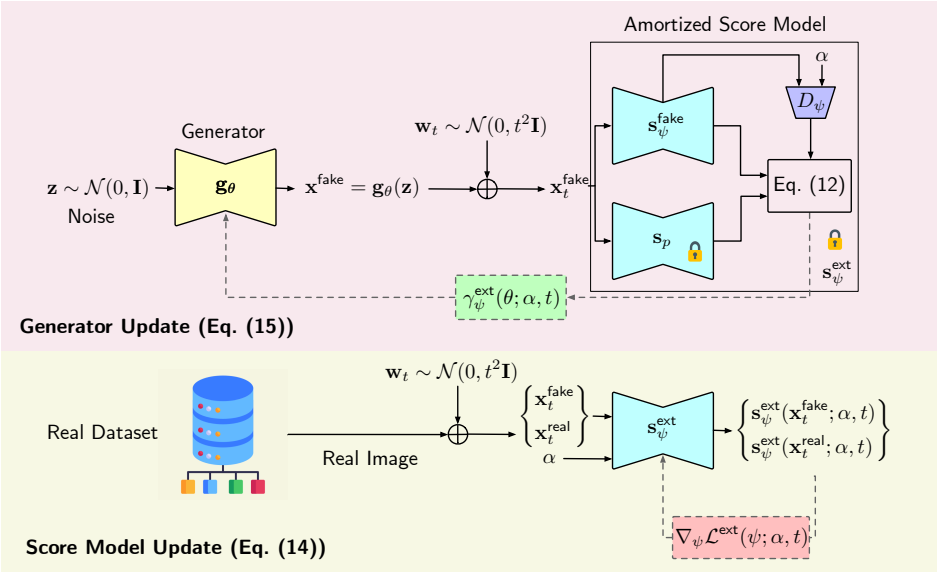


Figure 8.2.: Overview of Score-of-Mixture Distillation. **Top:** To update the generator weights, the fake image is diffused at noise level t and then used to compute the gradient of the α-skew divergence with the explicitly parametrized amortized score model using Eq. (8.14). **Bottom:** Amortized score model training involves computing the score of the mixture distribution on both fake and real samples diffused with noise level t and then updating the weights using the gradient of Eq. (8.13).

Table 8.1.: Image generation results on ImageNet 64x64 (class-conditional) and CIFAR-10 32x32 (unconditional). The size of the sampler is denoted by the number of parameters (# params), and NFE stands for the Number of Function Evaluations. The best FIDs from each category are highlighted in bold, and our methods **SMT** and **SMD** are highlighted with a blue shade.

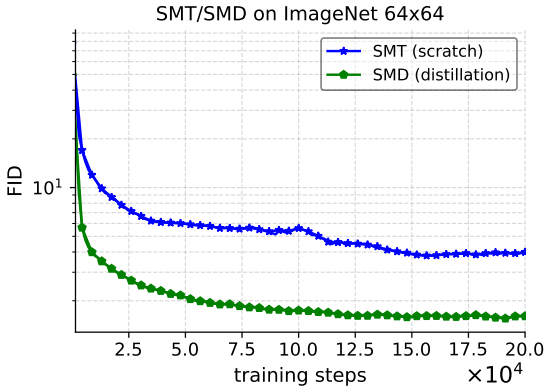
	ImageNet 64x64			CIFAR-10 32x32		
Method	# params	NFE	FID↓	# params	NFE	FID↓
Training from scratch: Diffusion models						
DDPM (Ho et al., 2020)	-	-	-	56M	1000	3.17
ADM (Dhariwal and Nichol, 2021)	296M	250	2.07	-	-	-
EDM (Karras et al., 2022)	296M	512	**1.36**	56M	35	**1.97**
Training from scratch: One-step models						
CT (Song et al., 2023b)	296M	1	13.0	56M	1	8.70
iCT (Song and Dhariwal, 2024)	296M	1	4.02	56M	1	2.83
iCT-deep (Song and Dhariwal, 2024)	592M	1	3.25	112M	1	**2.51**
ECT (Geng et al., 2024)	280M	1	5.51	56M	1	3.60
SMT (ours)	296M	1	**3.23**	56M	1	3.13
Diffusion distillation						
PD (Salimans and Ho, 2022)	296M	1	10.7	60M	1	9.12
TRACT (Berthelot et al., 2023)	296M	1	7.43	56M	1	3.78
CD (LPIPS) (Song et al., 2023b)	296M	1	6.20	56M	1	4.53
Diff-Instruct (Luo et al., 2024a)	296M	1	5.57	56M	1	4.53
MultiStep-CD (Heek et al., 2024)	1200M	1	3.20	-	-	-
DMD w/o reg (Yin et al., 2024b)	296M	1	5.60	56M	1	5.58
DMD2 w/ GAN (Yin et al., 2024a)	296M	1	1.51	56M	1	2.43
MMD (Salimans et al., 2024)	400M	1	3.00	-	-	-
SiD (Zhou et al., 2024)	296M	1	1.52	56M	1	**1.92**
SiM (Luo et al., 2024b)	-	-	-	56M	1	2.02
SMD (ours)	296M	1	**1.48**	56M	1	2.22
w/ expensive regularizer or finetuning						
CTM (Kim et al., 2024)	296M	1	1.92	56M	1	**1.98**
DMD w/ reg (Yin et al., 2024b)	296M	1	2.62	56M	1	2.66
DMD2 (finetuned) (Yin et al., 2024a)	296M	1	**1.23**	-	-	-

8.3 Experiments

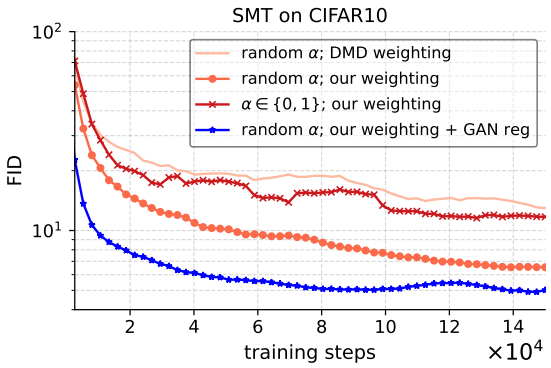
In this section, we first present results on the ImageNet 64×64 dataset. We then demonstrate the competitiveness of our method on the CIFAR-10 dataset and conduct a series of ablation studies. We measure performance through sample quality as measured by the Fréchet Inception Distance (FID) (Heusel et al., 2017). The exact hyerparameters, training configurations used and additional results can be found in Appendix 8.D.

8.3.1 Class-conditional ImageNet 64x64 Generation

Experimental Setup. We trained class-conditional one-step generative models on ImageNet 64×64 (Deng et al., 2009), experimenting with both distillation and training from scratch. In both cases, we used the ADM architecture (Nichol and Dhariwal, 2021) with EDM preconditioning (see Section 4.B.2) as the base score model architecture, and the discriminator



(a) ImageNet 64×64 (scratch and distillation).



(b) CIFAR-10 with ablation studies (scratch).

Figure 8.3.: FID evolution with training.

Figure 8.4.: Samples from SMT on ImageNet 64×64. Each row represents a unique class. Additional samples can be found in Appendix 8.D.3.

$\ell_\psi(\mathbf{x}_t; t)$ was implemented as a stack of convolution layers operating on the bottleneck feature from the score network, similar to DMD2 (Yin et al., 2024a). For training from scratch, we augmented the score architecture using an α-embedding as described in Section 8.1.4. The total number of parameters of the amortized score model remained unchanged otherwise. As a warmup stage, we pretrained the generator on the dataset using a standard diffusion denoising objective for 40k steps to initialize the weights. For distillation, we used a pretrained diffusion model from (Karras et al., 2022).

Results. We evaluated our method against several published baselines for both training from scratch and distillation. As shown in Table 8.1, when trained from scratch, our generator with 296M parameters outperforms both consistency training and its improved variant (Song and Dhariwal, 2024; Song et al., 2023b), with a much smaller training budget (200k iterations with batch size of 40 vs. 800k iterations with batch size of 512). Our model also competes favorably with iCT-deep, despite using a generator with half the number of parameters:

FID of 3.23 with 296M parameters (ours) vs. 3.25 with 592M parameters (iCT-deep). We observed stable training throughout, without requiring extensive hyperparameter tuning or special noise schedule adjustments as in consistency training, as visualized in Figure 8.3a. We also surpass the ECT model (Geng et al., 2024) of similar size and training budget that includes several modifications to induce stability in consistency training. Samples generated using our method can be found in Figure 8.4 and Appendix 8.D.

In the distillation setting, our model achieves a competitive FID of 1.48, outperforming several baselines. Notably, we outperform consistency distillation methods, such as multistep consistency distillation (Heek et al., 2024), despite using only a fraction of the model size (256M parameters against 1200M parameters). Our model also surpasses consistency trajectory models (CTM) (Kim et al., 2024), without the need for expensive simulation of the probability flow ODE. We also outperform reverse-KLD methods with similar compute or regularizers such as DMD (Yin et al., 2024b) and DMD2 with FIDs of 5.60 and 1.51 respectively. We note that on spending significant extra compute, DMD and DMD2 achieved improved results with expensive regularizers that require simulation of the pretrained model or lengthy finetuning stages of 400k steps. We did not resort to these techniques and sought to find an approach that worked best with a single execution of the training pipeline.

8.3.2 Unconditional CIFAR-10 Generation

Experimental Setup. We evaluated our method on the CIFAR-10 dataset (Krizhevsky et al., 2009) for unconditional one-step generative modeling, considering both training from scratch and distillation. In both cases, we employed a DDPM++ architecture (Song et al., 2020) with EDM preconditioning. The discriminator again followed the convolutional stack used in DMD2. For training from scratch, we modified the score model to incorporate the α-embedding (Section 8.1.4) while maintaining a similar network size. To mitigate overfitting due to the dataset's small size, we enabled dropout with $p = 0.13$, as in EDM. In the distillation setting, we initialized the generator with a pretrained unconditional diffusion model from (Karras et al., 2022), using the same UNet backbone and weights. Distillation performed well without dropout.

Results. The last three columns in Table 8.1 highlight the performance of our method on CIFAR-10 compared to various baselines. In our training from scratch setting, despite utilizing a lower training budget (150k steps with a batch size of 40) than many methods, our approach remains highly competitive. In terms of training budget, the most comparable baseline is ECT, which we are able to outperform without requiring excessive design considerations and hyperparameter tuning. Our distillation results are also competitive. In particular, we outperform DiffInstruct and DMD2, which are only based on minimizing the reverse KLD. This corroborates the benefit of our multi-divergence minimization approach. Image samples can be found in Appendix 8.D.

8.3.3 Ablation Studies

We use the CIFAR-10 dataset to study the effectiveness of the design choices that we have proposed; see Figure 8.3b.

Table 8.2.: Comparison of different generative modeling techniques capable of high-quality sample generation.

Generative models	Training idea	Generation	Training stability	Require pretrained model?
GAN	minimizing JSD, with discriminator	one-step	unstable	N
Diffusion models	training multi-noise-level denoisers via DSM	multi-step	stable	N
Diffusion distillation	(mostly) minimizing reverse KLD (in DMD)	{one,few}-step	stable	Y
Consistency distillation	simulating trajectories of probability flow ODE	{one,few}-step	stable	Y
Consistency training			unstable	N
Score-of-Mixture Training (ours)	minimizing $\{\alpha\text{-JSD}\}_{\alpha\in[0,1]}$ with multi-noise-level training, & scores of *mixtures* via DSM	one-step	stable	N
Score-of-Mixture Distillation (ours)				Y

Choice of Adaptive Gradient Weighting. Starting with our base objective without the GAN regularizer, we tested our (α, t)-adaptive weighting in Eq. (8.7). Figure 8.3b demonstrates the benefits of our weighting scheme, compared to the DMD weight function that only depends on t.

Learning with Single vs. Multiple α's. The α-JSD reduces to the reverse KLD of DMD and other distillation methods, when $\alpha = 1$. To test the efficacy with multi-α learning, we implemented an amortized variant, training the score model only with $\alpha \in \{0, 1\}$. Results show that conditioning on a range of α-values not only minimizes multiple divergences but also strengthens the α embedding as a conditioning signal thereby facilitating more accurate divergence minimization.

Accelerated Convergence with GAN Regularizer. We finally verify the benefits of our novel GAN-type regularizer for α-JSD minimization. As demonstrated by the second and fourth curves in Figure 8.3b, the GAN regularizer helps accelerate convergence especially in the beginning of training.

8.4 Summary

We show that high-quality one-step generative models can be trained from scratch and in a stable manner, without simulating the reverse diffusion process or probability flow ODE as in diffusion models and consistency models. The key distinctive idea in our framework is a new multi-divergence minimization paradigm implemented by estimating the score of mixture distributions. For stable training, we borrowed multi-level noise learning and denoising score matching techniques from the diffusion literature. Our empirical results show that accurate score estimation facilitates stable minimization of statistical divergences. We summarize our method alongside other generative models in Table 8.2.

8.5 Multi-Step Generative Modeling*

We can extend the SMT framework for multi-step generative modeling similar to consistency models. While the consistency model is a deterministic sampler, we will show that the resulting extension allows for stochastic sampling, thus leading to greater sample diversity in practice.

Suppose that we wish to train a multi-step generator of the form $\mathbf{g}_\theta(\mathbf{z}, \mathbf{x}_u, u)$, where $\mathbf{x}_u$ is a noisy version of an image $\mathbf{x}$ and u denotes the level of the noise. For a given u, let $q_\theta(\mathbf{x}|\mathbf{x}_u)$ denote the induced model (or fake) distribution.

Similar to the SMT framework, we consider minimizing

$$\min_\theta \mathbb{E}_{p(t)p(u)p(\alpha)p(\mathbf{x}_u|u)}\Big[\mathrm{D}^{(\alpha)}_{\mathrm{JSD}}\Big(q_\theta(\mathbf{x}_t|\mathbf{x}_u) \;\Big\|\; p(\mathbf{x}_t|\mathbf{x}_u)\Big)\Big]. \tag{8.15}$$

Here, we use t to denote the time step of the Gaussian corrupted version of $\mathbf{x}$ in the forward diffusion process which we leverage to apply the DSM trick, making a distinction from the time step u of $\mathbf{x}_u$, which is the noisy observation of $\mathbf{x}$, from which we wish to denoise.

In Eq. (8.15), the underlying probability model $p(\mathbf{x}, \mathbf{x}_u, \mathbf{x}_t) = p(\mathbf{x})p(\mathbf{x}_u|\mathbf{x})p(\mathbf{x}_t|\mathbf{x})$ satisfies the Markov chain

$$\mathbf{x}_u - \mathbf{x} - \mathbf{x}_t \quad \text{under } p(\mathbf{x}, \mathbf{x}_u, \mathbf{x}_t).$$

Similarly, our generative model assumes $q_\theta(\mathbf{x}, \mathbf{x}_u, \mathbf{x}_t) := p(\mathbf{x}_u)q_\theta(\mathbf{x}|\mathbf{x}_u)p(\mathbf{x}_t|\mathbf{x})$, which corresponds to the Markov chain

$$\mathbf{x}_u - \mathbf{x} - \mathbf{x}_t \quad \text{under } q_\theta(\mathbf{x}, \mathbf{x}_u, \mathbf{x}_t).$$

Here, $p(\mathbf{x}_u) := \mathbb{E}_{p(\mathbf{x})}[p(\mathbf{x}_u|\mathbf{x})]$ is the marginal noisy distribution of $p(\mathbf{x})$. Hence, the fake posterior distribution is written as

$$q_\theta(\mathbf{x}_t|\mathbf{x}_u) \triangleq \int p(\mathbf{x}_t|\mathbf{x})q_\theta(\mathbf{x}|\mathbf{x}_u)\,\mathrm{d}\mathbf{x} = \mathbb{E}_{q_\theta(\mathbf{x}|\mathbf{x}_u)}[p(\mathbf{x}_t|\mathbf{x})], \tag{8.16}$$

for any t and u.

Generator Update. We wish to train a multi-step generative model $(\mathbf{z}, \mathbf{x}_u) \mapsto \mathbf{g}_\theta(\mathbf{z}, \mathbf{x}_u, u)$. The gradient update for the generator is

$$\gamma_\psi(\theta; \alpha, t, \mathbf{x}_u, u) = \mathbb{E}_{q(\mathbf{z})}\left[\nabla_\theta \mathbf{g}_\theta(\mathbf{z}, \mathbf{x}_u, u)\frac{\mathbf{s}_\psi(\mathbf{x}_t; 0, t, \mathbf{x}_u, u) - \mathbf{s}_\psi(\mathbf{x}_t; \alpha, t, \mathbf{x}_u, u)}{\alpha}\bigg|_{\mathbf{x}=\mathbf{g}_\theta(\mathbf{z},\mathbf{x}_u,u)}\right],$$

where we now leverage an amortized score model $\mathbf{s}_\psi(\mathbf{x}_t; \alpha, t, \mathbf{x}_u, u)$ that is quadruply amortized over α, t, $\mathbf{x}_u$, and u.

Amortized Score Estimation. The amortized score model can be characterized by the optimal solution of a mixture of DSM losses

$$\mathcal{L}_{\text{score}}(\psi) := \mathbb{E}_{p(t)p(\alpha)}\Big[\mathcal{L}_{\text{score}}(\psi; \alpha, t)\Big],$$

where

$$\mathcal{L}_{\text{score}}(\psi;\alpha,t,u) := \mathbb{E}_{p(\mathbf{x}_u)}\Bigg[\alpha\ \mathbb{E}_{p(\mathbf{x}_t|\mathbf{x}_u)}\Big[\|\mathbf{s}_\psi(\mathbf{x}_t;\alpha,t,\mathbf{x}_u,u) - \nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x}_u)\|_2^2\Big] \\ + (1-\alpha)\,\mathbb{E}_{q_\theta(\mathbf{x}_t|\mathbf{x}_u)}\Big[\|\mathbf{s}_\psi(\mathbf{x}_t;\alpha,t,\mathbf{x}_u,u) - \nabla_{\mathbf{x}_t}\log q_\theta(\mathbf{x}_t|\mathbf{x}_u)\|_2^2\Big]\Bigg].$$

For the scores of posterior distributions $\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x}_u)$ and $\nabla_{\mathbf{x}_t}\log q_\theta(\mathbf{x}_t|\mathbf{x}_u)$, we apply Tweedie's formula and obtain

$$\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x}_u) = \mathbb{E}_{p(\mathbf{x}|\mathbf{x}_t,\mathbf{x}_u)}\big[\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x},\mathbf{x}_u)\big] = \mathbb{E}_{p(\mathbf{x}|\mathbf{x}_t,\mathbf{x}_u)}\big[\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x})\big].$$
$$\nabla_{\mathbf{x}_t}\log q_\theta(\mathbf{x}_t|\mathbf{x}_u) = \mathbb{E}_{q_\theta(\mathbf{x}|\mathbf{x}_t,\mathbf{x}_u)}\big[\nabla_{\mathbf{x}_t}\log q_\theta(\mathbf{x}_t|\mathbf{x},\mathbf{x}_u)\big] = \mathbb{E}_{q_\theta(\mathbf{x}|\mathbf{x}_t,\mathbf{x}_u)}\big[\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x})\big].$$

The last equation follows since $q_\theta(\mathbf{x}_t|\mathbf{x},\mathbf{x}_u) = p(\mathbf{x}_t|\mathbf{x})$. Hence, we can instead minimize

$$\mathcal{L}'_{\text{score}}(\psi;\alpha,t,u) := \mathbb{E}_{p(\mathbf{x}_u)}\Bigg[\alpha\ \mathbb{E}_{p(\mathbf{x}_t|\mathbf{x}_u)p(\mathbf{x}|\mathbf{x}_t,\mathbf{x}_u)}\Big[\|\mathbf{s}_\psi(\mathbf{x}_t;\alpha,t,\mathbf{x}_u,u) - \nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x})\|_2^2\Big] \\ + (1-\alpha)\,\mathbb{E}_{q_\theta(\mathbf{x}_t|\mathbf{x}_u)q_\theta(\mathbf{x}|\mathbf{x}_t,\mathbf{x}_u)}\Big[\|\mathbf{s}_\psi(\mathbf{x}_t;\alpha,t,\mathbf{x}_u,u) - \nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x})\|_2^2\Big]\Bigg]$$
$$\overset{(a)}{=} \alpha\ \mathbb{E}_{p(\mathbf{x})p(\mathbf{x}_u,\mathbf{x}_t|\mathbf{x})}\Big[\|\mathbf{s}_\psi(\mathbf{x}_t;\alpha,t,\mathbf{x}_u,u) - \nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x})\|_2^2\Big] \\ + (1-\alpha)\,\mathbb{E}_{p(\mathbf{x}_u)q_\theta(\mathbf{x}|\mathbf{x}_u)p(\mathbf{x}_t|\mathbf{x})}\Big[\|\mathbf{s}_\psi(\mathbf{x}_t;\alpha,t,\mathbf{x}_u,u) - \nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x})\|_2^2\Big].$$

Here, (a) follows from the definition of the probability models $p(\mathbf{x},\mathbf{x}_u,\mathbf{x}_t)$ and $q_\theta(\mathbf{x},\mathbf{x}_u,\mathbf{x}_t)$.

Appendix

8.A Deferred Proofs

8.A.1 Proof of Proposition 8.1

Proof of Proposition 8.1. We can simplify the gradient of each term separately as follows:

$$\begin{aligned}\nabla_\theta \mathrm{D}_{\mathrm{KL}}(q_\theta \| \alpha p + (1-\alpha) q_\theta) &= \mathbb{E}_{q_\theta(\mathbf{x})}\left[\nabla_\theta \log \frac{q_\theta(\mathbf{x})}{\alpha p(\mathbf{x}) + (1-\alpha) q_\theta(\mathbf{x})}\right] \\ &\quad + \mathbb{E}_{q(\mathbf{z})}\left[\nabla_\theta \mathbf{g}_\theta(\mathbf{z})(\mathbf{s}_{\theta;0}(\mathbf{x}) - \mathbf{s}_{\theta;\alpha}(\mathbf{x}))\Big|_{\mathbf{x}=\mathbf{g}_\theta(\mathbf{z})}\right], \\ \nabla_\theta \mathrm{D}_{\mathrm{KL}}(p \| \alpha p + (1-\alpha) q_\theta) &= -\mathbb{E}_{p(\mathbf{x})}\left[\nabla_\theta \log(\alpha p(\mathbf{x}) + (1-\alpha) q_\theta(\mathbf{x}))\right].\end{aligned}$$

Here, note that in the first expression, we invoke the chain rule: for some function $f_\theta \colon \mathcal{X} \to \mathbb{R}$, we have

$$\nabla_\theta f_\theta(\mathbf{g}_\theta(\mathbf{z})) = (\nabla_\theta f_\theta(\mathbf{x}))|_{\mathbf{x}=\mathbf{g}_\theta(\mathbf{z})} + \nabla_\theta \mathbf{g}_\theta(\mathbf{z})(\nabla_{\mathbf{x}} f_\theta(\mathbf{x}))|_{\mathbf{x}=\mathbf{g}_\theta(\mathbf{z})}.$$

Combining these two terms with the weights, we get the gradient of the α-skew JSD:

$$\begin{aligned}\nabla_\theta \mathrm{D}^{(\alpha)}_{\mathrm{JSD}}(q_\theta, p) &= \frac{1}{\alpha}\nabla_\theta \mathrm{D}_{\mathrm{KL}}(q_\theta \| \alpha p + (1-\alpha) q_\theta) + \frac{1}{1-\alpha}\nabla_\theta \mathrm{D}_{\mathrm{KL}}(p \| \alpha p + (1-\alpha) q_\theta) \\ &= \frac{1}{\alpha}\mathbb{E}_{q(\mathbf{z})}\left[\nabla_\theta \mathbf{g}_\theta(\mathbf{z})(\mathbf{s}_{\theta;0}(\mathbf{x}) - \mathbf{s}_{\theta;\alpha}(\mathbf{x}))\Big|_{\mathbf{x}=\mathbf{g}_\theta(\mathbf{z})}\right] \\ &\quad - \frac{1}{\alpha(1-\alpha)}\mathbb{E}_{\alpha p(\mathbf{x}) + (1-\alpha) q_\theta(\mathbf{x})}[\nabla_\theta \log(\alpha p(\mathbf{x}) + (1-\alpha) q_\theta(\mathbf{x}))] \\ &\quad + \frac{1}{\alpha}\mathbb{E}_{q_\theta(\mathbf{x})}[\nabla_\theta \log q_\theta(\mathbf{x})] \\ &= \frac{1}{\alpha}\mathbb{E}_{q(\mathbf{z})}\left[\nabla_\theta \mathbf{g}_\theta(\mathbf{z})(\mathbf{s}_{\theta;0}(\mathbf{x}) - \mathbf{s}_{\theta;\alpha}(\mathbf{x}))\Big|_{\mathbf{x}=\mathbf{g}_\theta(\mathbf{z})}\right].\end{aligned}$$

Here, we use the assumption that $\mathbb{E}_{q_\theta(\mathbf{x})}[\nabla_\theta \log q_\theta(\mathbf{x})] = 0$. □

8.A.2 Proof of Proposition 8.2

Proof of Proposition 8.2. We can write the objective $\mathcal{L}(\psi; \alpha)$ as

$$\begin{aligned}&\mathcal{L}(\psi; \alpha) \\ &= \int \left\{(\alpha p(\mathbf{x}) + (1-\alpha) q_\theta(\mathbf{x}))\|\mathbf{s}_\psi(\mathbf{x}; \alpha)\|^2 - 2(\alpha p(\mathbf{x}) s_p(\mathbf{x}) + (1-\alpha) q_\theta(\mathbf{x}))^\intercal \mathbf{s}_\psi(\mathbf{x}; \alpha)\right\} \mathrm{d}\mathbf{x} + C \\ &= \int (\alpha p(\mathbf{x}) + (1-\alpha) q_\theta(\mathbf{x}))\left\|\mathbf{s}_\psi(\mathbf{x}; \alpha) - \frac{\alpha p(\mathbf{x}) \mathbf{s}_p(\mathbf{x}) + (1-\alpha) q_\theta(\mathbf{x}) \mathbf{s}_{q_\theta}(\mathbf{x})}{\alpha p(\mathbf{x}) + (1-\alpha) q_\theta(\mathbf{x})}\right\|^2 \mathrm{d}\mathbf{x} + C'.\end{aligned}$$

Hence, it is clear that the global minimizer should be

$$\begin{aligned}\mathbf{s}_{\psi^*}(\mathbf{x};\alpha) &= \frac{\alpha p(\mathbf{x})\mathbf{s}_p(\mathbf{x}) + (1-\alpha)q_\theta(\mathbf{x})\mathbf{s}_{q_\theta}(\mathbf{x})}{\alpha p(\mathbf{x}) + (1-\alpha)q_\theta(\mathbf{x})} \\ &= \frac{\alpha\nabla_{\mathbf{x}} p(\mathbf{x}) + (1-\alpha)\nabla_{\mathbf{x}} q_\theta(\mathbf{x})}{\alpha p(\mathbf{x}) + (1-\alpha)q_\theta(\mathbf{x})} \\ &= \frac{\nabla_{\mathbf{x}}(\alpha p(\mathbf{x}) + (1-\alpha)q_\theta(\mathbf{x}))}{\alpha p(\mathbf{x}) + (1-\alpha)q_\theta(\mathbf{x})} \\ &= \nabla_{\mathbf{x}} \log(\alpha p(\mathbf{x}) + (1-\alpha)q_\theta(\mathbf{x})).\end{aligned}$$

□

8.A.3 Proof of Proposition 8.2

Proof. We can write the objective $\mathcal{L}(\psi;\alpha,t)$ as

$$\mathcal{L}_{\text{score}}(\psi;\alpha,t) = \iint (\alpha p(\mathbf{x}_t) + (1-\alpha)q_\theta(\mathbf{x}_t))\left\|\mathbf{s}_\psi(\mathbf{x}_t;\alpha,t) + \frac{\boldsymbol{\epsilon}}{\sigma_t}\right\|^2 \mathrm{d}\mathbf{x}\,\mathrm{d}\boldsymbol{\epsilon}.$$

This is a standard minimum mean square estimation (MMSE) problem for which the global minimizer is the conditional mean,

$$\begin{aligned}\mathbf{s}_{\psi^*}(\mathbf{x}_t;\alpha,t) &= -\frac{1}{\sigma_t}\mathbb{E}_{\alpha p_t + (1-\alpha)q_{\theta,t}}[\boldsymbol{\epsilon}|\mathbf{x}_t] \\ &= -\frac{1}{\sigma_t^2}\mathbb{E}_{\alpha p_t + (1-\alpha)q_{\theta,t}}[\mathbf{x}_t - \mathbf{x}|\mathbf{x}_t] \\ &= -\frac{1}{\sigma_t^2}\mathbf{x}_t + \frac{1}{\sigma_t^2}\mathbb{E}_{\alpha p_t + (1-\alpha)q_{\theta,t}}[\mathbf{x}|\mathbf{x}_t] \\ &= \nabla_{\mathbf{x}_t} \log(\alpha p(\mathbf{x}_t) + (1-\alpha)q_\theta(\mathbf{x}_t)).\end{aligned}$$

Here we use that $\mathbf{x}_t = \mathbf{x} + \sigma_t\boldsymbol{\epsilon}$ and make the connection to the marginal score in the last line using Tweedie's formula using Eq. (2.8). □

8.A.4 Proof of Proposition 8.4

Proof. The amortized score can be expressed as

$$\begin{aligned}&\nabla_{\mathbf{x}} \log(\alpha p(\mathbf{x}) + (1-\alpha)q_\theta(\mathbf{x})) \\ &\qquad= \frac{\nabla_{\mathbf{x}}(\alpha p(\mathbf{x}) + (1-\alpha)q_\theta(\mathbf{x}))}{\alpha p(\mathbf{x}) + (1-\alpha)q_\theta(\mathbf{x})} \\ &\qquad= \frac{\alpha p(\mathbf{x})}{\alpha p(\mathbf{x}) + (1-\alpha)q_\theta(\mathbf{x})}\nabla_{\mathbf{x}} \log p(\mathbf{x}) + \frac{(1-\alpha)q_\theta(\mathbf{x})}{\alpha p(\mathbf{x}) + (1-\alpha)q_\theta(\mathbf{x})}\nabla_{\mathbf{x}} \log q_\theta(\mathbf{x}) \\ &\qquad= D(\mathbf{x};\alpha)\nabla_{\mathbf{x}} \log p(\mathbf{x}) + (1 - D(\mathbf{x};\alpha))\nabla_{\mathbf{x}} \log q_\theta(\mathbf{x}).\end{aligned}$$

We can now simplify the scaling factor as

$$D(\mathbf{x};\alpha) = \frac{\alpha p(\mathbf{x})}{\alpha p(\mathbf{x}) + (1-\alpha)q_\theta(\mathbf{x})} = \sigma\left(\log\frac{p(\mathbf{x})}{q_\theta(\mathbf{x})} + \log\frac{\alpha}{1-\alpha}\right).$$

□

8.B Discriminator Training via Score-of-Mixture-Distillation

In Section 8.2, we plugged in the explicit parameterization

$$\mathbf{s}_\psi^{\mathsf{exp}}(\mathbf{x};\alpha) := D_\psi(\mathbf{x};\alpha)\mathbf{s}_p(\mathbf{x}) + (1 - D_\psi(\mathbf{x};\alpha))\mathbf{s}_\psi^{\mathsf{fake}}(\mathbf{x}),$$

to the mixture regression loss in Eq. (8.4), to train the fake score and the discriminator simultaneously. If we consider an ideal scenario where we have the perfect score models for both p and q, then all we need to train is the discriminator andthe mixture regression objective can be interpreted as a discriminator objective. Here we reveal its connection to an instance of f-GAN discriminator objective.

Let $\mathbf{s}_p(\mathbf{x})$ and $\mathbf{s}_q(\mathbf{x})$ be the underlying score functions for p and q, respectively. Then, the explicit parameterization becomes

$$\mathbf{s}_\psi^{\mathsf{exp}}(\mathbf{x};\alpha) = D_\psi(\mathbf{x};\alpha)\mathbf{s}_p(\mathbf{x}) + (1 - D_\psi(\mathbf{x};\alpha))\mathbf{s}_q(\mathbf{x}),$$

and the mixture regression objective becomes only a function of the discriminator, i.e.,

$$\begin{aligned}
\mathcal{L}(\psi;\alpha) &= \alpha\,\mathbb{E}_{p(\mathbf{x})}[\|\mathbf{s}_\psi^{\mathsf{exp}}(\mathbf{x};\alpha) - \mathbf{s}_p(\mathbf{x})\|^2] + (1-\alpha)\,\mathbb{E}_{q_\theta(\mathbf{x})}[\|\mathbf{s}_\psi^{\mathsf{exp}}(\mathbf{x};\alpha) - \mathbf{s}_q(\mathbf{x})\|^2] \\
&= \alpha\,\mathbb{E}_{p(\mathbf{x})}[(1 - D_\psi(\mathbf{x};\alpha))^2\|\mathbf{s}_p(\mathbf{x}) - \mathbf{s}_q(\mathbf{x})\|^2] + (1-\alpha)\,\mathbb{E}_{q(\mathbf{x})}[D_\psi(\mathbf{x};\alpha)^2\|\mathbf{s}_p(\mathbf{x}) - \mathbf{s}_q(\mathbf{x})\|^2] \\
&= \int \Big\{\alpha p(\mathbf{x})(1 - D_\psi(\mathbf{x};\alpha))^2 + (1-\alpha)q(\mathbf{x})D_\psi(\mathbf{x};\alpha)^2\Big\}\|\mathbf{s}_p(\mathbf{x}) - \mathbf{s}_q(\mathbf{x})\|^2\,\mathrm{d}\mathbf{x}.
\end{aligned}$$

Here, we note that the term $\|\mathbf{s}_p(\mathbf{x}) - \mathbf{s}_q(\mathbf{x})\|^2$ is common in both expectation, and can be safely dropped to train the discriminator, which leads to a simplified objective

$$\begin{aligned}
\mathcal{L}'(\psi;\alpha) &= \alpha\,\mathbb{E}_{p(\mathbf{x})}[(1 - D_\psi(\mathbf{x};\alpha))^2] + (1-\alpha)\,\mathbb{E}_{q(\mathbf{x})}[D_\psi(\mathbf{x};\alpha)^2] \\
&= \int \Big\{\alpha p(\mathbf{x})(1 - D_\psi(\mathbf{x};\alpha))^2 + (1-\alpha)q(\mathbf{x})D_\psi(\mathbf{x};\alpha)^2\Big\}\,\mathrm{d}\mathbf{x}.
\end{aligned}$$

We note that this is equivalent to the discriminator objective induced by the following f-divergence

$$D_{f_\alpha}(p \parallel q) := 1 - \int \frac{p(\mathbf{x})q(\mathbf{x})}{\alpha p(\mathbf{x}) + (1-\alpha)q(\mathbf{x})}\,\mathrm{d}\mathbf{x} := D_{\alpha\text{-}\mathsf{LC}}(p \parallel q),$$

where $f_\alpha(r) := \frac{(1-\alpha)(1-r)}{\alpha r + (1-\alpha)}$ is a convex function over $[0,\infty)$ for $\alpha \in (0,1)$. For $\alpha = \frac{1}{2}$, this divergence becomes symmetric in p and q and is known as the Le Cam distance (Le Cam, 2012, p. 47) in the literature (Polyanskiy and Wu, 2019). We thus call the general divergence for $\alpha \in (0,1)$ the α-Le Cam distance. In the GAN literature, this is known as the LSGAN objective (Mao et al., 2017).

As revealed, our discriminator training in distillation can also be done separately using the α-Le Cam-distance-based objective. However, we conjecture that our score-regression-based end-to-end objective may have benefit, as our primary goal of discriminator training is to use it in the generator update in the form of an approximate score of mixture.

8.C Algorithm Blocks

Algorithm 8.1 Score-of-Mixture Training

Inputs: Randomly initialized generator $\mathbf{g}_\theta$, amortized score model $\mathbf{s}_\psi$, discriminator ℓ_ψ, real dataset $\mathcal{D}$, score training sub-iterations $= 5$, learning rates $(\eta_{\text{gen}}, \eta_{\text{score}})$, GAN regularizer weights (score $= \mu$, gen $= \lambda$)

Pretraining: Train $\mathbf{g}_\theta$ with DSM using $\mathcal{D}$

for each pretraining iteration **do**

  Sample mini-batch $\mathbf{x} \sim \mathcal{D}$ and add noise $\mathbf{x}_t = \mathbf{x} + \sigma_t \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$

  Compute DSM loss $\mathcal{L}_{\text{DSM}}(\theta)$ (see Section 2.2.1.3)

  Update parameters: $\theta \leftarrow \theta - \eta_{\text{DSM}} \nabla_\theta \mathcal{L}_{\text{DSM}}(\theta)$

Training: Alternating updates of $\mathbf{g}_\theta$ and $\mathbf{s}_\psi$

for each training iteration **do**

  Generator Training: Freeze $\mathbf{s}_\psi$

  Sample mini-batch of fake samples $\mathbf{x}^{\text{fake}} = \mathbf{g}_\theta(\mathbf{z}), \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$

  Sample $t \sim p(t)$ and α as described in Section 8.1.5

  Compute weighted generator gradient $\gamma_\psi^w(\theta; \alpha, t)$ from Eq. (8.6)

  Compute GAN regularizer loss $\mathcal{L}_{\text{GAN}}^{(\alpha,t)}$ from Eq. (8.8)

  Update parameters:

$$\theta \leftarrow \theta - \eta_{\text{gen}} \mathbb{E}_{p(\alpha)p(t)}[w(\mathbf{x}_t^{\text{fake}}, \mathbf{x}^{\text{fake}}, \alpha, t)\gamma_\psi^w(\theta; \alpha, t) + \lambda \nabla_\theta \mathcal{L}_{\text{GAN}}^{(\alpha,t)}(\theta)]$$

  Amortized Score Training: Freeze $\mathbf{g}_\theta$

  for each sub-iteration **do**

    Sample mini-batch of real samples $\mathbf{x}^{\text{real}} \sim \mathcal{D}$

    Sample $t \sim p(t)$ and α

    Compute score matching loss $\mathcal{L}_{\text{score}}(\psi; \alpha, t)$ from Eq. (8.5)

    Compute non-saturated discriminator loss $\mathcal{L}_{\text{disc}}(\psi)$ (see Section 7.1)

    Update parameters:

$$\psi \leftarrow \psi - \eta_{\text{score}} \nabla_\psi (\mathbb{E}_{p(\alpha)p(t)}[\mathcal{L}_{\text{score}}(\psi; \alpha, t)] + \mu \mathcal{L}_{\text{disc}}(\psi))$$

Return: Trained model parameters θ, ψ

Algorithm 8.2 Score-of-Mixture Distillation

Inputs: Randomly initialized generator $\mathbf{g}_\theta$, fake score model $\mathbf{s}_\psi^{\mathsf{fake}}$, discriminator ℓ_ψ, pretrained score model $\mathbf{s}_p$, real dataset $\mathcal{D}$, score training sub-iterations $= 5$, learning rates $(\eta_{\text{gen}}, \eta_{\text{score}})$, GAN generator regularizer weight λ

Initialization Initialize $\mathbf{s}_\psi^{\mathsf{fake}}$ and $\mathbf{g}_\theta$ with weights from $\mathbf{s}_p$

Training: Alternating updates of $\mathbf{g}_\theta$ and $\mathbf{s}_\psi$

for each training iteration **do**

  Generator Training: Freeze $\mathbf{s}_\psi^{\mathsf{fake}}$ and ℓ_ψ

  Sample mini-batch of fake samples $\mathbf{x}^{\mathsf{fake}} = \mathbf{g}_\theta(\mathbf{z}), \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$

  Sample $t \sim p(t)$ and α as described in Section 8.1.5

  Compute generator gradient $\gamma_\psi^{\mathsf{ext}}(\theta; \alpha, t)$ from Eq. (8.14)

  Compute GAN regularizer loss $\mathcal{L}_{\text{GAN}}^{(\alpha,t)}$ from Eq. (8.8)

  Update parameters:

$$\theta \leftarrow \theta - \eta_{\text{gen}} \mathbb{E}_{p(\alpha)p(t)}[\gamma_\psi^{\mathsf{ext}}(\theta; \alpha, t) + \lambda \nabla_\theta \mathcal{L}_{\text{GAN}}^{(\alpha,t)}(\theta)]$$

  Amortized Score Training: Freeze $\mathbf{g}_\theta$

  for each sub-iteration **do**

    Sample mini-batch of real samples $\mathbf{x}^{\mathsf{real}} \sim \mathcal{D}$

    Sample $t \sim p(t)$ and α

    Compute score matching loss using explicit parametrization $\mathcal{L}_{\text{score}}^{\mathsf{ext}}(\psi; \alpha, t)$ from Eq. (8.13)

    Update parameters:

$$\psi \leftarrow \psi - \eta_{\text{score}} \nabla_\psi \mathbb{E}_{p(\alpha)p(t)}[\mathcal{L}_{\text{score}}^{\mathsf{ext}}(\psi; \alpha, t)]$$

Return: Trained model parameters θ, ψ

8.D Additional Experimenal Details and Results

We present some additional experiments and results in this section. We first provide a more detailed training configuration for our experiments in Section 8.3 and then evaluate our proposed method on a synthetic swiss-roll dataset in Appendix 8.D.2. Finally, we present some samples generated from Score-of-Mixture Training and Score-of-Mixture Distillation in Figures 8.6-8.9.

8.D.1 Training Configuration

We summarize the detailed training configuration in Table 8.3.

Table 8.3.: Hyperparameters used for training one-step generators with Score-of-Mixture Training and Distillation.

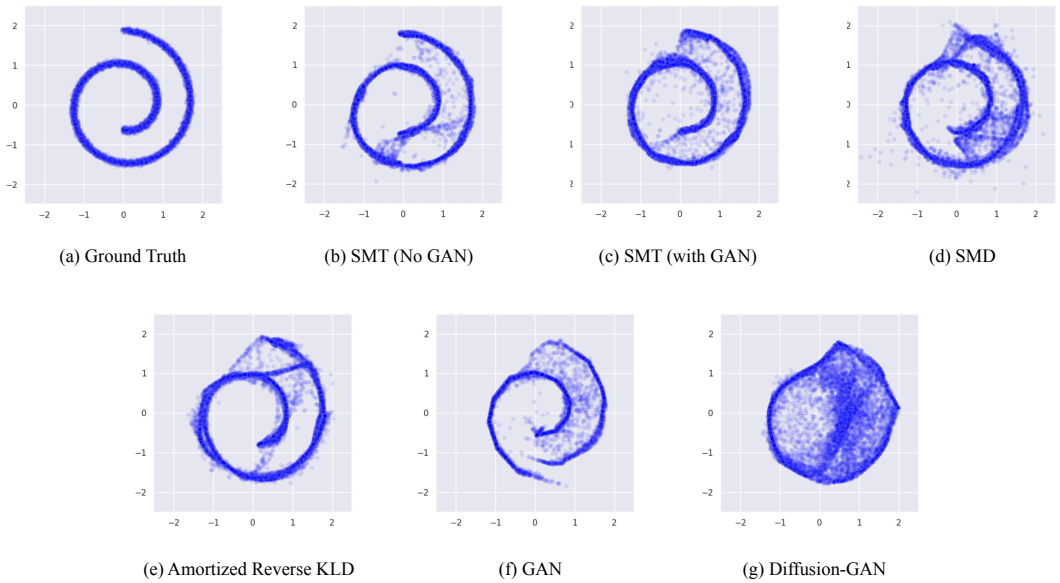
Hyperparameter	CIFAR-10		ImageNet 64 × 64	
	Scratch	Distillation	Scratch	Distillation
Generator learning rate	1e-4	5e-5	5e-6	2e-6
Score learning rate	5e-4	5e-5	5e-5	2e-6
Score learning rate decay	cosine	None	cosine	None
Batch size	280	280	280	280
Diffusion pretraining steps	15k	N/A	40k	N/A
Training iterations	150k	150k	200k	200k
Score dropout probability	0.13	0.00	0.00	0.00
Number of GPUs	2 × A100	4× A100	7× A100	7× A100

8.D.2 Toy Swiss Roll

We tested our proposed framework and ablated various design choices on a synthetic swiss roll dataset. We followed the dataset setup by Che et al. (2020). We trained models with SMT and SMD and compared this against an amortized version of reverse KL minimization with DMD weighting ($\alpha \in \{0, 1\}$) similar to the ablations in Section 8.3.3. Additionally we compared against non-score-based baselines including the vanilla GAN and Diffusion-GAN (Wang et al., 2023).

Across all experiments, we use the same generator architecture — a two-layer MLP with a hidden dimension of 128 and leaky ReLU nonlinearity. We train all models for 200k steps on a single NVIDIA 3090 GPU with a batch size of 256. All score-based methods leverage a learning rate of 1e-5 for the generator and 1e-4 for the amortized score (and discriminator when applicable) whereas the GAN-based methods use a learning rate of 1e-4 for both generator and discriminator. We use the AdamW optimizer without any learning rate schedulers.

The samples produced are shown in Figure 8.5. Notice how the GAN is unable to perfectly cover the entire continuous mode of the swiss roll. The multi-noise-level extension of GAN-based on Diffusion-GAN covers the mode but also samples from areas of low density. We



(a) Ground Truth (b) SMT (No GAN) (c) SMT (with GAN) (d) SMD

(e) Amortized Reverse KLD (f) GAN (g) Diffusion-GAN

Figure 8.5.: Samples produced by generators trained using different methods. All figures are created using 10,000 samples from the respective generator.

found the latter to be sensitive to the chosen noise levels in comparison to the methods based on updating the generator using the score.

Our results for training from scratch and distillation are presented in Figure 8.5(b)-(d). All three methods successfully capture the modes of the underlying distribution. While the impact of the GAN regularizer is less pronounced than in our high-dimensional experiments, we observe that enabling it reduces the number of samples in low-density regions. The distillation results appear slightly noisy, likely due to the quality of the pre-trained score model. This highlights the advantage of training from scratch, as it avoids amplifying existing estimation errors in the pre-trained model.

Figure 8.5(e) presents the results of ablating the α-sampler. Unlike in the high-dimensional setting, the choice of the number of α values is less critical in this case. However, our method in (b) still produces fewer noisy samples, suggesting that training the amortized score model with multiple α values may be beneficial.

8.D.3 Samples

We present some image samples generated by SMT and SMD in Figures 8.6-8.9.

Figure 8.6.: One-step generated samples from SMT on CIFAR-10 (unconditional).

Figure 8.7.: One-step generated samples from SMD on CIFAR-10 (unconditional).

Figure 8.8.: One-step generated samples from SMT on ImageNet 64×64 (conditional).

Figure 8.9.: One-step generated samples from SMD on ImageNet 64×64 (conditional).

9 Concluding Remarks and Future Directions

In this final chapter we review the contributions of this thesis and summarize potential directions for future work.

9.1 Summary of Contributions

In this thesis, we provided an overview of the relationship between score estimation and generative modeling through principled theoretical development and applications in practical engineering contexts. We first tackled the deficiencies of existing score estimation frameworks and introduced new estimation objectives that lead to tangible improvements in generative modeling as measured by sample quality. Beyond their traditional use in diffusion-based sampling, we highlighted the broader potential of score functions by developing new score-driven algorithms for solving inverse problems. These methods are thoroughly examined in the context of interference mitigation for digital communication signals. We then revisited the foundational challenge of efficient neural synthesis and propose a new score estimation framework that supports a one-step generative modeling approach via multi-divergence minimization. A detailed summary of these contributions along with future directions and preliminary analysis is provided below.

9.1.1 From Nonparametric to Parametric Score Estimation

The first part of this thesis was centered on score estimation, with a particular focus on how parametric techniques can enhance the accuracy of existing methods. To showcase the power of parametric approaches, we revisited spectral-based nonparametric score estimation methods, which typically expand the score function using test functions. Existing techniques often rely on a fixed, and potentially mismatched, set of test functions derived from the eigenbasis of an arbitrary kernel. In contrast, we proposed a parametric method that learns the optimal eigenbasis that minimizes the estimation error—a technique we refer to as principal direction score estimation.

Furthermore, taking inspiration from the nonparametric score estimation method, we proposed a new optimization framework for learning the score by lifting it into the space of its outer product with itself. This can again be interpreted as solving for the optimal eigenbasis of a matrix-valued kernel. Furthermore, by augmenting this method with iterative residual

learning to compensate for empirical estimation errors, the resulting framework called lifted residual score estimation was able to outperform several baselines for training both implicit and score-based generative models.

An interesting insight from the latter framework is its natural extension to higher-order lifted spaces, using tensor products of the score of order $m \in \mathbb{Z}_+$. This generalization yields loss functions with increasingly sharp curvature near the global optimum, which may facilitate faster convergence under appropriate hyperparameter settings. Extending this lifting principle to other regression-based objectives and understanding the theoretical implications of such transformations—especially in combination with residual estimation—opens up exciting directions for future research.

9.1.2 Score-based Statistical Signal Priors

In the second part of this thesis, we shifted our focus from learning score functions to leveraging parametric score models for solving inverse problems. In many engineering domains, signal recovery tasks are traditionally guided by hand-crafted priors designed to encompass the structure of the unknown signal. However, these manually designed models are often overly simplistic or mismatched, leading to suboptimal performance and requiring significant domain expertise—making them difficult to scale or generalize across applications.

To address these limitations, we proposed a novel Bayesian algorithm for single-channel source separation, inspired by MAP estimation principles. Unlike conventional methods, our approach does not rely on prior knowledge of the mixture model or joint signal statistics. Instead, it uses independent signal priors, expressed as score functions of the individual components, and introduces randomness across noise levels to perform iterative signal recovery. We demonstrate significant improvements in signal recovery performance when applied to mixtures of digital communication signals. Additionally, the proposed algorithm effectively handles the underlying discreteness of such signals, recovering both their continuous and discrete structures.

Although this thesis emphasized the source separation problem, the proposed algorithm is general and can be applied to a wide class of inverse problems. In contrast to existing diffusion-based methods that use pre-trained score models within a posterior sampling framework—often relying on Langevin dynamics—our method formulates an optimization problem to recover the MAP estimate. This avoids the need for artificial consistency constraints often used to enforce uniqueness in probabilistic reconstructions. The generality of our framework opens the door to a number of future extensions, including multi-source separation, explicit background noise modeling, and real-time deployment in communication systems. Future work could explore these directions and investigate how this optimization-based framework can be adapted to broader domains such as image, audio, and biomedical signal processing.

9.1.3 Score-of-Mixture Training for One-Step Generative Modeling

In recent years, data-driven techniques have become increasingly sophisticated and are now widely integrated into engineering systems to enhance both efficiency and accuracy in solving complex downstream tasks. The driving force behind such progress is access to large, diverse datasets. However, in many practical scenarios—such as proprietary domains or when only

partial datasets are open-sourced—data may be scarce or difficult to obtain. In such cases, the ability to simulate high-quality synthetic data becomes an invaluable asset.

Diffusion models currently represent the state of the art in generative modeling, producing high-fidelity samples across various data modalities. Despite their impressive performance, these models are computationally expensive to sample from, as they typically require many evaluations of the underlying score network. Recently, methods that distill pre-trained diffusion models into few-step generators have gained popularity, offering a more efficient sampling process while maintaining quality. However, these approaches still rely on an expensive pre-training phase. In contrast, one-step generative models—such as GANs or consistency models—can be trained from scratch but often suffer from training instability and require careful tuning.

To overcome these limitations, we introduced a new framework for training one-step generative models from scratch by directly minimizing a class of statistical divergences. The core idea is to learn the score of mixture distributions across multiple noise levels, inspired by diffusion models but without the need for costly trajectory simulation or adversarial objectives. This approach is not only simple to implement but also flexible: it can seamlessly incorporate a pre-trained model if available, allowing for efficient distillation. Our experiments showed that this divergence-based framework achieves state-of-the-art performance in one-step image generation, highlighting its potential as a scalable and practical alternative for high-quality data synthesis.

Although this thesis primarily focused on one-step generative modeling, the proposed framework can be naturally extended to train multi-step neural samplers, similar to consistency models. While consistency models train a deterministic sampler, it can be shown that the multi-step neural sampler learned through our framework is stochastic. This stochasticity is particularly useful for estimating uncertainty and providing greater control over the generation process. A promising direction for future research is to extend this multi-step generator for posterior sampling. The ability to generate samples from the posterior distribution is essential, particularly for solving inverse problems, and remains a key challenge for popular inverse problem solvers based on posterior sampling, such as SBI (see Section 1.1.2) and DPS (see Section 5.7.1). We will comment on this some more later in this chapter.

9.2 Preliminary Exploration Toward Future Directions

Throughout this thesis, we have offered guidance on possible extensions to the methods presented. Nevertheless, the ideas developed here open the door to many more directions worth exploring. In this final section, we amalgamate several of the proposed techniques and extensions to provide readers with a clearer, more actionable path highlighting how the tools introduced in this work can be leveraged to tackle a broad range of challenging problems.

9.2.1 MAP Estimation with Latent Diffusion Models

In Section 5.7, we introduced an extension of α-RGS for tackling general inverse problems, formulated as an optimization problem in the ambient signal space. While the use of randomized Gaussian smoothing helps to stabilize optimization, the high dimensionality

of the signal space can still lead to rugged loss landscapes, increasing the likelihood of convergence to suboptimal local minima. This issue becomes especially pronounced in high-resolution settings, such as inverse problems defined over pixel spaces.

Meanwhile, recent advances in generative modeling have turned to latent space diffusion models. These approaches begin by training an autoencoder—comprised of an encoder $\mathbf{f}_\phi$ and decoder $\mathbf{g}_\psi$—to map high-dimensional data into a compact latent space. This latent representation is often regularized to follow a smooth, typically Gaussian, prior distribution. A diffusion model is then trained directly in this latent space, where the reduced dimensionality simplifies both sampling and optimization. Once a sample is generated, it can be mapped back to the original data space through the decoder, enabling generation of high-fidelity outputs.

Given an input image $\mathbf{x}$, assume that it was generated from some latent embedding $\mathbf{z}$, i.e., $\mathbf{x} := \mathbf{g}_\psi(\mathbf{z})$. Given a measurement $\mathbf{y} = \mathsf{A}\mathbf{x} + \mathbf{n}$, $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_D)$, we can solve the MAP estimation problem in the latent space,

$$\arg\max_{\boldsymbol{\theta}} p_{\mathbf{z}|\mathbf{y}}(\boldsymbol{\theta}|\mathbf{y}).$$

Developing the α-RGS framework around this optimization problem we can derive a new objective whose gradient is,

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{latent-inv}}(\boldsymbol{\theta}) \approx -\mathbb{E}_{p(t)q(\boldsymbol{\epsilon}_z)}[\mathbf{s}_{\mathbf{z}_t}(\mathbf{z}_t(\boldsymbol{\theta}))] + \alpha \mathbb{E}_{p(u)q(\boldsymbol{\epsilon}_z)}[\|\mathbf{y} - \mathsf{A}\mathbf{g}_\psi(\mathbb{E}[\mathbf{z}|\mathbf{z}_u(\boldsymbol{\theta})])\|_2^2],$$

where $\mathbf{s}_{\mathbf{z}_t}(\mathbf{z}_t(\boldsymbol{\theta}))$ is the score obtained from a latent diffusion model. The second term is again simplified by using the DPS approximation in Eq. (5.22).

We conducted preliminary experiments to evaluate the effectiveness of the proposed method on the problem of recovering signals from motion blur. As baselines, we compared against two posterior sampling approaches: DPS and PSLD (Rout et al., 2023), a latent-space variant of DPS. For training our model, we adopted a cosine annealing learning rate schedule that decays from 1e-1 to 1e-6 and we gradually increase the value of α over time. This schedule encourages broad exploration during the early stages of training while progressively enforcing the measurement constraints more strictly as training progresses. In all experiments we used an open-source Stable Diffusion v1.5 pre-trained diffusion model[1] that was trained for several weeks on large scale image datasets at a resolution of 512×512.

Figure 9.1 presents the results of this experiment. While DPS, which operates directly in pixel space, struggles to preserve fine structures—evident in the distortion of the human figure in the second row—both PSLD and α-RGS produce more visually coherent reconstructions. Overall, α-RGS demonstrates promising performance, though the outputs tend to be slightly oversmoothed. This limitation could potentially be mitigated by applying a few refinement steps using a diffusion sampler. Exploring strategies to enhance the perceptual sharpness and sample quality of α-RGS outputs remains a compelling direction for future research.

[1] https://github.com/faraday/runway-stable-diffusion-inpainting

(a) Measurement (b) Ground Truth (c) DPS (d) PSLD (e) α-RGS

Figure 9.1.: Recovered images using different methods. All images were recovered from the same motion-blurred measurement.

9.2.2 Exact Diffusion Posterior Sampling

Recall that the DPS framework approximates the gradient of the log-likelihood conditioned on a noisy signal of interest using the expression:

$$\nabla_{\mathbf{x}_t^{(i)}} \log p_{\mathsf{y}|\mathsf{x}_t}(\mathbf{y}|\mathbf{x}_t^{(i)}) \approx \nabla_{\mathbf{x}_t^{(i)}} \log p_{\mathsf{y}|\mathsf{x}}(\mathbf{y}|\mathbb{E}[\mathbf{x}|\mathbf{x}_t^{(i)}]).$$

Notably, we can also express the likelihood conditioned on the noisy input more generally as,

$$p_{\mathsf{y}|\mathsf{x}_t}(\mathbf{y}|\mathbf{x}_t) = \mathbb{E}_{p_{\mathsf{x}|\mathsf{x}_t}(\mathbf{x}|\mathbf{x}_t)}[p_{\mathsf{y}|\mathsf{x}}(\mathbf{y}|\mathbf{x})], \tag{9.1}$$

where we often assume that we have access to the likelihood model $p_{\mathsf{y}|\mathsf{x}}(\mathbf{y}|\mathbf{x})$.

Now, suppose we have access to an implicit neural sampler capable of drawing samples from the posterior $p_{\mathsf{x}|\mathsf{x}_t}$. Then the expectation in Eq. (9.1) can be rewritten as,

$$p_{\mathsf{y}|\mathsf{x}_t}(\mathbf{y}|\mathbf{x}_t) = \mathbb{E}_{q(\mathbf{z})p(\mathbf{x}_t)}[p_{\mathsf{y}|\mathsf{x}}(\mathbf{y}|\mathbf{x})|_{\mathbf{x}=\mathbf{g}_\theta(\mathbf{z},\mathbf{x}_t)}],$$

where $\mathbf{g}_\theta$ denotes the generator conditioned on the noisy signal of interest $\mathbf{x}_t$ and $\mathbf{z} \sim q(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I}_D)$. This naturally leads to a more accurate way of computing the gradient of the conditional likelihood:

$$\nabla_{\mathbf{x}_t^{(i)}} \log p_{\mathsf{y}|\mathsf{x}_t}(\mathbf{y}|\mathbf{x}_t^{(i)}) = \mathbb{E}_{q(\mathbf{z})}[\nabla_{\mathbf{x}_t^{(i)}} p_{\mathsf{y}|\mathsf{x}}(\mathbf{y}|\mathbf{x})|_{\mathbf{x}=\mathbf{g}_\theta(\mathbf{z},\mathbf{x}_t^{(i)})}].$$

This expectation can be efficiently approximated in practice via Monte Carlo sampling.

But how do we obtain such a neural sampler? This is precisely the outcome of the multi-step extension of the SMT framework introduced in Section 8.5. When trained properly, this multi-step generator not only enables high-quality sample generation but also serves as a flexible inference tool for solving inverse problems more accurately within the DPS framework. This connection opens up promising avenues for integrating learned generative samplers into principled Bayesian inference pipelines.

9.2.3 Fast and Efficient Posterior Sampling

Let's extend the proposed ideas one-step further and assume that we have access to paired samples of $(\mathbf{x}, \mathbf{y})$. This could be given to us ahead of time or could be simulated with a black box simulator. In this scenario, we can extend the SMT framework to train a one-step generator capable of sampling from the posterior distribution $p(\mathbf{x}|\mathbf{y})$, when a likelihood model $p(\mathbf{y}|\mathbf{x})$ is given.

In this case, the underlying probabilistic models are

$$\begin{aligned} p(\mathbf{x}, \mathbf{y}, \mathbf{x}_t) &:= p(\mathbf{x})p(\mathbf{y}|\mathbf{x})p(\mathbf{x}_t|\mathbf{x}), \\ q_\theta(\mathbf{x}, \mathbf{y}, \mathbf{x}_t) &:= p(\mathbf{y})q_\theta(\mathbf{x}|\mathbf{y})p(\mathbf{x}_t|\mathbf{x}), \end{aligned} \tag{9.2}$$

where $p(\mathbf{x}_t|\mathbf{x})$ is the forward variance exploding Gaussian diffusion kernel and $p(\mathbf{y}) := \mathbb{E}_{p(\mathbf{x})}[p(\mathbf{y}|\mathbf{x})]$ denotes the marginal distribution over the noisy observation $\mathbf{y}$.

To train a one-step generator $\mathbf{g}_\theta$ to sample $\mathbf{x}$ from a given $\mathbf{y}$ we can minimize the skew Jensen–Shannon divergence between the induced posterior distribution and the true posterior distribution,

$$\min_\theta \mathrm{D}^{(\alpha)}_{\mathrm{JSD}}(q_\theta(\mathbf{x}|\mathbf{y}) \parallel p(\mathbf{x}|\mathbf{y})).$$

As in the SMT framework, we can again extend this to multiple noise levels and multiple α's. Additionally we can amortize different observations,

$$\min_\theta \mathbb{E}_{p(t)p(\alpha)p(\mathbf{y})}[\mathrm{D}^{(\alpha)}_{\mathrm{JSD}}(q_\theta(\mathbf{x}_t|\mathbf{y}) \parallel p(\mathbf{x}_t|\mathbf{y}))].$$

Here, the noisy posterior distribution is defined as

$$p(\mathbf{x}_t|\mathbf{y}) \triangleq \int p(\mathbf{x}_t|\mathbf{x})p(\mathbf{x}|\mathbf{y})\,\mathrm{d}\mathbf{x} = \mathbb{E}_{p(\mathbf{x}|\mathbf{y})}[p(\mathbf{x}_t|\mathbf{x})] \tag{9.3}$$

and similarly for the fake posterior, i.e.,

$$q_\theta(\mathbf{x}_t|\mathbf{y}) \triangleq \int p(\mathbf{x}_t|\mathbf{x})q_\theta(\mathbf{x}|\mathbf{y})\,\mathrm{d}\mathbf{x} = \mathbb{E}_{q_\theta(\mathbf{x}|\mathbf{y})}[p(\mathbf{x}_t|\mathbf{x})]. \tag{9.4}$$

Generator Update. We wish to train a one-step generative model $(\mathbf{z}, \mathbf{y}) \mapsto \mathbf{g}_\theta(\mathbf{z}, \mathbf{y})$. The gradient update for the generator is

$$\gamma_\psi(\theta; \alpha, t, \mathbf{y}) = \mathbb{E}_{q(\mathbf{z})}\left[\nabla_\theta \mathbf{g}_\theta(\mathbf{z}, \mathbf{y}) \frac{\mathbf{s}_\psi(\mathbf{x}_t; 0, t, \mathbf{y}) - \mathbf{s}_\psi(\mathbf{x}_t; \alpha, t, \mathbf{y})}{\alpha}\bigg|_{\mathbf{x}=\mathbf{g}_\theta(\mathbf{z},\mathbf{y})}\right],$$

where we now leverage an amortized score model $\mathbf{s}_\psi(\mathbf{x}_t; \alpha, t, \mathbf{y})$ that is triply amortized over α, t and $\mathbf{y}$.

Amortized Score Estimation. The amortized score model can be characterized by the optimal solution of a mixture of DSM losses

$$\mathcal{L}_{\mathrm{score}}(\psi) := \mathbb{E}_{p(t)p(\alpha)}\Big[\mathcal{L}_{\mathrm{score}}(\psi; \alpha, t)\Big],$$

where

$$\mathcal{L}_{\text{score}}(\psi;\alpha,t) := \mathbb{E}_{p(\mathbf{y})}\Bigg[\alpha\ \mathbb{E}_{p(\mathbf{x}_t|\mathbf{y})}\Big[\|\mathbf{s}_\psi(\mathbf{x}_t;\alpha,t,\mathbf{y}) - \nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{y})\|_2^2\Big] \\ + (1-\alpha)\,\mathbb{E}_{q_\theta(\mathbf{x}_t|\mathbf{y})}\Big[\|\mathbf{s}_\psi(\mathbf{x}_t;\alpha,t,\mathbf{y}) - \nabla_{\mathbf{x}_t}\log q_\theta(\mathbf{x}_t|\mathbf{y})\|_2^2\Big]\Bigg].$$

However, we do not know how to explicitly compute the posterior scores $\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{y})$ and $\nabla_{\mathbf{x}_t}\log q_\theta(\mathbf{x}_t|\mathbf{y})$. As a detour, we apply Tweedie's formula to Eq. (9.3) and Eq. (9.4) and obtain

$$\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{y}) = \mathbb{E}_{p(\mathbf{x}|\mathbf{x}_t,\mathbf{y})}\big[\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x},\mathbf{y})\big] = \mathbb{E}_{p(\mathbf{x}|\mathbf{x}_t,\mathbf{y})}\big[\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x})\big],$$
$$\nabla_{\mathbf{x}_t}\log q_\theta(\mathbf{x}_t|\mathbf{y}) = \mathbb{E}_{q_\theta(\mathbf{x}|\mathbf{x}_t,\mathbf{y})}\big[\nabla_{\mathbf{x}_t}\log q_\theta(\mathbf{x}_t|\mathbf{x},\mathbf{y})\big] = \mathbb{E}_{q_\theta(\mathbf{x}|\mathbf{x}_t,\mathbf{y})}\big[\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x})\big].$$

In both equations, the second equalities follow since $\mathbf{y}-\mathbf{x}-\mathbf{x}_t$ forms a Markov chain under both models $p(\mathbf{x}_t|\mathbf{x})p(\mathbf{x}|\mathbf{y})$ and $p(\mathbf{x}_t|\mathbf{x})q_\theta(\mathbf{x}|\mathbf{y})$. Hence, as we derive the DSM loss, we can instead minimize

$$\begin{aligned}
\mathcal{L}'_{\text{score}}(\psi;\alpha,t) &:= \mathbb{E}_{p(\mathbf{y})}\Bigg[\alpha\ \mathbb{E}_{p(\mathbf{x}_t|\mathbf{y})p(\mathbf{x}|\mathbf{x}_t,\mathbf{y})}\Big[\|\mathbf{s}_\psi(\mathbf{x}_t;\alpha,t,\mathbf{y}) - \nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x})\|_2^2\Big] \\
&\qquad + (1-\alpha)\,\mathbb{E}_{q_\theta(\mathbf{x}_t|\mathbf{y})q_\theta(\mathbf{x}|\mathbf{x}_t,\mathbf{y})}\Big[\|\mathbf{s}_\psi(\mathbf{x}_t;\alpha,t,\mathbf{y}) - \nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x})\|_2^2\Big]\Bigg] \\
&\overset{(a)}{=} \alpha\ \mathbb{E}_{p(\mathbf{x})p(\mathbf{y}|\mathbf{x})p(\mathbf{x}_t|\mathbf{x})}\Big[\|\mathbf{s}_\psi(\mathbf{x}_t;\alpha,t,\mathbf{y}) - \nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x})\|_2^2\Big] \\
&\qquad + (1-\alpha)\,\mathbb{E}_{p(\mathbf{y})q_\theta(\mathbf{x}|\mathbf{y})p(\mathbf{x}_t|\mathbf{x})}\Big[\|\mathbf{s}_\psi(\mathbf{x}_t;\alpha,t,\mathbf{y}) - \nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x})\|_2^2\Big] \\
&\overset{(b)}{=} \alpha\,\mathbb{E}_{p(\mathbf{x})p(\mathbf{y}|\mathbf{x})q(\boldsymbol{\epsilon})}\left[\left\|\mathbf{s}_\psi(\mathbf{x}_t;\alpha,t,\mathbf{y}) + \frac{\boldsymbol{\epsilon}}{\sigma_t}\right\|_2^2\right] \\
&\qquad + (1-\alpha)\,\mathbb{E}_{p(\mathbf{y})q_\theta(\mathbf{x}|\mathbf{y})q(\boldsymbol{\epsilon})}\left[\left\|\mathbf{s}_\psi(\mathbf{x}_t;\alpha,t,\mathbf{y}) + \frac{\boldsymbol{\epsilon}}{\sigma_t}\right\|_2^2\right].
\end{aligned}$$

Here, (a) follows since $p(\mathbf{y})p(\mathbf{x}_t|\mathbf{y})p(\mathbf{x}|\mathbf{x}_t,\mathbf{y}) = p(\mathbf{x})p(\mathbf{y}|\mathbf{x})p(\mathbf{x}_t|\mathbf{x})$ and $q_\theta(\mathbf{x}_t|\mathbf{y})q_\theta(\mathbf{x}|\mathbf{x}_t,\mathbf{y}) = q_\theta(\mathbf{x}|\mathbf{y})p(\mathbf{x}_t|\mathbf{x})$ and (b) follows from the reparameterization trick.

This approach holds promise for efficient SBI, enabling fast and accurate posterior sampling in complex, data-driven settings. By directly learning a one-step generative model conditioned on observations, it bypasses the computational bottlenecks of iterative inference methods. This makes it especially well-suited for applications where real-time decision-making or uncertainty quantification is essential, such as scientific discovery, engineering design, and medical diagnostics.

9.3 Epilogue

In this thesis, we have explored the intersection of score estimation and generative modeling, advancing both theoretical understanding and practical applications. From developing novel score estimation frameworks to introducing new approaches for solving inverse problems and performing one-step generation, we have demonstrated the potential of score-based methods across various domains. While the work presented here represents a significant step forward, it also opens the door to many exciting possibilities for future research. Whether extending these methods to new data modalities or applying them to complex real-world problems, the potential for further exploration is vast. As the tools this thesis is built on continue to evolve, we hope that the contributions of this thesis will inspire future work on the next generation of scalable, efficient, and innovative solutions that leverage generative models.

Bibliography

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. In *CVPR*.

Ascher, U. M. and Petzold, L. R. (1998). *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM.

Baker, C. (1977). *The Numerical Treatment of Integral Equations*. Monographs on Numerical Analysis. Clarendon Press.

Baldassarre, L., Rosasco, L., Barla, A., and Verri, A. (2012). Multi-Output Learning via Spectral Filtering. *Machine Learning*, 87(3):259–301.

Ballé, J., Chou, P. A., Minnen, D., Singh, S., Johnston, N., Agustsson, E., Hwang, S. J., and Toderici, G. (2020). Nonlinear Transform Coding. *IEEE Journal of Selected Topics in Signal Processing*, 15(2):339–353.

Berthelot, D., Autef, A., Lin, J., Yap, D. A., Zhai, S., Hu, S., Zheng, D., Talbott, W., and Gu, E. (2023). TRACT: Denoising Diffusion Models with Transitive Closure Time-Distillation. *arXiv Preprint arXiv:2303.04248*.

Brock, A., Donahue, J., and Simonyan, K. (2019). Large Scale GAN training for High Fidelity Natural Image Synthesis. In *International Conference on Learning Representations*.

Bromiley, P. (2003). Products and Convolutions of Gaussian Probability Density Functions. *Tina-Vision Memo*, 3(4):1.

Chandna, P., Miron, M., Janer, J., and Gómez, E. (2017). Monoaural Audio Source Separation using deep Convolutional Neural Networks. In *Latent Variable Analysis and Signal Separation: 13th International Conference, LVA/ICA 2017, Grenoble, France, February 21-23, 2017, Proceedings 13*, pages 258–266. Springer.

Che, T., Zhang, R., Sohl-Dickstein, J., Larochelle, H., Paull, L., Cao, Y., and Bengio, Y. (2020). Your GAN is Secretly an Energy-Based Model and you should use Discriminator Driven Latent Sampling. *Advances in Neural Information Processing Systems*, 33:12275–12287.

Chung, H., Kim, J., Kim, S., and Ye, J. C. (2023). Parallel Diffusion Models of Operator and Image for Blind Inverse Problems. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6059–6069.

Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. (2022a). Diffusion Posterior Sampling for General Noisy Inverse Problems. *arXiv preprint arXiv:2209.14687*.

Chung, H., Sim, B., and Ye, J. C. (2022b). Improving Diffusion Models for Inverse Problems Using Manifold Constraints. *Advances in Neural Information Processing Systems*.

Chung, H. and Ye, J. C. (2022). Score-Based Diffusion Models for Accelerated MRI. *Medical Image Analysis*, 80:102479.

Cranmer, K., Brehmer, J., and Louppe, G. (2020). The Frontier of Simulation-Based Inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062.

Csiszár, I., Shields, P. C., et al. (2004). Information Theory and Statistics: A Tutorial. *Found. Trends Commun. Inf. Theory*, 1(4):417–528.

Damnjanovic, A., Montojo, J., Wei, Y., Ji, T., Luo, T., Vajapeyam, M., Yoo, T., Song, O., and Malladi, D. (2011). A Survey on 3GPP Heterogeneous Networks. *IEEE Wireless Communications*, 18(3):10–21.

De Vito, E., Umanità, V., and Villa, S. (2013). An Extension of Mercer Theorem to Matrix-Valued Measurable Kernels. *Appl. Comput. Harmon. Anal.*, 34(3):339–351.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Dhariwal, P. and Nichol, A. (2021). Diffusion Models Beat GAns On Image Synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794.

Eldar, Y. C., Goldsmith, A., Gündüz, D., and Poor, H. V. (2022). *Machine Learning and Wireless Communications*. Cambridge University Press.

Elfwing, S., Uchibe, E., and Doya, K. (2018). Sigmoid-weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning. *Neural Networks*, 107:3–11.

Frank, M. and Ilse, M. (2020). Problems Using Deep Generative Models for Probabilistic Audio Source Separation. In *"I Can't Believe It's Not Better!" NeurIPS 2020 Workshop*.

Gandelsman, Y., Shocher, A., and Irani, M. (2019). Double-DIP: Unsupervised Image Decomposition via coupled Deep-Image-Priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11026–11035.

Geng, Z., Pokle, A., Luo, W., Lin, J., and Kolter, J. Z. (2024). Consistency Models Made Easy. *arXiv preprint arXiv:2406.14548*.

Gloeckler, M., Deistler, M., Weilbach, C., Wood, F., and Macke, J. H. (2024). All-in-One Simulation-Based Inference. *arXiv preprint arXiv:2404.09636*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, volume 27.

Gorham, J. and Mackey, L. (2015). Measuring Sample Quality with Stein's Method. In *Advances in Neural Information Processing Systems*, volume 28.

Gouillart, E., Krzakala, F., Mezard, M., and Zdeborová, L. (2013). Belief-Propagation Reconstruction for Discrete Tomography. *Inverse Problems*, 29(3):035003.

Grünwald, P. (2012). The Safe Bayesian: Learning the Learning Rate via the Mixability Gap. In *Algorithmic Learning Theory: 23rd International Conference, ALT 2012, Lyon, France, October 29–31, 2012. Proceedings 23*, pages 169–183. Springer.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Heath Jr, R. W. (2017). *Introduction to Wireless Digital Communication: A Signal Processing Perspective*. Prentice Hall.

Heek, J., Hoogeboom, E., and Salimans, T. (2024). Multistep Consistency Models.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*, volume 30.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851.

Holmes, C. C. and Walker, S. G. (2017). Assigning a Value to a Power Likelihood in a General Bayesian Model. *Biometrika*, 104(2):497–503.

Hoogeboom, E., Heek, J., and Salimans, T. (2023). Simple Diffusion: End-to-end Diffusion for High Resolution Images. In *Proceedings of the International Conference on Machine Learning*, pages 13213–13232. PMLR.

Hoydis, J., Cammerer, S., Ait Aoudia, F., Vem, A., Binder, N., Marcus, G., and Keller, A. (2022). Sionna: An Open-Source Library for Next-Generation Physical Layer Research. *arXiv preprint*.

Hutchinson, M. F. (1989). A Stochastic Estimator of the Trace of the Influence Matrix for Laplacian Smoothing Splines. *Commun. Stat. - Simul. Comput.*, 18(3):1059–1076.

Hyvärinen, A. (2005). Estimation of Non-Normalized Statistical Models by Score Matching. *JMLR*, 6(4).

Jayaram, V. and Thickstun, J. (2020). Source Separation with deep Generative Priors. In *International Conference on Machine Learning*, pages 4724–4735. PMLR.

Jayashankar, T., Lee, G. C., Lancho, A., Weiss, A., Polyanskiy, Y., and Wornell, G. (2023). Score-Based Source Separation with Applications to Digital Communication Signals. In *Advances in Neural Information Processing Systems*, volume 36, pages 5092–5125.

Jayashankar, T., Ryu, J. J., and Wornell, G. (2025). Score-of-Mixture Training: Training One-Step Generative Models Made Simple via Score Estimation of Mixture Distributions.

Jayashankar, T., Ryu, J. J., Xu, X., and Wornell, G. W. (2024). Lifted Residual Score Estimation. In *ICML 2024 Workshop on Structured Probabilistic Inference and Generative Modeling*.

Jayashankar, T. K. (2022). Image Compression using Sum-Product Networks. Master's thesis, Massachusetts Institute of Technology.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. (2021). Highly Accurate Protein Structure Prediction with AlphaFold. *nature*, 596(7873):583–589.

Kadkhodaie, Z. and Simoncelli, E. (2021). Stochastic Solutions for Linear Inverse Problems Using the Prior Implicit in a Denoiser. *Advances in Neural Information Processing Systems*, 34:13242–13254.

Karras, T., Aittala, M., Aila, T., and Laine, S. (2022). Elucidating the Design Space of Diffusion-based Generative Models. In *Advances in Neural Information Processing Systems*, volume 35, pages 26565–26577.

Karras, T., Laine, S., and Aila, T. (2021). A Style-Based Generator Architecture for Generative Adversarial Networks. *TPAMI*, 43(12):4217–4228.

Kawar, B., Elad, M., Ermon, S., and Song, J. (2022). Denoising Diffusion Restoration Models. In *Advances in Neural Information Processing Systems*.

Khandekar, A., Bhushan, N., Ji, T., and Vanghi, V. (2010). LTE-Advanced: Heterogeneous Networks. In *2010 European Wireless Conference (EW)*, pages 978–982. IEEE.

Kim, D., Lai, C.-H., Liao, W.-H., Murata, N., Takida, Y., Uesaka, T., He, Y., Mitsufuji, Y., and Ermon, S. (2024). Consistency Trajectory Models: Learning Probability Flow ODE Trajectory of Diffusion. In *International Conference on Learning Representations*.

Kingma, D. and Gao, R. (2024). Understanding Diffusion Objectives as the ELBO with simple Data Augmentation. In *Advances in Neural Information Processing Systems*, volume 36.

Kingma, D., Salimans, T., Poole, B., and Ho, J. (2021). Variational Diffusion Models. In *Advances in Neural Information Processing Systems*, volume 34, pages 21696–21707.

Kingma, D. and Welling, M. (2014). Auto-encoding Variational Bayes. In *International Conference on Learning Representations*.

Kong, X., Brekelmans, R., and Steeg, G. V. (2023). Information-theoretic Diffusion. In *International Conference on Learning Representations*.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. (2021). Diffwave: A Versatile Diffusion Model for Audio Synthesis. In *International Conference on Learning Representations*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning Multiple Layers of Features from Tiny Images. Technical report, U. Toronto.

Kulikov, V., Yadin, S., Kleiner, M., and Michaeli, T. (2022). SinDDM: A Single Image Denoising Diffusion Model. *arXiv preprint arXiv:2211.16582*.

Lancho, A., Weiss, A., Lee, G. C., Jayashankar, T., Kurien, B., Polyanskiy, Y., and Wornell, G. W. (2024). RF Challenge: The Data-Driven Radio Frequency Signal Separation Challenge. *arXiv preprint arXiv:2409.08839*.

Lapidoth, A. (2017). *A Foundation in Digital Communication*. Cambridge University Press.

Le Cam, L. (2012). *AsymptoticMmethods in Statistical Decision Theory*. Springer Science & Business Media.

LeCun, Y. (1998). The MNIST Database of Handwritten Digits. *http://yann.lecun.com/exdb/mnist/*.

Lee, G. C., Weiss, A., Lancho, A., Tang, J., Bu, Y., Polyanskiy, Y., and Wornell, G. W. (2022). Exploiting Temporal Structures of Cyclostationary Signals for Data-Driven Single-Channel Source Separation. In *2022 IEEE 32nd International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE.

Li, Y. and Turner, R. E. (2018). Gradient Estimators for Implicit Models. In *ICLR*.

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. (2023). Flow Matching for Generative Modeling. In *International Conference on Learning Representations*.

Liu, X., Gong, C., and Liu, Q. (2023). Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. In *International Conference on Learning Representations*.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep Learning Face Attributes in the Wild. In *ICCV*.

Loshchilov, I. and Hutter, F. (2017). SGDR: Stochastic Gradient Descent with Warm Restarts. In *International Conference on Learning Representations*.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. (2022). DPM-solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps. In *Advances in Neural Information Processing Systems*, volume 35, pages 5775–5787.

Luo, W., Hu, T., Zhang, S., Sun, J., Li, Z., and Zhang, Z. (2024a). Diff-Instruct: A Universal Approach for Transferring Knowledge from Pre-Trained Diffusion Models. In *Advances in Neural Information Processing Systems*, volume 36.

Luo, W., Huang, Z., Geng, Z., Kolter, J. Z., and Qi, G.-J. (2024b). One-step Diffusion Distillation through Score Implicit Matching. In *Advances in Neural Information Processing Systems*.

Lutati, S., Nachmani, E., and Wolf, L. (2023). Separate and Diffuse: Using a Pretrained Diffusion Model for Improving Source Separation. *arXiv preprint arXiv:2301.10752*.

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Paul Smolley, S. (2017). Least Squares Generative Adversarial Networks. In *CVPR*, pages 2794–2802.

Mariani, G., Tallini, I., Postolache, E., Mancusi, M., Cosmo, L., and Rodolà, E. (2023). Multi-Source Diffusion Models for simultaneous Music Generation and Separation. *arXiv preprint arXiv:2302.02257*.

Milanfar, P. and Delbracio, M. (2024). Denoising: A Powerful Building-Block for Imaging, Inverse Problems, and Machine Learning. *arXiv preprint arXiv:2409.06219*.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral Normalization for Generative Adversarial Networks. In *International Conference on Learning Representations*.

Nash, C., Menick, J., Dieleman, S., and Battaglia, P. W. (2021). Generating Images with sparse Representations. In *ICML*.

Nichol, A. Q. and Dhariwal, P. (2021). Improved Denoising Diffusion Probabilistic Models. In *Proceedings of the International Conference on Machine Learning*, pages 8162–8171. PMLR.

Nielsen, F. (2010). A Family of Statistical Symmetric Divergences based on Jensen's Inequality. *arXiv preprint arXiv:1009.4004*.

Nokia (2023). 6G Explained. Accessed: 2025-03-04.

Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In *Advances in Neural Information Processing Systems*, volume 29.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio. *arXiv preprint arXiv:1609.03499*.

OpenAI (2022). ChatGPT: Optimizing Language Models for Dialogue. Accessed: 2025-03-04.

Oyedare, T., Shah, V. K., Jakubisin, D. J., and Reed, J. H. (2022). Interference Suppression Using Deep Learning: Current Approaches and Open Challenges. *IEEE Access*, 10:66238–66266.

Perrotta, L. (2020). Practical Calibration of the Temperature Parameter in Gibbs Posteriors. *arXiv preprint arXiv:2004.10522*.

Polyanskiy, Y. and Wu, Y. (2019). Lecture Notes on Information Theory.

Poole, B., Jain, A., Barron, J. T., and Mildenhall, B. (2022). Dreamfusion: Text-to-3D using 2D diffusion. *arXiv preprint arXiv:2209.14988*.

Ramachandran, P., Zoph, B., and Le, Q. V. (2018). Searching for Activation Functions. In *ICLR*.

Rezende, D. and Mohamed, S. (2015). Variational Inference with Normalizing Flows. In *Proceedings of the International Conference on Machine Learning*, pages 1530–1538. PMLR.

Robbins, H. E. (1956). An Empirical Bayes Approach to Statistics. *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability*.

Rout, L., Raoof, N., Daras, G., Caramanis, C., Dimakis, A., and Shakkottai, S. (2023). Solving Linear Inverse Problems Provably via Posterior Sampling with Latent Diffusion Models. *Advances in Neural Information Processing Systems*, 36:49960–49990.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved Techniques for Training GANs. In *NeurIPS*, volume 29.

Salimans, T. and Ho, J. (2022). Progressive Distillation for Fast Sampling of Diffusion Models. In *International Conference on Learning Representations*.

Salimans, T., Mensink, T., Heek, J., and Hoogeboom, E. (2024). Multistep Distillation of Diffusion Models via Moment Matching. In *Advances in Neural Information Processing Systems*.

Samuel, N., Diskin, T., and Wiesel, A. (2019). Learning to Detect. *IEEE Transactions on Signal Processing*, 67(10):2554–2564.

Sauer, A., Schwarz, K., and Geiger, A. (2022). Stylegan-XL: Scaling Stylegan to Large Diverse Datasets. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10.

Shi, J., Sun, S., and Zhu, J. (2018). A Spectral Approach to Gradient Estimation for Implicit Distributions. In *ICML*, pages 4644–4653. PMLR.

Shlezinger, N., Farsad, N., Eldar, Y. C., and Goldsmith, A. J. (2020a). ViterbiNet: A Deep Learning Based Viterbi Algorithm for Symbol Detection. *IEEE Transactions on Wireless Communications*, 19(5):3319–3331.

Shlezinger, N., Fu, R., and Eldar, Y. C. (2020b). Deep Soft Interference Cancellation for MIMO Detection. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8881–8885.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *Proceedings of the International Conference on Machine Learning*, pages 2256–2265. PMLR.

Song, J., Meng, C., and Ermon, S. (2021a). Denoising Diffusion Implicit Models. In *International Conference on Learning Representations*.

Song, J., Vahdat, A., Mardani, M., and Kautz, J. (2023a). Pseudoinverse-Guided Diffusion Models for Inverse Problems. In *International Conference on Learning Representations*.

Song, Y. and Dhariwal, P. (2024). Improved Techniques for Training Consistency Models. In *International Conference on Learning Representations*.

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. (2023b). Consistency Models. In *Proceedings of the International Conference on Machine Learning*, volume 202, pages 32211–32252. PMLR.

Song, Y. and Ermon, S. (2019). Generative Modeling by Estimating Gradients of the Data Distribution. In *Advances in Neural Information Processing Systems*, volume 32.

Song, Y., Garg, S., Shi, J., and Ermon, S. (2020). Sliced Score Matching: A Scalable Approach to Density and Score Estimation. In *UAI*, pages 574–584. PMLR.

Song, Y., Shen, L., Xing, L., and Ermon, S. (2022). Solving Inverse Problems in Medical Imaging with Score-Based Generative Models. In *International Conference on Learning Representations*.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021b). Score-based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021c). Score-based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*.

Stein, C. M. (1981). Estimation of the Mean of a Multivariate Normal Distribution. *AnnStat*, pages 1135–1151.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schölkopf, B. (2018). Wasserstein Autoencoders. In *ICLR*.

Tse, D. and Viswanath, P. (2005). *Fundamentals of Wireless Communication*. Cambridge University Press.

Tzinis, E., Wang, Z., and Smaragdis, P. (2020). SuDoRM-RF: Efficient networks for universal Audio Source Separation. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE.

Vaswani, A. (2017). Attention is All You Need. In *Advances in Neural Information Processing Systems*.

Vincent, P. (2011). A Connection Between Score Matching and Denoising Autoencoders. *Neural Computation*, 23(7):1661–1674.

Wang, Z., Zheng, H., He, P., Chen, W., and Zhou, M. (2023). Diffusion-GAN: Training GANs with Diffusion. In *International Conference on Learning Representations*.

Williams, C. and Seeger, M. (2000). Using the Nyström Method to speed up kernel Machines. In *Advances in Neural Information Processing Systems*, volume 13. MIT Press.

Xu, Y., Gui, G., Gacanin, H., and Adachi, F. (2021). A Survey on Resource Allocation for 5G Heterogeneous Networks: Current Research, Future Trends, and Challenges. *IEEE Communications Surveys & Tutorials*, 23(2):668–695.

Yin, T., Gharbi, M., Park, T., Zhang, R., Shechtman, E., Durand, F., and Freeman, W. T. (2024a). Improved Distribution Matching Distillation for Fast Image Synthesis. In *Advances in Neural Information Processing Systems*.

Yin, T., Gharbi, M., Zhang, R., Shechtman, E., Durand, F., Freeman, W. T., and Park, T. (2024b). One-Step Diffusion with Distribution Matching Distillation. In *CVPR*.

Zhang, T. (2003). Learning Bounds for a Generalized Family of Bayesian Posterior Distributions. *Advances in Neural Information Processing Systems*, 16.

Zhou, M., Zheng, H., Wang, Z., Yin, M., and Huang, H. (2024). Score Identity Distillation: Exponentially Fast Distillation of Pretrained Diffusion Models for One-Step Generation. In *Proceedings of the International Conference on Machine Learning*.

Zhou, Y., Shi, J., and Zhu, J. (2020). Nonparametric Score Estimates. In *ICML*, pages 11513–11522. PMLR.

Zou, Z., Lei, S., Shi, T., Shi, Z., and Ye, J. (2020). Deep Adversarial Decomposition: A Unified Framework for Separating Superimposed Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12806–12816.